Chapter 1

Introduction

1.1 Background

Now days our dependence on web application has increased and we continue to integrate them in our everyday routine activities. Web applications are reliable in online shopping, reading newspaper, paying bills etc. Many services are provided via the world wide web, efforts from both academic and industry are striving to create techniques and standards that meets the sophisticated requirement's of today's web applications and users. From the hackers' perspective, web application is divided in to the several layers. Current technologies such as antivirus software program and network firewalls offer some security at the host and network levels but not at the application level. Although the application level firewalls offer immediate assurance of web application security, they have at least two drawbacks: they required careful configuration and they only offer web application protection, i.e., they do not identify errors. Similarly SSL and data encryption only secure the data transmission, but not the traffic.

In many situations, security remains a major roadblock to universal acceptance of the web for all kinds of transactions. According to a report released earlier in 2004, over the preceding 12 months in 2002, there was an 81.5% increase in documented vulnerabilities, with the majority associated with a handful of very severe vulnerability [24]. Vulnerabilities of web application occur due to different reasons. Some of them are input invalidation, parameter manipulation etc. The recognitions of this problem are reflected by the recent burst of effort that aims to improve web application security via numerous different approaches. Scott and sharp[35] proposed the use of a gateway that filters invalid and malicious input at the application level. Most of the leading firewall vendors are also using deep packet inspection [29] technologies in their attempts to filter application level traffic. Huang et al.[24] designed a web application security assessment

framework that offers black-boxed testing for identifying web application vulnerabilities. However, testing process cannot guarantee identification of all bugs, and cannot provide immediate security for web application.

1.2 Problem

Different researchers have published different papers in the field of web application and its security mechanism. According to Spett [38], from the hacker's perspective, web application consists of five layers, i.e., desktop layer, transport layer, access layer, network layer and application layer. Spett [38] further mentions that SQL injection attack occurs as well as can be prevented at the application layer. Current technologies such as anti-virus software program and network firewall offer comparatively secure protection at the host and network levels, but not at the application level. Similarly, Huang et al. [24] designed a web application security assessment framework that offers black-box testing for identifying web application vulnerabilities. However, testing processes cannot guarantee identification of all bugs. Scott and Sharp [35] proposed the use of a gateway that filters invalid and malicious inputs at the application level. According to the Gartner report over 300 web sites such found that 97% of the sites audited were vulnerable to this SQL injection attack. Then as this report corporations, security professionals and hackers continue to announce that SQL Injection vulnerabilities are inherent in web applications and reports of compromised applications are frequently published. But when we studied different papers about SQL Injection we conclude that in the attacking filed there are no actual systematic method which helps us to protect our application. Then in this work we studied different attacking method & their prevention technique and implement them, among the different technique for preventing web application through SQL-Injection attack we purpose a transition table validation method, we believe that when we follow this idea we really secure our application in some aspect.

1.3 Objective and Outline

Objective of this thesis work is to study the SQL Injection attack in the web application with their prevention technique and then implement them in the ASP.NET. Especially objective of this thesis is categorized as follows points.

1. To show how SQL statements can be injected and attacked.
2. To show how the injected query can be prevented from the executing on server.
3. To show the execution overhead for normal and guarded statements.

In this dissertation work there are six chapters

Chapter 2 is the discussion on common web application including their architecture.

In Chapter 3, we cover various aspects of computer security.

In Chapter 4, we briefly discuss the Database concept required for the web application and security issues which are related to them.

In Chapter 5, we discuss SQL Injection attack in web application including prerequisites, method and defense techniques, execution time overhead and validate our technique in .NET platform.

Chapter 6 is our last chapter where we discuss conclusion and further work in our subject matter.

1.4 Literature review

Use of web application is increasing rapidly everywhere. In the field of that, numerous research works is published whether it may be in paper or internet. We already discussed that where web application is applicable in background. Companies and organizations use web applications to provide a broad range of service to users. Database of web application contain the important information of that organization like customer and financial record, these applications are frequent target for attacks. There are numerous attacking technique one of them is SQL Injection, can give attackers a way to gain access

to the database underlying web application and with that power to leak, modify and in some time delete information that is stored in the database. In the other word, SQL-Injection attacks can occur when a web application receive user input and use it to build a database query without validating it. Conceptually, SQLIAs could be prevented by a more rigorous application of defensive coding techniques.

To prevent the SQLIAs, different researcher present the different view; like Buehrer et al. [8] secure vulnerable SQL statements by comparing the parse tree of a SQL statement before and after input and only allowing a SQL statements to execute if the parse trees match. Here if a malicious user successfully injects SQL in to a database query, the parse tree of the intended SQL query and the resulting SQL query do not match. Huang et al. [24] potential vulnerabilities can be secured by combining static analysis with runtime monitoring. They present WebSSARI tool, which analyzes source code, finds potential vulnerabilities, including SQLIVs and inserts runtime guards into the source code, which sanitize input.

Halfond and Orso [21] secure vulnerabilities statements by combining static analysis with statement generation and runtime monitoring. These two researcher design a new tool i.e. statement and delete the illegal queries. The main purpose of this tool is to deleting and preventing SQL injection attack. SQLIAs performed through ports (i.e. port 80 and 443) used for regular web traffic, which are usually open in firewalls and work at application layer. There are very few analysis based technique for vulnerabilities that target SQLIAs directly and they provide a partial solution to problem.

Chapter 2

Web Application

2.1 Introduction

According to the definition of OWASP [33], “web application is a client/server software application that interacts with users or other systems using Hyper Text Transfer Protocol (HTTP)”. Web application can take many forms-an informational website, an e-commerce website, a search engine, a transaction engine, an e-business. When web pages were first implemented they only displayed unchanging information, which is called static contents. They may have included hyperlinks that would point to other web pages, which made it easy for users to get information from the multiple or related web sites. These web sites did not offer any interactively with users and did not required user input. But such type of web application did not able to fulfill users requirement then various programming language were developed that allows newer web page to include text box, radio button, drop down list etc so that user can insert data. These web sites that change consist of dynamic web contents thus it also called interactive. All of these applications are linked to a computer system that contains weakness that can pose risk to a company. Weakness exists in system architecture, system configuration and operations. The risk includes the possibility of incorrect calculations, damaged hardware and software; data accessed by unauthorized users, data theft or loss, misuse of the system, and disrupted business operations[27].

As a digital enterprise embraces the benefits of e-business, the use of web based technology will continue to grow. Corporation today use the web as a way to manage their customer relationship, enhance their supply chain operations, expand into new markets, deploy new products and services to customer & employees. However,

successfully implementing the powerful benefits of web based technology cannot be achieved without a consistent approach to web application security.

Due to use of web application, number of business move to take advantage of the internet and they discover that the web is also new operating environment. In this new environment, conventional security measures are outdated and frequently inefficient. A new level of security breach has begun to occur through continuously open internet port i.e. port 80 and 443. Because these ports are open to all incoming internet traffic from the outside, they are gateways through which hackers' access secure files and proprietary corporate and customer data. While dishonest hackers make the news, there exits a much more likely thereat in the form of online theft, terrorism. In addition to the vulnerabilities inherent in the new internet operating environment, negligence also accounts for a portion of the risk to a company's data. According to the SANS institute, following are the management errors that lead to computer security vulnerabilities [37].

1. Pretending the problem will go away.
2. Assigning untrained people to maintain security and not providing the training or the time to make it possible to do the job.
3. Failing to realize how much money their information and organization reputations are worth.
4. Relying primarily on a firewall and IDS.
5. Failing to understand the relationship of information security with the business problem-they understand physical security but don't see the consequences of poor information security.

When we develop their security policies and implement basic security foundation, the professional hackers continues to find new ways to attack. Most hackers are use different technique to gain escalated privileges, or executed commands. Attackers on web-connected server are becoming common. Everyday there is news of another major corporation whose security has been opened. Attacker stole credit card number, stealing sensitive quest information to protect the web application from the attackers only the password, data encryption is not enough.

2.2 Web Application Vulnerability Statistics

Before deeply enter in our thesis work, we think it is needs to presents statistics that show how common various categories of vulnerabilities are. There are numbers of web sites which gives the information, data etc about the security of web application. Here, maximum number of data is taken from those sites which are publicly available. Web applications Vulnerabilities have a short but an illustrious history. Below table shows the first time each type of vulnerabilities was discovered. We can expect new types of Vulnerabilities also occur in the future.

Month	Year	Vulnerabilities
February	2000	Cross site script
October	2001	Path traversal
May	2001	SQL Injection
February	2002	Cookie poisoning
March	2003	Cross site tracing
March	2004	HTTP response splitting
June	2005	HTTP request smuggling
July	2005	DOM-based cross site scripting
June	2006	Domain contamination

Table1: History of common web application vulnerabilities.

2.2.1 NIST Study

The National Institute of Standards and Technology (NIST) and the Department of Homeland Security have been aggregating Vulnerability data for many years. Statistics summarizing the total number of vulnerabilities is the NIST database is present in below Fig1. These studies collect the record from the year 1999 to mid 2006. This chart shows that number of vulnerabilities is increase continuously from the 2004 to till 2006. But it shows that application vulnerabilities are decrease in year 2000 and 2004. It is difficult to give the reason why vulnerabilities are decrease in these two years.

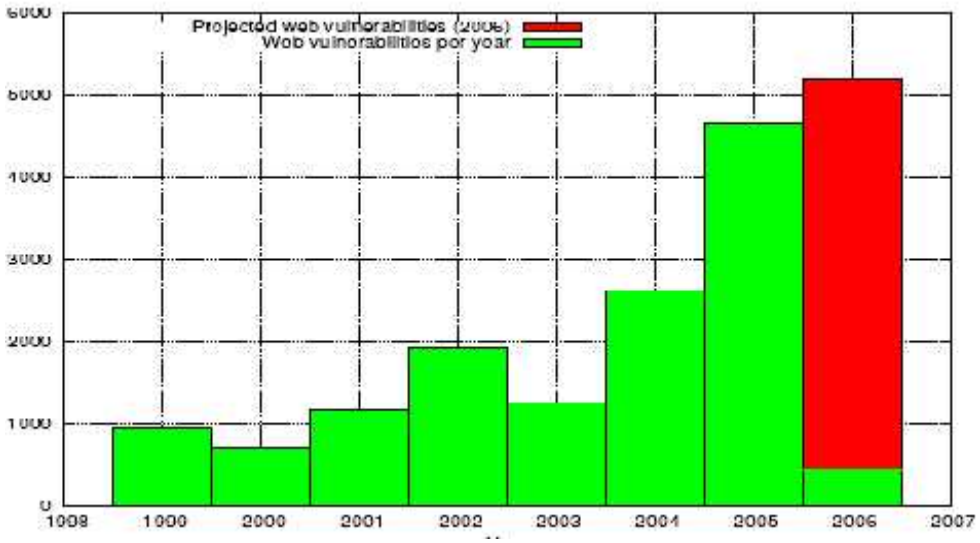


Figure: 1. Number of vulnerabilities reported by year (based on NIST/DHS data) [34]

2.2.2 Study on Web Sites

To gain insight into the relative frequencies of different vulnerability types, we study the various security web sites and research paper from this study we found that main reason for vulnerability of web application is occur due to the lack of input validation. Which is also illustrated by the Fig 2. According to the securityfocus.com in sample of 500 vulnerability 50% of them are due to the reason of input validation vulnerabilities.

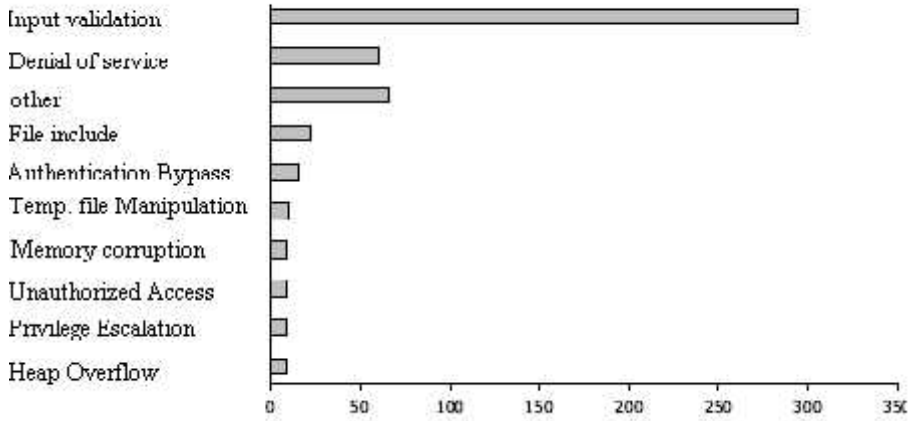


Figure: 2. Relative frequency of vulnerabilities in the SecurityFocus.com[31]

Focusing on input validation vulnerabilities shows that most are vulnerabilities specific to Web applications, as can be seen from the Figure 3. This shows that most of the web application attack occurs due to the SQL Injection attack and after that cross site scripting is second reason and so on.

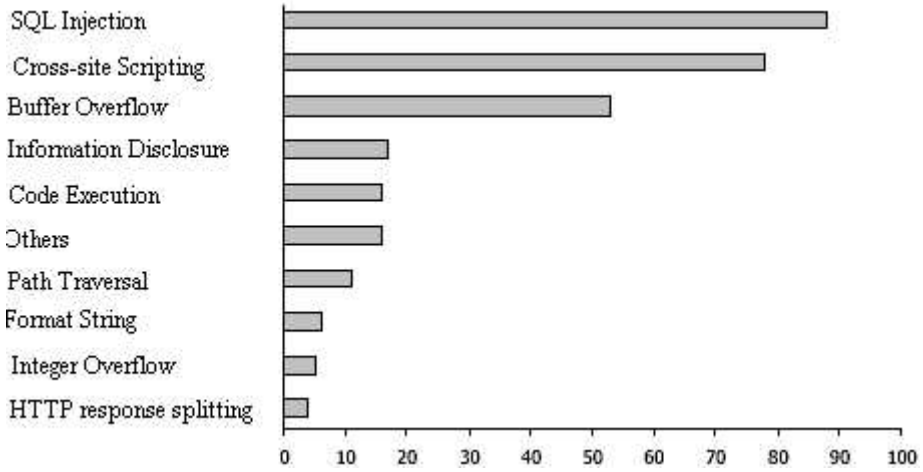


Figure: 3. Relative frequencies of input/output validations vulnerabilities [31]

2.3 Common Web Application Attack Type

Below the list of some of the most common web attack type

SQL Injection: A security vulnerability that occurs in the database larger of an application. The basic idea behind SQL Injection attack is abusing web pages which allow users to enter text in form of fields, which are used for database queries. Hackers can enter a disguised SQL query, which changes the nature of the intended query. Hence the queries can be used to access the connected database and change or delete its data. There are a big number of internet systems which are vulnerable to this kind of attack [27]. SQL query generated by concatenation of the static part of the query and values intended is the base of this attack. (Discussed detail in later).

Cross- site scripting: CSS or XSS is possible with dynamic pages showing the input that is not properly validated. This allows the hacker to inject the malicious JavaScript code in a generated page and execute the script on a computer of any visitor of that web site. Cross- site scripting could potentially impact any site that allows users to enter data[13].

In another word, cross-site scripting is an attack technique that forces a website to echo attacker supplied executable malicious code, which load in user's browser.

XSS can generally be divided into two categories: reflected and stored attack. Reflected XSS is found on a page which reflected user's input sting directly back to the user.

Consider if one enter a string of JavaScript, like

<SCRIPT>alert("XSS vulnerability test")</SCRIPT>

The generated page will contain the JavaScript and will be executed on the user's web browser.

In stored XSS when malicious code is injected to a web site, it stored over a longer period, and displayed to users in a web page later. This kind of XSS is more serious than other types, because an attacker can inject malicious code just once and affect a larger number of unsuspecting users, it is even hardly necessary for attacker to trick the users in to clicking on a link containing malicious code. For example, if the

malicious code is stored in a database, without clicking on any link, the innocent user may become victim by just viewing the page that contains the stored malicious code.

Parameter tempering: Manipulation of the data within a parameter transmitted from one web page to another. It can achieve in the following ways.

1. URL manipulation: Here parameters are appended as query string to the URL, if the HTTP method GET is used instead of POST. Data within a parameter can be easily manipulated in the query string, because it is alterable in the address bar of a web browser. After the modification new parameter will be send to the web application and be proceed correspondingly.

2. Cookie poisoning: Cookies are small amount of data transmitted between web server and web client used for authentication and remembering users' performance. Using cookies, user may not have to type their login information, so that quick login can be achieved. It store sensitive information such as accounts and passwords, they are usually aimed for security attack. Because of this sensitive information, cookies poisoning is more dangerous than other parameter tempering attacks. By tampering with the values contained in the cookie, attackers are capable of authenticating themselves to a web site fraudulently and thus gain unauthorized information and perform action on behalf of the victim.

2.4 Anatomy of a Web Application Attack

At the time of performing attack on web application hacker does various steps of work some of them are list below [38].

Scan: The hacker starts by running a port scan to detect the open HTTP and HTTPS ports for each server retrieving the default page from each open port.

Information Gathering: The hacker then identifies the type of server running on each port, and each page is parsed to find normal links. This enables the hacker to determine the structure of the site and logic of the application. Then the attacker analyzes the found

pages and checks for comments and other possibly useful bits of data that could refer to files and directories that are not intended public use.

Testing: The hacker goes through a testing process for each of the application scripts or dynamic functions of the application, looking for development errors to enable him to gain further access into the application.

Planning the Attack: When the hacker has identified every bit of information that can be gathered by passive (undetectable) means, he selects and deploys attacks. These attacks center on the information gained from the passive information gathering process.

Launching the Attack: After all of these above procedures, the hacker engages in open warfare by attacking each web application that he identified as vulnerable during the initial review of our site.
The results of the attack could be lost data, content manipulation, or even theft and loss of customers. The average corporation does not have the mechanisms to detect such attack and can spend significant resources just trying to diagnose the implication of an attack.

The potential for loss is significant. A hacker could easily copy sensitive corporate information, such as proprietary customer databases or records, and disseminate that information to competitors, or even to the general public, without our knowledge.

2.5 Architecture

Architecture is the structure of web application. From the hacker's perspective, a corporation's web applications can be viewed as a horizontal value chain of layers [38]. Web application architecture is mainly concerned with that part where different task are performed. Thus here we described the client-server architecture.

2.5.1 Client-Server Architecture

Client-server is the term which denotes to that processes with which different components of software are interacts to form a system. In more clearly, client system needs resource which are obtained from the server processes.

In web application client and server have different meaning: client represents the different web browser for example Opera, Mozilla Firefrox, Internet explorer, Netscape Navigator and so on. And server represents the web server some of them are Apache, Tomcat, Microsoft Internet Information Server etc [6]. The combination of different client-server architecture can be called topology which consist the a) single client, single server b)multiple client, single server c) single client ,multiple server and d) multiple client, multiple server.

Client –Server architecture may be two tier or three tier. Tier is defined as one of two or more rows, and ranks arranged one above another.

Two-tiered architecture

Connolly et .al[11] mention the two-tier client-server architecture as the basic model for separating tasks, client constitute the first tier and server the second tier. A client is primarily responsible for presentation services, including handling user interface actions, performing application logic and presentation of data to user and performing main business application logic. The server is primarily concerned with supplying data services to the client.

Three-tiered architecture

Web applications are generally structured as three-tiered which consists of:

1. The *first tier* is a web browser such as internet Explorer, Mozilla Firefox and Netscape, etc.
2. The *middle tier* is an engine, which generates pages dynamically using technologies such as hypertext processor (PHP), Active Server Pages technology (ASP), and Java Server Pages technology (JSP).
3. The *third tier* is a database; it enables Web applications to store data and other content elements. By using the Structured Query Language (SQL), Web

applications can interact with database to create customized data for each user dynamically.

As illustrated in Figure4 (b), a client (web browser) send request to the middle tier, which handles these requests, searches information required by making SQL queries against the database and generate response pages using this information, and shows them to user in the browser.

a) Static Web Sites

b) Dynamic Web Applications

Figure: 4.Static Web Sites VS. Dynamic Web Application

N-tier architecture

This means that there are any numbers of tiers. Simple example is when several web server and database server resides on separate computer. Another example is when

several database server is used and one computer is dedicated responsible of managing to access to each database server, running in separate computer.

2.6 Web Application Model

Generally architecture of web application is seen as follow figure

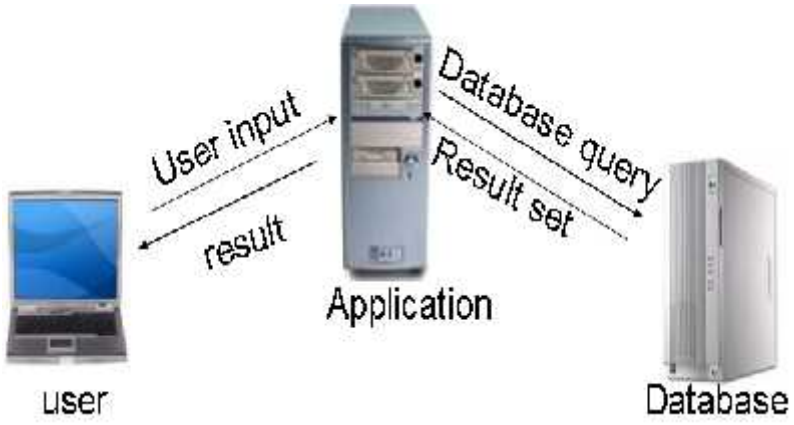


Figure: 5. typical system architecture for web application

Web application is normally three tier architecture. It includes the following components.

Presentation tier

The presentation tier, which very commonly referred to the "font end", is the portion of the application that user interacts with directly and normally displayed by software known as browser. Presentation tier consist of different web browser like Mozilla Firefox, Opera, Internet Explorer etc and web server such as Microsoft IIS, Apache etc.

presentation tier take request from user i.e. Client- side and process them after that it send the required information of user to the web-browser in the form of output.

Middle tier

The middle tier also known as the business tier, encapsulate the business logic that derives the application. This software layer is responsible for constructing information requests to the database layer. Middle tier generates a SQL request based on input supplied by the user [8]. It is the brain of application where all commands and instruction are carried out. In this tier programming code is written and stored.

Data tier

It is the portion of web application that stores all of the data and it also referred to as back end. Database tier is typically RDBMS that provides storage services. It does various operations like inserts, retrieves and manipulates data in database with the help of SQL queries.

To illustrate the working structure of web application: consider an online banking application. The credit union accounts clients to log in and view their accounts, make payment etc. Clients provide credentials by typing their username and password in to a web page (the presentation tier). The web page posts this information as name: value to create a query or request, to the database layer. This is always SQL (Structured Query Language) such as SELECT * FROM users WHERE username= ‘Madhav’ AND password= ‘bibash’. The data layer processes the request and return a record set back to the middle tier, the middle tier manipulates the data, creates a session and then passes a subset to the presentation tier. The presentation tier renders the information as HTML for the browser, displaying a menu of accounts options and personalized information about account balances and transaction history.

2.7 Communications

In the web application communication is established whenever user requests some information. In client-server architecture, when user enter his/her request then check the syntax after that generate database request in SQL. Then, the client transmits the message to the server, waits for a response, and formats the response for the end-user. The server accepts and processes the data requests, and then transmits the results back to the end-user.

2.7.1 Information

Information on the web is stored in documents and formatting languages, system, most commonly used is the HTML. Using HTML, documents are marked up, or tagged, to allow for publishing on the web in a platform independent manner. HTML documents are displayed in web browsers that understand and interpret HTML.

2.7.2 Content

HTML documents stored in files constitute static content i.e. the content of the document does not change unless the file itself is changed. However documents resulting from requests such as queries to database need to be generated by the web servers. These documents are dynamic content and as database are dynamic, changing as users creates, insert, update and delete data, the generation of dynamic web pages is a more appropriate approach than static content, particularly in web application [11].

2.7.3 Protocol

The exchange of information in web application is mainly governed by protocols such as HTTP or FTP, which define how clients, i.e. web browser and servers i.e. web servers, communicate. HTTP relies on request-response paradigm and a transaction consist of the following stages

Connection: The client establishes a connection with the web server.
Request: The client send a request message to the web server using HTTP.
Response: The web server send a response i.e. HTML document, back to the client.
Close: The connection is closed by the web server.

Chapter 3

Computer Security

Computer security is defined as the prevention and detection of unauthorized action performed by a user of computer system. In different papers, different researcher point out that the increased accessibility to corporations' systems through web application has also impact on the ever increasingly need for a computer security.

Security and protection are very important areas of computer science and IT industry. One way to describe this area is: "Security can be defined as set of methods and techniques which control data accessed by executing applications. Even wide definition includes a set of methods, techniques and legal standards which control data access by applications and humans, and protect the physical integrity of whole computer system, no matter if it is a described or not, or if it is centralized or decentralized" [12].

The security is one of the most important and wide area. Development of software and computer is increase rapidly: "To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computer, programming become a mild problem, and now we have gigantic computers, programming had become and equally gigantic problem", which create the several types of security violence [37].

In this section, we defined various components of computer network. Furthermore, we try to give a comprehensive image of what computer security is and which role it has in organizations.

3.1 Assets

The goals of security is to protect an organization's assets, meaning to prevent assets from being damaged, detect when such damage occurs, and then react in order recover the assets[19].

3.2 Security Services

Stallings [39] points out that "A service […] enhances the security of the data processing systems and information transfers of an organization". Gollmann mentions that three aspects are most frequently proposed to maintain the computer security, namely Confidentiality, Integrity and Availability (CIA). In a word, that is "the right information to the right person at the right time in the right context" [13]. Supplemented to CIA, Authenticity and Non- repudiation are also considered as important component of information security.

Confidentiality: Confidentiality is also called secrecy, which enables the protection of private information; it refers to preventing information from being disclosed to anyone who is unauthorized to access. For critical information such as medical and business data, confidentiality is a very important attribute. Using encryption techniques is a way to guarantee the confidentiality. In other word, Application confidentiality is the assurance that the information created and used by the application cannot be disclosed to unauthorized persons.

Integrity: Integrity is the prevention of unauthorized modification of information. Integrity also called accuracy is the trust of information, it consist of:

1. Data integrity, namely, that information has not been altered or corrupted before the recipient reads it.
2. Source integrity, namely, that information really comes from the supposed sender.

Availability: Information should be available when required. If one is authorized to get information, the Web application should be provide him with the required information efficiency, and the system should be able to recover quickly and completely in the case of a failure. For service information such as airline schedules and online stock system, availability is particularly important.

Authenticity: Authenticity means verifying a user's identity in a communication. Such process of verification is essential for web application like online banking and online shops. Methods for the authenticity can be classified in to the following cases:

1. User's knowledge such as passwords, PIN, etc.
2. Something the user such as security tokens, smart cards, etc.
3. Biometric identifier such as fingerprints, retinal patterns.

Non-Repudiation: Non-repudiation means that authentication cannot subsequently be refuted, that means, the sender of a message cannot later deny having sent this message and the recipient cannot deny having received it.

3.3 Threats

From a security perspective, computer systems have three general goals, with corresponding threats to them. First one, *data confidentiality*, is concerned with having secret data remain secret. More specifically, if the owner of some data has decided that these data are only to be made available to certain people and no others, the system should guarantee that release of the data to unauthorized people does not occur.

Goal	**Threat**
Data confidentiality	Exposure of data
Data integrity	Tempering with data
System availability	Denial of Service

Table2: Security goals and threats

The second goal, *data integrity*, means that unauthorized users should not be able to modify any data without the owner's permission. Data modification in this context

includes not only changing the data, but also removing data and adding false data as well. If a system cannot guarantee that data deposited in it remain unchanged until the owner decides to change them, it is not worth much as an information system.

The third goal, *System availability*, means that nobody can disturb the system to make it unusable. Such denial of service attacks is increasingly common [41].

Threats are any event that could adversely affect a system, and consequently an organization [11]. This affect can harm a system in various ways, and can be caused by person or an action, whether intentionally or unintentionally. In this research work, we Intend to concentrate our discussion about intentional threats, or security attack. These threats can be grouped by the security service that they threat:

Interruption: An attempt to destroy an asset or make. It unusable i.e. a threat to availability.

Interception: An attempt to gain access to an asset, i.e. a threat to confidentiality.

Modification: An attempt to tamper with an asset, i.e. a threat to integrity.

Fabrication: An attempt to creation objects in the system, i.e. a threat to authenticity.

Furthermore, Stallings suggests that attacks (which are source to threats) can be divided into two categories: passive attack in which the attack is trying to eavesdrop or monitor information and active attack which involves modification or fabrication of data.

3.4 Vulnerabilities

According to Dek [13] vulnerability is "A weakness that a person can exploit to accomplish something that is not authorized or intended as legitimate use of network or system. Invalidate input is the root of the issue. According to the OWASP Top Ten Project [32], invalidated input is the number one of the top ten most critical web application security vulnerabilities.

Anderson [3] and Stallings [39] refer to vulnerabilities as breeches in security mechanisms that can be used to perform attacks and thus constitute a threat to a computer system.

Examples of vulnerabilities are given below:
1. Lack of implemented security mechanisms, e.g. ignoring the virus threat by not installing anti-virus programs.
2. Deficient configuration of security mechanisms, e.g. configuring firewalls to allow any kind of traffic between networks.
3. Inadequate updating routines security mechanisms, e.g. not installing patches and new virus definitions for anti-virus programs.

Chapter 4

Database in Web Application

In this chapter, we simply present basic introduction about database which is used in web application and its related component because detailed study is not need for the purpose of our thesis work.

4.1 Introduction

Database is the shared collection of logically related data, which is build to obtain the required information for the particular organization. Database is designed to manage large bodies of information so those large bodies of information need to be manage properly and the work for manage information is done by the Database Management System (DBMS). DBMS is a collection of inter-related data and set of program to access those data. In other word, we can say that DBMS is software that handles all access to the database. DBMS are defined by Connolly et al. [11] as “A software system that enable

users to define, create and maintain the database and provide controlled access to this database."

There are number of database model like hierarchical model, network model, relational model etc. Among these different database models, relational model is mainly used. It uses a collection of tables to represent both data and the relational among those data.

4.2 SQL

SQL stands for structured query language. Its objective are to allow users to create database and relation structures, managing tables by inserting, modifying and deleting data as well as retrieve information from the database through queries. SQL does different tasks like: define the structure of data such as create schema, table, index, view, alter table, modify data in the database such as select ,update, delete, insert and it also does security and transaction control like commit, rollback, grant, create assertion[36]. The basic structure of SQL expression consists of three clauses: SELECT, FROM, WHERE.

A typical SQL query has the form:

SELECT A1, A2,…,An

FROM r_1, r2,……,r_m

WHERE P

Where A_i is attributes, r_i is relation and P is predicate

SQL language has several parts; Data Manipulation Language (DML) and Data Definition Language (DDL) are important two of them. Using DML user can manipulate data stored inside tables in the database, while the DDL allows creating and destroying database objects such as schemes, tables, views etc[36].

4.2.1 DML

The Data Manipulation Language has four available statements, namely SELECT, INSERT, UPDATE, and DELETE. Which supports the manipulation or processing of data objects. We describe each of these statements as below

Syntax	Represent
SELECT, INSERT……	Reserved words
Table_name, column_list	User defined words
|	Choice among alternatives
{}	Required element, for example{a}
[]	Optimal element, for example[a]
……	Optimal repetition(zero or more times)

Table3: SQL syntax

SELECT: It is the basic statement for retrieving information from table in the database. Basic form of the SQL SELECT statement is called mapping or SELECT-FORM-WHERE block.

SELECT <attribute list>

FROM <table list>

WHERE <condition>;

-<attribute list>is a list of attribute names whose values are to be retrieved by the query.

-<table list> is a list of the relation names required to process the query.

-<condition> is a condition (Boolean) expression that identifies the tuples to be retrieved by the query.

The sequence of the SELECT statement processing, and the meaning of the reserved word are [11]:

FROM: specifies which tables to choose.

WHERE: filters the selected data rows due to condition.

GROUP BY: groups together rows with same column value.

HAVING: filters the selected groups due to a condition.

SELECT: specifies which column should appear in the result.

ORDERED BY: specifies the order to sort output upon.

INSERT used to add one or more tuples to a relation. The result of SELECT query can be provided as data for INSERT. The syntax of the INSERT statement is given below.

INSERT INTO table_name [(column_list)]
VALUES (date_value_list)

Where table_name represents the name of the database table or view table, column_list represents the list of table to update and data_value_list represents the list of values to enter into each column in the new row.

UPDATE used for modifying data rows in a table. Update can apply to a single relation. Tuples can be modified using the UPDATE-SET-WHERE command. The syntax of the UPDATE statement is given below:

UPDATE table_name
SET column_name1= data-value1
[Column_name2= data-value2…]
[WHERE search-condition]

WHERE table-name represent the name of the database table_name, column_name represent the column name to modify and data-value represent the new value to enter into column. The new given value must correspond to the table column. The WHERE clause specifies which row is to be modified search-condition.

DELETE used for deleting data rows from a table. Tuples can be deleted from a table using the DELETE-FROM-WHERE command.
The syntax of the DELETE statement is given below:
DELETE FORM table_name
[WHERE search-condition]

Where table-name represent the name of database table and the WHERE clause specifies which row is to be delete, according to search-condition.

SELECT statements can be used to retrieve data in many different ways. For explain this, we give here a list of different queries formulations [11, 17, 29, 36].

Simple Queries: It can be used to retrieve either all or a selection of columns and rows from one or more tables.

Sorting Results: Output of a SELECT statement can be sorted based on one or more attributes by using ORDER BY clause.

Aggregate Functions: They are used to retrieve numeric information about the data. The clauses COUNT, SUM, AVG, MIN and MAX are used to retrieve numbers of rows, sum of values, values average, minimum value and maximum value respectively.

Grouping Results: For grouping results GROUP BY clause is used.
Scalar Sub Query: It is an inner query whose output is a single column and a single row. Sub query can help creating complex queries where in result from a secondary query can be used for instance as condition for primary query.

Multi-table queries: These are used to combine columns from different tables through usage of different JOIN clauses. JOIN may be natural join or outer join and outer join may be left outer join, right outer join or full join.

EXIST and NON EXIST CLAUSE: These clause can be used to check if a value exist or not in a table or in a result from a secondary query.

4.2.2 DDL

The SQL DDL provides commands for defining relation schemas, deleting relations and modifying relation schema [36]. In other word it allows creating and destroying database objects. The DDL does not allow user to manipulate data stored in database.
Basic SQL-DDL Commands are as follows [29, 36].
- For database schemas: CREATE SCHEMA, DROP SCHEMA

- For domains: CREATE DOMAIN, DROP DOMAIN
- For tables: CREATE TABLE, DROP TABLE, ALTER TABLE
-For views: CREATE VIEW, DROP VIEW

4.3 Query Techniques

An SQL query to be executed in a RDBMS can be constructed using two techniques. Either the query is allowed to be dynamically tailored with respect of both SQL keywords and query arguments, or the query syntax is unchangeable, only allowing arguments to be passed [11].

4.3.1 Dynamic SQL

Dynamic SQL refers to the concept of allowing an SQL query to be dynamically built by concatenating statements and using variables that supply the query with dynamic values. According to Connolly et al. [11] and Harper [23], the query is typically stored in variables and query builders consist of application a logic component that adds SQL syntax and arguments to the variable in a process governed by specified conditions. Such queries are interpreted and compiled at run-time by RDBMS, meaning that the query will be compiled every time it is executed. Since dynamic SQL allow SQL syntax to be added, both SQL keywords and values may be passed arguments to queries.

4.3.2 Static SQL

Static SQL refers to the concept of using fixed and unchangeable SQL queries. Such queries are predefined and compiled and are not permitted to add SQL keywords. Only arguments to clauses, e.g. WHERE, may be allowed to be passed to the queries. Either the query is embedded in application logic code in form of prepared statement or it resides in RDBMS as stored procedures.

Stored procedures are pre-compiled collections of SQL statements, or sub-routines that reside in the RDBMS. Either they are supplied by the database vendor, i.e. system stored procedures or additionally constructed by system developers, database administrator or

application programmers. They allow a developer to access and manipulate databases quickly and effectively. Since they are compiled in advance, they possess the property of being executed faster than dynamic SQL. Another property is that once created, stored procedures cannot be modified via dynamic SQL. Stored procedures are executed by invoking a command that includes the procedure identifier. This can be done either from a command prompt or from application programs written in languages such as C or Visual Basic. Several RDBMS supports this feature, but the set of stored procedures that follow with the installation and the syntax for invoking them varies [11, 17, 23].

4.4 Security

Database security refers to protection from malicious access. Among the form of malicious access are: unauthorized reading of data, unauthorized modification of data, unauthorized disruption of data. Database security is the "The protection of database against intentional or unintentional threats using computer-based or non-computer-based controls" [11]. Besides the effect that poor database security can have on the database, it may also threaten other parts of a system and thus an entire organization.
The risks related to database security are:

1. ***Theft and Fraud*:** Which are activities made intentionally by people. This risk may result in loss of confidentially or privacy.
2. ***Loss of confidentiality*:** This refers to loss of organizational secrets.
3. ***Loss of privacy*:** This refers to exposure of personal information.
4. ***Loss of integrity*:** This refers to invalid or corrupt data.
5. ***Loss of availability*:** This means that data or system cannot be reached.

Threats that correspond to those risks are such situations or events in which it is likely that an action. Threats can be tangible, that is cause loss of hardware or software or intangible as in with loss of credibility or confidence. In order to be able to face threats, a risk analysis should be conducted in which a group of people in an organization tries to identify and gather information about the organizations assets, the risks threats that may

harm the organization and the countermeasures that can be used to face those risks. After analyze such risk different security mechanisms are used in the system. These security measures can be computer-based controls or non-computer-based controls.

4.4.1 Computer-based Components

According to Connolly et al. [11], computer-based controls are used for protecting DBMS through means of authentications, views, backup and recovery, integrity, encryption and associated procedures.

Authorization: Authorization is use to define which privileges are granted to different users, which allows them to manipulate or retrieve information from different database objects. In order to ensure that the user is who he claims, authorization is used. Usually, a simple mechanism of username and passwords is used, whether in the DMBS or in combination with the operating system where the DBMS resides. A user is asked to fill his name and password, and the authentication mechanism confirms that the user is who he claims to be by comparing the password with the corresponding password in a list it maintains.

Views: Views are virtual tables that are created through some operations on database objects. A views or virtual table is a single table that is derived from other table called defining table. Update operation to view is limited; but querying is not.

*Backup and Recovery***:** Recovery mechanisms should regulate how backup and log can be used in case of failure.
In order to be able to recover from a failure, a DBMS must regularly make a copy of the database and log files. Log files are a list of activities made in the database that can be used to recover the database after failure. Checkpoint made in certain time intervals can assure that the backup and log files synchronized.

Integrity Constraints: Integrity constraints can be used to see that data in database does not get corrupt. In other word, integrity constraints ensure that change made to database by authorized user do not result in a loss of data consistency. Such constraints are called relational integrity constraints, and are rules that some database implement internally to maintain data validity. There are three main types of constraints [29], *Key constraints*, *Entity integrity constraints* and *Referential integrity constraints.*

Encryption: Data encryption is the process of protection for highly sensitive data. Encryption method is used for encoding the data so that other person cannot read it. There are vast numbers of techniques for encryption of data. As an example of weak encryption technique, consider that substitution of each character with next character in alphabet. Thus RAM becomes SBN so unauthorized person can not understands it.

Authorization and Authentication: In order for these mechanisms to work properly, a password policy should be maintained, which regulates matters like minimum password length, how often they should be replaced, as well as revoking old passwords.

Installation of New Software: Before any new software is to be installed, it should be properly tested so that it would not harm any data or mechanisms.

Installation/upgrading of System Software: Any system upgrades should be documented and reviewed. Before such upgrades take place, the risk of such an act should be considered and plans should be made for possible failures and changes.

4.4.2 Non-Computer-based-Control

The most important countermeasure among non-computer-based controls is the security policy and contingency plan. A security policy concerns security maintenance in an organization, and contain "...Set of rules that state which actions are permitted and which actions are prohibited" [19].

A contingency plan is a detailed description of the actions that should be taken in order to deal with unusual events, such as fire or flood.

Chapter 5

SQL Injection Attack

5.1 Introduction

SQL Injection is simply a term describing the act of passing SQL code into an application that was not intended by the developer. SQL Injection is one of the most well-known security vulnerabilities found in web application, which are caused by unchecked user input being passed to a back end database for execution. It causes unauthorized access to sensitive data, update, and deletion of data from the database. SQL Injection attacks pose a serious security threat to web application: It allow attacker to take control over underlying application and potentially sensitive information that database contain. Most of the web applications are attacked due to the reason of such kind of attack. A study by Gartner Group over 300 Internet web sites has showed that most of them could be vulnerable to SQL Injection Attack (SQLIAs) [21, 22].

In the view of some researcher SQL Injection is a technique used for manipulating server-side scripts that send SQL queries to RDBMS. The goal of attacker who uses SQL Injection is to manipulate the SQL query used by the script so that it could yield

unwanted results, fetching, inserting, manipulating or deleting protected rows or tables in the database [22]. Problems of SQL Injection are not the fault of database server but due to the poor input validation and coding. The main cause of SQL Injection vulnerabilities is due to insufficient validation of user input [1, 2].

Measurements of damaging data depend upon the how much access permission provides for account used to access database. There are a big number of applications which are vulnerable to this kind of attack. We need to know that, this kind of attack is not limited to one particular application; it deploys its effect in various applications like PHP, ASP, JAVA etc.

SQL queries generated by concatenation of the static part of the query and values intended from fields is the base of this attack. For example, if there is a field for entering the username (tbUser) and a field for entering password (tbPassword) and we perform authentication in the following manner[1,2].

```
String Query="SELECT COUNT (*) FROM [User] WHERE Username='"+ tbUser.Text
+"' AND Password='"+ tbPassword.Text +" ' ";
SqlCommand Command=new sqlCommand (Query, Connection);
If ((int) Command.ExecuteScalar ()>0)
                    …….
                    …….
```

It is easy to unauthorized user to enter disguised SQL query in the username field, like

'OR 1=1--

This would turn the database query into

SELECT COUNT (*) FROM *[User] WHERE Username=' 'or 1=1--''AND Password=' '

Here--represent the comment in SQL which is supported by many relational servers, including Microsoft SQL Server, Oracle and MySQL this query is equivalent to

SELECT COUNT (*) FORM [User] WHERE Username= ' 'or1=1

Since 1=1 always evaluates to true, this query will always return more than 0 rows, if there are records in table. So that malicious user can easily authenticate with invalid credentials.

SQL injection basic

We know the basic SQL syntax:

SELECT<database name> FROM<table> WHERE<condition>

Consider the table name dbtable

User_id	Username	Userpass
16	RamLal	12345

Here the query,

SELECT user_id FROM dbtable WHERE username='RamLal' AND userpass= '12345'

Would return the value 16

In the above SQL,

SELECT user_id FROM dbtable WHERE username= 'RamLal' AND userpass= '12345'

Injection point

5.1.1 Basic

According to spett [38], a web application from a hackers perspective, be viewed as consisting of the following layers: desktop layer, transport layer, access layer, network layer and application layer. At the desktop layer, computer with web browser acting as clients are used for accessing a system. The transport layer represents web and access layer constitutes the entrance point into a corporation's internal system from the web. The network layer consists of the corporation's internal network infrastructure and finally, the application layer includes web servers, application server have its own implemented countermeasures attack as figure.

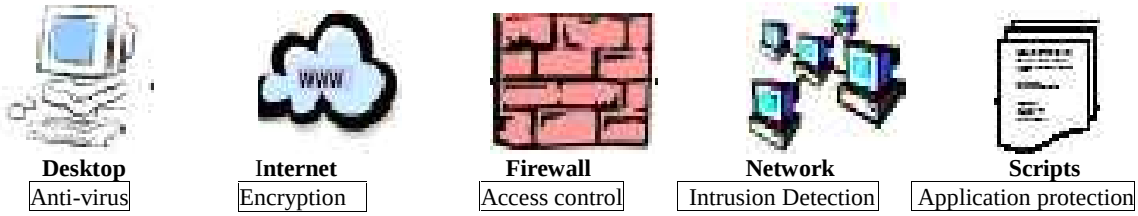


Figure: 6.Security Layer in web applications

Unfortunately, SQL injection attacks can only prevented in application logic components such as scripts and programs in countermeasure like encryption, firewalls, intrusion detection and database security are also important. They are effective when dealing when other type of attacks for example encryption only protects stored data or data during transport in and between lower layers. Firewalls, IDS, encryption, SSL are not sufficient for the application protection. Firewall control traffic but not the application and similarly SSL and data encryption protect data transmission but not the application. Thus applications need protection through separate and specific security measure. In the context of web application user input may be encrypted between the client-side and server-side. Furthermore SQL query may be encrypted during transport between components such as scripts and programs on the server-side but in order for the server-side to construct SQL queries are decrypted. Decrypt is the process of converting the encrypted data into the original form of data. Only a user who knows how to decrypt the encrypted data can read them. So in terms of data security encryption, acts as a great role in hiding the sensitive data in own code from the unwanted users, who may not know how to break the code.

5.2 Prerequisites

There are different types of prerequisites which are inter-related and need to occur SQL-Injection attack in web application. These prerequisites are related to query execution properties and feature that database support.
Some of the prerequisites are as follows

End-of Line Comments: The ability to comment out parts of a SQL statements, meaning that the RDBMS will not take notice of the SQL syntax followed by a comment symbol. For example, some RDBMS user '--' as a comment symbol.

Sub-Selects: Sub-Selects are multiple SELECT statements used together. A top-level statements to retrieve values to be used in a WHERE clause.

Multiple Statements: It refers to the ability to allow execution of multiple SQL statements, where each statement is separated by a delimiter, e.g. semicolon [1, 18, 25].

Error Message: Errors that occurs in the RDBMS or in any server-side script or program can produce an error message that can be sent to the client machine and printed in the web browser [1, 18, 25, 26].

Weak Data Types: Several script and programming languages used in web application development support variables of weak type, i.e. variables that can store data of arbitrary type [16, 17].

Data Type Conversion: Several RDBMS support variable type conversion e.g. allowing numeric values to be converted automatically into a string type.

5.3 Vulnerabilities

The Vulnerabilities in the computer security is discussed in the chapter 3. But here, we present vulnerabilities that might be inherent in web application and that can be exploited by SQL injection attack.

Invalidated Input: Invalidated input is the main causes of the vulnerabilities, unchecked parameter to SQL queries that are dynamically built can be used in SQL injection attack. Those parameter may contain SQL keyword, e.g. INSERT or SQL control character such as quotation marks and semicolons [1, 4].

Error Message Feedback: Error messages that are generated by the server-side programs may be returned to the client-side and printed in to the web browser. While these messages can be useful during development for debugging purpose, they can also constitute risks to the applications. Attacker can analyze these messages to obtain information about database in order to construct their attack [1, 9, 18, 25].

Uncontrolled Variable Size: Variable that allows storage of data that is larger than expected may allow attackers to enter modified or fabricated SQL statements. Scripts that do not control variable length may even open for other attackers.

Variable Morphism: If a variable can contain any data, it is possible for an attacker to store other data than expected. Such variable are either of weak type e.g. variable in PHP are automatically converted from one type to another by the RDBMS e.g. numeric values converted into a string type. For example, SQL keywords can be stored in a variable that should contain numeric values.

Sub-Select: if the RDBMS supports sub-select, the variables of attack methods used by an SQL injections attacker increase inserted in WHERE clauses of the original SELECT clauses [23, 27].

Multiple Statements: If the RDBMS supports JOIN and UNION the variations of attack methods used by an SQL injection attacks increases. For example an additional INSERT statement could be added after a SELECT statement causing two different queries to be executed. If this performed in a login form, the attacker may add himself to the table of the users [16, 18, 25].

Dynamic SQL: Dynamic SQL is one of the most important factors of the SQL Injection attack. Dynamic SQL refers to the SQL queries dynamically built by script or program into a query string. Typically, one or more scripts or programs contribute and successfully build the query using user input such as names and password as values in e.g. WHERE clauses. The problem with this approach is that query building component can also receive SQL keyword and control characters, creating a completely different query than the intended.

5.4 Techniques of SQL Injection Attack

There are numerous ways to conduct SQL injection attack, and the chosen method depends upon the attacker will accomplish, i.e. which security services to endanger, and what vulnerabilities the web application contains. These techniques constitute the threats and they can be grouped into two main categories:

1. Access through Login Page
    1.1. Using ‘or’ condition.
    1.2. Using ‘having’ clause.
    1.3. Using multiple queries.
    1.4. Using extended stored procedures
2. Access through URL

5.4.1 Access through login Page

The easiest SQL Injection is to bypass the login form where the user is authenticated against a password supplied by the user.

A sample login form and authorization script is given below

Login form:

Authorization script in the web page:

```
<%@page language=vb%>
```

```
<%@import namespace=System.Data.OleDb%>
<form runat=server>
<table>
 <tr>      <td>UserName</td>
           <td><asp:TextBox id=txtUN runat=server/></td>
 </tr>
 <tr>      <td>Password</td>
           <td><asp:TextBox id=txtPW TextMode=Password runat=server/></td>
 </tr>
</table>
<br>

<pre>              <asp:Button id=btnSubmit Text=Submit onClick=onSubmit runat=server/>
<asp:Button id=btnReset Text=Reset onClick=onReset runat=server/>
<asp:Label id=lblmsg runat=server/>
 </pre>
</form>
<script runat=server>
sub onSubmit(o as object, e as eventArgs)
dim con as new oledbConnection("Provider=SQLOLEDB;Data Source=DESKTOP\MADHAV;
Initial Catalog=mydb; User Id=sa; Password=123456")
dim cmd as new oledbCommand
dim query as string
query="select  * from dbtable where userName='" & txtUN.text &"' and userPass='" & txtPW.text
& "'"

con.open
cmd.Connection=con
cmd.CommandText=query
dim rdr as OLEDBDataReader
rdr=cmd.executeReader

if not rdr.read then lblmsg.text="Bad Credential" else lblmsg.text=" Logged In sql "
con.close
end sub
sub onReset(o as object, e as eventArgs)
if IsPostBack() then
txtUN.text=""
txtPW.text=""
lblmsg.text="Bad Credential"
end if
end sub

</script>
```

5.4.1.1 Using ‘or’ condition

To bypass this authorization, the user will have to enter the following code in the above form.aspx

User Name=Madhav
Password= ‘or 1=1--

Then we get the output as “Logged In sql”

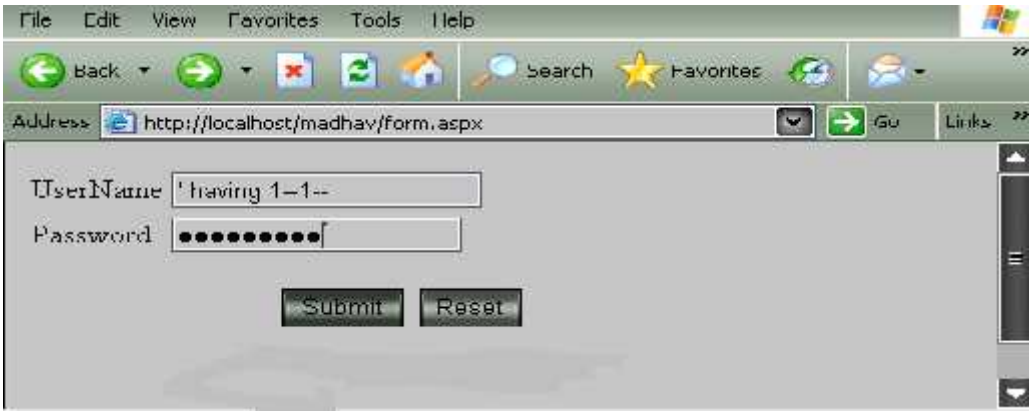
Then query would now look like:

select * from dbtable where userName='Madhav' and userPass='' or 1=1 - -‘

The query now checks for an empty password, or the conditional equation of 1=1, then a valid row has been found in the dbtable. Here ‘(single quote) is used to terminate the string and ‘--‘ is used to comments the remaining part of the query.

5.4.1.2 Using ‘having’ clause

When we enter the having clause in the form.aspx as below

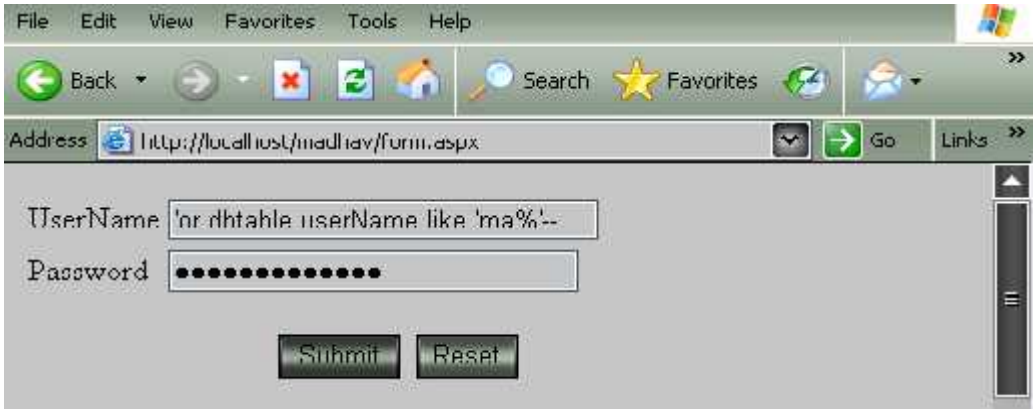


Here, UserName: ‘having 1=1--
Password: [anything]
Then we get the Error message as below in the screen of web browser.

System.Data.OleDb.OleDbException: Column 'dbtable.userPass' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.
Column 'dbtable.userName' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

Having clause can be used to know the table name and column name of the first column in the query. This error message now tells the attacker the name of one field from the database dbtable.userName using the name of this field, attacker can now use SQL server's 'LIKE' keywords to login with the following identity

User Name: ' or dbtable.userName like 'ma%'- -
Password: [anything]
Then attacker can easily login into the database



The resultant query would now look like this

select * from dbtable where userName= ' 'or dbtable.userName like 'ma%'' and userPass="

This query checks for username starting from 'ma' in the database table.

5.4.1.3 Using multiple queries

SQL server delimits queries with a semi-colon. The use of a semi-colon allows multiple queries to be submitted as one batch and executed sequentially.

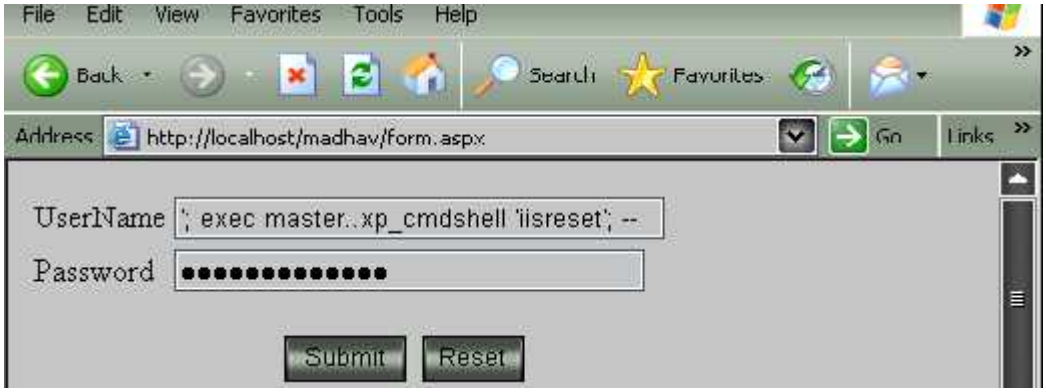
Here, UserName: ‘ or 1=1; drop table dbtable --
Password: [anything]
Then query execute in two parts.

In the first part, Select the Username field for all rows in the dbtable and in the second part delete the dbtable, after then when user logged in following error will appear,

```
System.Data.OleDb.OleDbException: Invalid object name 'dbtable'.
```

5.4.1.4 Using extended stored procedures

Executing an extended procedure using our form.aspx with an injected command as the username, seems as follow



Here,
UserName: ‘; execmaster..xp_cmdshell ‘iisreset’;--
Password: [anything]

This would send the following query to SQL server

```
select *
from dbtable
Where username= ‘ ‘; execmaster..xp_cmdshell ‘iisreset’; -- and userpass=’’
```

To execute stored procedure user or database should have necessary privileges. If IIS installed on the same machine as SQL Server, then user could restart it by using the ‘xp_cmdshell’ extended stored and ‘iisreset’.

5.4.2 Access through URL

There are different techniques through which attacker can inject the malicious SQL code in database through URL, one of them is

By manipulating the query string in URL.

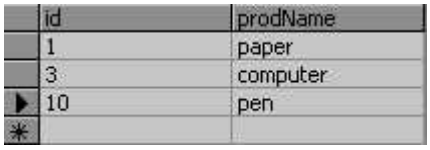
Many times URL looks like this:

www.mysite.com/products.aspx?p_id=1

Where 1 is the ID of the product and many sites would simply build a query around the p_id query sting, like this

select prodName from products where p_id=1

Assume that we take following table (products) and rows setup on our SQL Server.



id	prodName
1	paper
3	computer
10	pen

Suppose that we have created the following sqlproducts.aspx script

```
<%@page language=vb%>
<%@import namespace=System.Data.OleDb%>
<script runat=server>
sub page_Load(o as object, e as eventargs)
dim con as new oledbConnection("Provider=SQLOLEDB;Data Source=DESKTOP\MADHAV;
Initial Catalog=mydb; User Id=sa; Password=123456")
dim cmd as new oledbCommand
dim query as string,prodId as string

prodId=Request.QueryString("p_id")
query="select  prodName from products where id=" & prodId

con.open
cmd.Connection=con
cmd.CommandText=query

dim rdr as OLEDBDataReader
rdr=cmd.executeReader

if rdr.read then
response.write("Got product " & rdr.item("prodName"))
else
response.write ("No Product Found" )
end if

con.close
end sub
```

```
</script>
```

So if we visited sqlproducts.aspx in the browser with the following URL

http://loalhost/madhav/sqlproducts.aspx?p_id=1

Then, we see the following line of text in our browser

Got product paper

To know the field name of products table attacker can write

http://localhost/madhav/sqlproduct.aspx?p_id=1%20having%20 1=1

Each %20 in the URL represents a URL-encoded space character. Using a bit of know-how and some URL-encoding; we can just as easily pull the name of the products field from the products table.

This would produce the following error in browser

```
System.Data.OleDb.OleDbException: Column 'products.prodName' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.
```

Now using products field call up the following URL. In the browser

http://localhost/madhav/sqlproducts.aspx?p_id=0;insert into products(prodName) values (left(@@version,50)),then the output is *Not product found*

However the above query run an INSERT query on the products table, adding the first 50 character of SQL server's@@version variable as a new record in the products table. which contains the details of SQL Server's version, build, etc.

An attacker could get the version of SQL server by writing:

http://localhost/sqlproducts.aspx?productId= (select%20max(id) %20from%20products)

After getting the version details of SQL server an attacker could exploit the vulnerabilities associated with this version, if the SQL server is not fully patched.

5.5 Preventing SQL Injection Attack

When we designed our applications with care, SQL injection attacks can be avoided most of the time. SQL injection occurs due to poor coding and poor website administration. There are number of things that developers can do to reduce site's susceptibility to attack.

Limited User Access: It means that, the account used by the system administrator, should never be used for the web application access. Instead we have to create different accounts for different client profile. In case that the RDBMS chosen has a default system administrator password, change it immediately after installation. The default system account (sa) for SQL server 2000 should never be used because of its unrestricted nature. We should always setup specific accounts for specific purpose.

Limit the length of User Input:

1. Keep all text boxes and form fields as short as possible. By keeping text boxes and form short, we are taking away the number of characters that can be used to formulate an SQL injection attack. For example, regular expression {\d_a-zA-Z}{4, 12} is used in the code of application for password and login fields with 4-12 length.

2. Always try to post our forms with method attribute set to POST so clued-up users don't get any ideas.

Modify Error Reports: To avoid SQL injection, developer handle or configure the error reports in such a way that error cannot be show to outside users. In these error reports some time full query is shown, pointing to the syntax error involved and attacker could use it for further attacks. Display of errors should be restricted to internal users only.

Limited Privileges: As possible as least authority is given to the database accounts of the web applications. The actions made by the web application user on stored procedure should be limited too. By doing so it minimizes the damage an attacker can cause in case of authorization by pass. Normally SELECT is a privilege that in almost any case will be given to an account. Which is used for losing in to the system and retrieve information. But at that time user should not have INSERT or DELETE privileges.

*Static SQL***:** To prevent the database of web application from malicious attack to use the static SQL instead of dynamic SQL. Static SQL are those SQL statement that cannot be altered by inserting SQL keywords values are expected.

Input Validation: All data send by the client-side should be considered potentially harmful. While input validation could be implemented in client-side scripts for performance factors, one should never rely on client-side scripts for security. Therefore, every parameter send from the client-side should always be examined and validated by server side script and programs. For example all single quote (') should be replace by the two single quotes for this in VBscript '*replace*' function is used like

```
Function escape (input)
Input=replace (input, "'", " ''")
Escape=input
End function
```

Also we need to remove the *culprit character* such as --, select, insert xp_ etc. by using the *rem_culp_char* function.

In the prevented application when we insert the data with malicious code with single quote, having clause, comment line etc then we simply sees the window with message *password not valid* because in the database there is a username Madhav and password is also madhav, thus other person who does not know these username and password can not able to login. For example, when we enter the username as Madhav in the username field and password as ‘or 1=1- - in the password field then instead of display the query as
select count(*) from dbtable where userName='Madhav' and userPass='' or 1=1 --‘
It change this query like

select count(*) from dbtable where userName='Madhav' and userPass=''

It stops the occurrence of attacks, because WHERE clauses now requires both the userName and Password fields to be valid. Similarly, when we enter the having clause in the username field and anything or empty in the password field instead of displaying the valuable error message, it display nothing so attacker could not found the table name then s/he does not proceed forward. This code also does not accept the culprit character. Thus attacker does not able to attack the web application without valid username and password.

5.6 Query Validation

We know that select query used in login page of web application is of the form:
“select * from users where UserName=“‘&txtUN.Text&”’ and UserPass= “ ‘&txtPW.Text&””“.
Suppose there is SQL-Injection caused by submitting malicious code ‘ or 1=1--in the password Textbox, then our select query become as follows
“select * from users where UserName= “‘&txtUN.Text&”’and UserPass= ‘‘ or 1=1--
It means that username can be anything and password should be empty or 1=1 and rest character. This is always true because it is a tautology thus anybody can login using malicious code.

To prevent this SQL-Injection attack, our select query should always be one which is used in our program. It means no other additional characters should be added beside UserName and Password in our select query. For this we develop a system i.e. procedure that checks pattern or format of select query before executing the query. If pattern matches to our desire query pattern then we assume that there is no attack in our SQL otherwise there will be SQL-Injection attack. This idea is illustrated in the following flow chart.

Figure7: Flow chart for validating query

For simple, login with username and userpassword, the pattern for select query can be as:
select * from table where col1= ‘UserName’ and col2= ‘UserPassword’
There can be many whitespace between select, *, from, table, where, col1 and col2 in select query but we develop a pattern matching procedure using transition table that simulate DFA for a more generalized select query i.e.
select *|col1,col2…..from table where col1= ‘string’|col1= number and|or col2= ‘string’|col2=number and|or …….

The DFA for this query is as follows:

Here states q_{12} and q_{14} are final states. If we parse the select query and either state q_{12} or q_{14} was reached then we conclude that there is no SQL-Injection attack otherwise there will be attack. Each state in DFA takes either a single character or a sequence of characters as input. If a state takes a sequence of characters as input then that state can be transited to next state only if a character "$" appears at the end otherwise state remain in the same state. For example in case of keyword "select":

$tt(q_0,s)= q_0$

$tt(q_0,e)= q_0$

$tt(q_0,l)= q_0$

$tt(q_0,e)= q_0$

$tt(q_0,c)= q_0$

$tt(q_0,t)= q_0$

$tt(q_0,\$)= q_1$

It shows that it is transited from q_0 to q_1 only when there is "$" at the end of keyword "select" so, in our select query used in login web page we have to insert $ at the end of each keywords. This can be done as:

query =selectquery. to lower

query = query. Replace("select", "select$")

query = query. Replace("from", "from$")

query = query. Replace("where", "where$")

query = query. Replace("and", "and$")

query= query. Replace("or", "or$")

query = query. Replace(" ,", " ,")[According to DFA whitespace is need before ","]

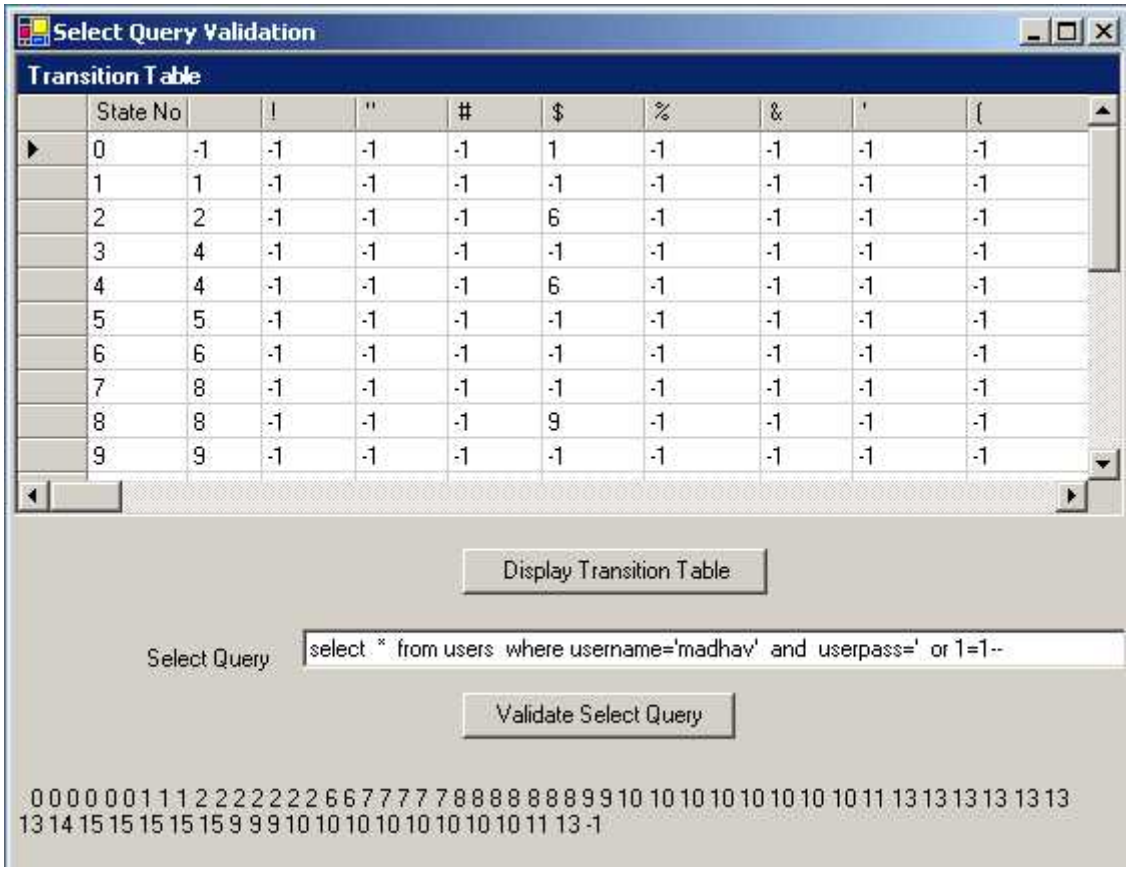
We assume that transition table as 2- dimensional matrix tt(20,127), initially all element initialized -1 in the transition table tt, (*See transition table in appendix*)

$tt(2,32)=q_2$, means when state q_2 takes input as ASCII value of 32 i.e. whitespace then q_2 remains in same state. In this way $tt(6,65)=q_7$ means, when state q_6 takes input as ASCII value of 65 i.e. character "A", state q_6 is transited to state q_7 and tt(0,32)= -1 means, there is no transited of state from q_0 when input is a character whose ASCII value is 32 i.e. whitespace.

Now if we parse the injected query

“select * from users where UserName= “ ‘&txtUN.Text&”’ and UserPass= ‘ ‘or 1=1--

Finally we reach the state at -1(full transition for this query is shown in label of below application), this is invalid state our select query is not one as assumed then we found that there is SQL-Injection attack. So, if there is SQL-Injection attack then we can stop it from inserting malicious code in the database of web application. When we insert any character in the Textbox of following application, then its transition value is display in the Label. Thus, by see the transition values we can find out whether valid code is inserted or not in our selected query.



State No		!	"	#	$	%	&	'	(
0	-1	-1	-1	-1	1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1
2	2	-1	-1	-1	6	-1	-1	-1	-1
3	4	-1	-1	-1	-1	-1	-1	-1	-1
4	4	-1	-1	-1	6	-1	-1	-1	-1
5	5	-1	-1	-1	-1	-1	-1	-1	-1
6	6	-1	-1	-1	-1	-1	-1	-1	-1
7	8	-1	-1	-1	-1	-1	-1	-1	-1
8	8	-1	-1	-1	9	-1	-1	-1	-1
9	9	-1	-1	-1	-1	-1	-1	-1	-1

5.7 Execution Overhead

To determine the execution time overhead, we compare the executing time taken by SQL queries without injection checking with the SQL queries after injection checking. We tested this result in a machine with window xp os with 128 MB of main memory and 1.80 GHz processor. In this study we determine the time to parse the query, execute the database call and return the results. Here, we take the response time for 5 and 10 users in 10 times and calculate their average time by taking mean for each of the user. Instead of allowing different machines for different users, all of the user processes were simulated on a single machine.

This study shows that guarded SQL statement take more time to execute than SQL queries before guarded. Table 4: shows the response times for load tests with 5 and 10 users and this graph shows that response times increase as the load is applied. In other word, we can say that execution time overhead increase as the load increase.

By using the test results, we plotted the following graphs to show the execution time overhead in normal statement and guarded statement for five and ten users respectively.

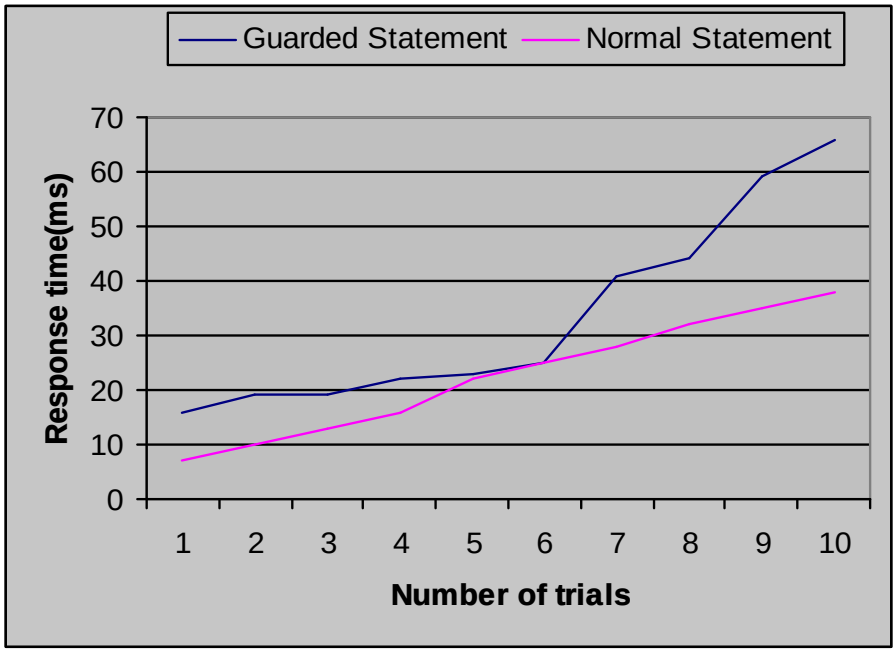


Figure9: Execution time for 5 users

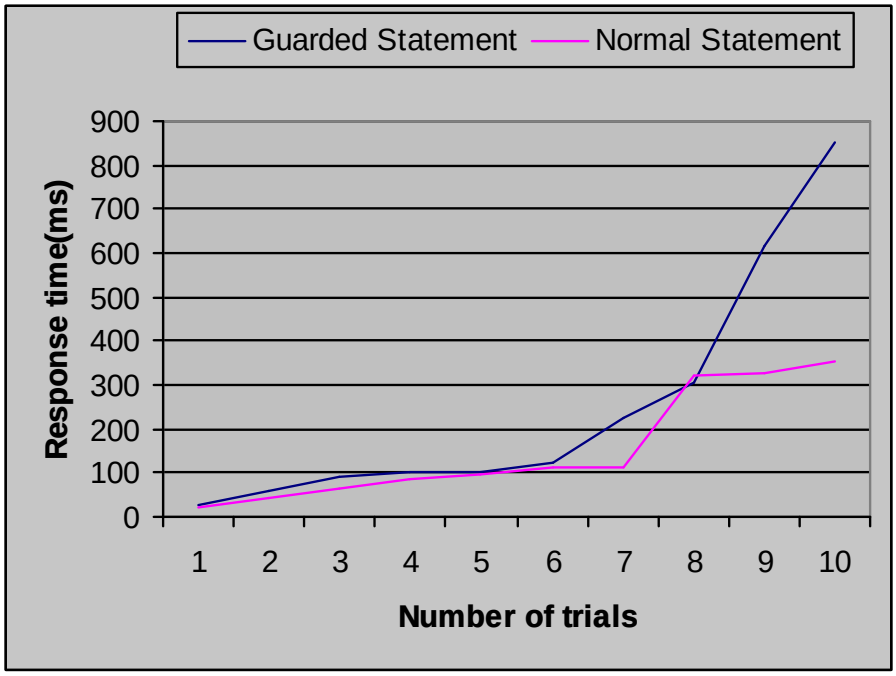


Figure 10: Execution time for 10 users

These two graphs easily shows that response time (in micro second) for guarded statement is greater than that time for the normal statement.

In summary, average response time for guarded and normal statement is grouped as follows:

Number of users	Query Method	Avg. Response time(ms)
5	Normal Statements	23
5	Guarded Statements	34
10	Normal Statements	154
10	Guarded Statements	251

Table 4: Response time testing for normal statements and guarded statements

Chapter 6

Conclusion and Future work

6.1 Concluding remarks

SQL Injection has frequently been discussed in everywhere like various security Communications, Organizations, Conferences, hacker site and software manufactures. SQL Injection requires neither specialized tools nor extensive experience knowledge. A web browser is sufficient in order to perform SQL Injection attack against web application as long as the attacker has basic knowledge of HTTP, relational database and SQL.

In this thesis work, we show the different technique of attacking the web application through SQL Injection attack and some of their prevention technique. Here we also show the execution time overhead for normal statements and guarded statements, which show that more time is, need to execute the query in guarded statement than for the normal statement and response time increase while load is increase. We can also say that if we follow the prevention technique given in this work including our own purpose prevention technique i.e. prevention through transition table validation method our application is secure in some aspect. During the time of this work when we studied various research paper we conclude that the main reason of SQL Injection attack occur due to the lack of adequate validating the user input. Here we collect the different techniques which cause to occurs the SQL Injection and their prevention technique and test them in ASP.NET.

Through the study of different research paper and from our thesis work we conclude that security of web application is depends upon the programmer who designs the application and how much prevention measure is applied by him. At last we said that during the design of application we need to give least privilege to user as possible and every users input should be validate.

6.2 Future work

In this thesis work, we concentrated our attention on specific attacking technique on database of web application i.e. SQL Injection attack. We strongly believe that only this work is not sufficient for secure the web application then in this area further investigation is need. In this work we only concerned with some technique of how SQL Injection attack occur and preventing technique about it and their execution time. During preparing this work we studied that there are number of other issues in the SQL Injection that are harmful threats to web application but these are not included in this small thesis work. Then further research work is concentrated on the different issues which are related to the security of web application to increase the lever of security awareness among people and organization.

References

[1] Anley, Chris (2002): *"Advanced SQL Injection In SQL Server Applications"* Next Generation Security Software Ltd. White paper http://www.nextgenss.com/papers/advanced_sql_injection.pdf.

[2] Anley, Chris (June, 2002): **"**(*more) advanced sql injection in sql server application."* Technical report, NGSSoftware Insight Security Research (NISR), http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf.

[3] Anderson, Ross J (2001): **"***Security Engineering***".** John Wiley & Sons, Inc., Pages 250-275

[4] Andrews, Chip (April, 2003): *"Sql injection faq".* http://www.sqlsecurity.com/DesktopDefault.aspx?

[5] Aucsmith, D (Sep, 2004): *"Creating and maintaining software that resists malicious attack"* http://www.gtisc.gatech.edu/aucsmith.bio.htm Distinguished Lecture Series Atlanta, GA

[6] Bass, Len., Clements, Paul and Rick Kazman, Rick (1999): *"Software Architecture in Practice"*. Addison-Wesley , Pages 190-195

[7] Boyd, S.W and Keromyris, A.D (June, 2004): *"SQLand: "preventing SQL injection attack."* In processing of the 2nd Applied Cryptography and Network Security (ACNS) Conference

[8] Buehrer, G.T., Weide, B.W and Sivilotti, P.A.G (2005): *"Using Parse Tree Validation to Prevent SQL Injection Attacks"***.** 5th International Workshop on Software Engineering and Middleware (SEM), Pages 106-113

[9] Cerrudo, Cesar: *"Manipulating microsoft sql server usingsql injection***".** Technical report, Application Security, Inc.

[10] Chartier, Rober (2001): *"Application architecture: An n-tier approach"* – part 1 Online Documentation, http://www.15seconds.com/issue/ 011023. htm.

[11] Connolly, Thomas.,Begg, Carolyn and Strachan, Ann (1999):*"Database Systems*

- *A Practical Approach to Design, Implementation, and Management"*.Addison – Wesley

[12] D, simic (2005): Materials form lectures for post graduate studies of *"Security techniques in computer network"*. Faculty of organizational science, Belgrade

[13] Dekker, Marcel (1997): *"Security of the Internet"*, Low Price Edition, page 201

[14] Dijkstra, E (1972): *"The Humble programmer"***,** ACM Turing lecture, 3rd International workshop on software engineering for securing systems, Volume III

[15] Dornseif, M (May, 2005): *"Common Failures in Internet Applications"* http://md.hudora.de/presentations/

[16] Eizner, Marthin (2001): *"Direct sql command injection."* Technical report http://qb0x.net/papers/MalformedSQL/sqlinjection.html.

[17] Farrow, Rik (May, 2002): *"Databases under fire"***.** Web advisory, http://www.airscanner.com/pubs/sql.pdf.

[18] Finnigan, Pete (Nov, 2002): *"Sql injection and oracle, part one"*. Technical report, Security Focus. http://www.securityfocus.com/infocus/1644.

[19] Gollmann, Dieter (2001): *"Computer Security"*, John Wiley & Sons, Pages 35-40

[20] Halfond, W.G. and Orso (2005): *"AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attack"*, 20th IEEE/ACM International Conference an Automated Software Engineering, Long Beach, CA, USA

[21] Halfond, W.G. and Orso (2005):*"Combining static Analysis and Runtime monitoring to counter SQL-Injection Attack"*, in proceedings of the third international ICSE workshop on Dynamic Analysis (WODA)

[22] Halfond, W.G., Viegas, J., and Orso, A.A (2006): *"Classification of SQL-Injection Attack and Countermeasures"*. Technical report Volume I, Pages 40-43

[23] Harper, Mitchell (2001): *"SQL Injection Attacks - Are You Safe?"* http://www.sitepoint.com/article/sql-injection-attacks-safe

[24] Hung, Y-W., Hang, C., Tsai, C-H., Lee, D and Yu, F (May, 2004): *"Securing Web Application Code by Static Analysis and Runtime Protection"*, in processing of the 12th international World Wide Web conference (WWW), Pages 40-52

[25] Jaquith, Andrew (Feb., 2002.): *"The security of applications: Not all are created Equal"*. Technical report, @stake, http://www.atstake.com/research/reports/acrobat/atstake_app_unequal.pdf
.

[26] Litchfield, D: *"Web application disassembly with ODBC error message"*. In http://www.nextgeness.com/papers/webappdis.doc

[27] McDonald, S (April, 2004): *"SQL Injection: Modes of attack, defense and why it is matters"*, In Wall Street Journal, pages 5-15 http://www.Governmentsecurity.org/article/SQLInjectionModesofAttackDefenseandWhyItMatters.php

[28] McMillan, R (2004): *"Web security flaw settlement: FTC charges that Pecto web site left customer data exposed"*. In http://pcworld.com/news/article/0,aid,118638,00.asp

[29] Nevathe, S (2002): "Fundamental of Database systems", Pearson education 3rd edition, Pages 67-69

[30] Newman, Aaron C (2001): "*Protecting oracle database*". Technical report, Application Security, Inc, http://www.appsecinc.com/presentations/protecting_Oracle_Databases_White_Paper.pdf

[31] NIST (Jan, 2006): *"Vulnerability statistics generation engine."* http://nvd.nist.gov/statistics.cfm.

[32] Open Web Application Security Project (2004): *"The ten most critical Web application security vulnerabilities."* http://umn.dl.sourceforge.net/sourceforge/owasp/OWASPTopTen2004.pdf

[33] Open Web Application Security Project (2005): *"A guide to building secure Web Applications."* http://easynews.dl.sourceforge.net/sourceforge/owasp/OWASPGuide2.0.1.pdf

[34] Robertson, William (Feb, 2006): "*Generalization and Characterization Techniques for the Anomaly-based Detection of Web Attacks*". In 20th Annual ACM Conference on, Software Engineering Languages, Pages 20-22

[35] Scott, D .Sharp, R (2002): *"Abstracting Application-level Security"*, in processing of the 11th International Conference on the World Wide Web (WWW), Pages 396-407

[36] Silberschtz,A; Koth,H.F and Sudarshan,s: "*Database system concept*", McGraw-Hill, 4th edition, Pages 11-12

[37] Sima, Caleb (2004): *"Security at the Next level"*. http://www.spidynamics.com/whitepapers/webappwhitepaper.pdf

[38] Spett, Kevin (2002): *"SQL injection-are your web applications vulnerable?"* Technical report, SPI Dynamics http://www.spidynamics.com/whitepapers/webappwhitepaper.pdf

[39] Stallings, William (1999): *"Network Security Essentials: Applications and Standards."* Prentice-Hall, Inc, 3rd edition

[40] Su, Zhendong and Wassermann, Gary (Jan, 2006): *"The Essence of Command Injection Attacks in Web Applications"*, Pages 372-382

[41] Tanenbaum, Andrew.S (2004): "*Modern Operating Systems*", Prentice-Hall Inc, 2nd Edition, Pages 584-585