# DETECTING AND PREVENTING SQL INJECTION ATTACKS IN DATABASE USING WEBSERVICE

Lords Institute Of Engineering and Technology

*MD AKRAMUL HAQUE, ** MD AFROZ KHAN ***G.KUMAR
*B .Tech Dept Of CSE Lords Institute Of Engineering and Technology
***Associate Prof Dept Of CSE Lords Institute of Engineering and Technology

## ABSTRACT

SQL injection attacks are major threats for user application security. The task is to identify and prevent them. It takes advantage of poor input validations in code, weak firewall and website administration. By passing malicious SQL statements through web application,which is executed on backend. This project can be implemented on all the existing DBMS's for increasing security by avoiding SQL injection attacks

## Objective of the project:

The main goal of this project is to identify SQL injection attacks in web application and prevent the database from such attacks by which the web application will be more secure and reliable.

## Introduction

SQL injection attacks have been described as one of the most serious threats for Web applications. Web applications that are vulnerable to SQL injection may allow an attacker to gain complete access to their underlying databases. Because these databases often contain sensitive user information, the resulting security violations can include identity theft, loss of confidential information, and fraud. In some cases, attackers can even use an SQL injection attack to take control of and

## Existing System

There is no method to handle SQL injection attacks on database. We propose a technique that prevents the web applications and its database from any type of SQL injection attacks in an efficient manner.

## Purpose System

This proposed technique consists of two filtration models to prevent SQLIA'S. 1) Active Guard filtration model 2) Service Detector filtration model. The steps are summarized and then describe them in more detail in following sections. This Technique is used to detect and prevent SQLIA's with runtime monitoring.

## Introduction

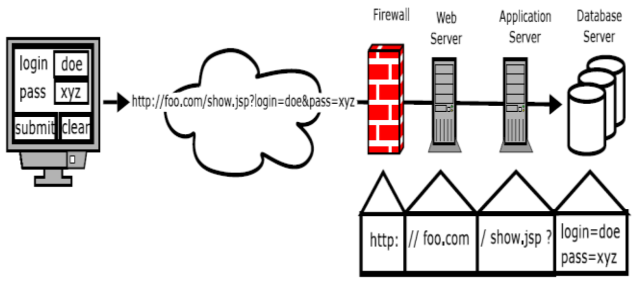
SQL injection refers to a class of code-injection attacks in which data provided by the user is included in an SQL query in such a way that part of the user's input is treated as SQL code. By leveraging these vulnerabilities, an attacker can submit SQL commands directly to the database. Web application that receives input from users and incorporates it into SQL queries to an underlying database. Most Web applications Used on the Internet or within enterprise systems work this way and could therefore be vulnerable to SQL injection.

## Categories of Sql Injuction Attacks

There are four main categories of SQL Injection attacks against Oracle databases

1. SQL Manipulation
2. Code Injection
3. Function Call Injection
4. Buffer Overflows

## Example of SQL Injection Attack:



## Runtime Monitoring

During execution, when the application reaches a hotspot, the runtime monitor is invoked and the string that is about to be submitted as a query is passed as a parameter

### Example of parsed runtime queries

### study of the system

The purpose of this document is to present a detailed description of the approach for protecting web applications against SQL Injection Attacks

1. Administrator
2. User
3. Attacker

### 1. Administrator

Administrator is responsible for the adding new items to the shopping cart. Administrator can view the details of the users and their transactions.

### 2. User:

New user can register into the site by providing his/her details. User can login to the site to view the products available in cart and do shopping.

### 3. Attacker:

Attacker will login to site as an existing user by using some attacking methodologies and do transactions as a original user or pass some malicious SQL statements in place of user name to gain access the database

## Requirement Specification

**Software Configuration:**

| | |
|---|---|
| Language | :ASP .net |
| Tool | :Visual Studio 2008 |
| Operating system | :Windows XP |
| Database | :SQL Server 2005 |

**Hardware Configuration:**

| | |
|---|---|
| Processor | :Pentium3 |
| RAM | :1 GB RAM |
| Hard disk | :80GB |

- Keyboard :Standard 104keys
- Mouse :Standard mouse

## Portability

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented.Microsoft's commercial implementations of the framework cover Windows, Windows CE, and the Xbox 360.

## Architecture

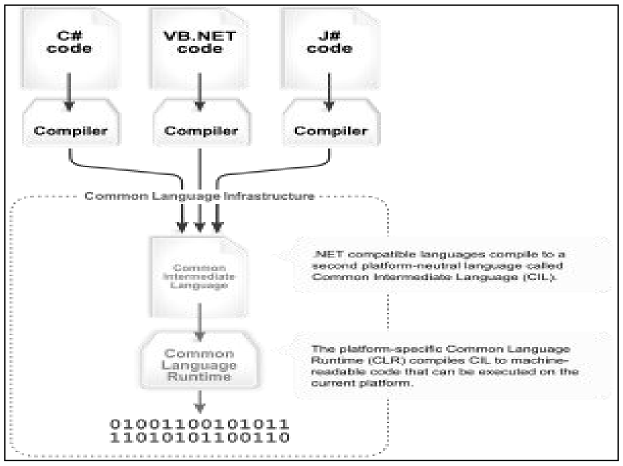

C# code
VB.NET code
J# code
Compiler
Compiler
Compiler
Common Language Infrastructure
Common Intermediate Language
.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).
Common Language Runtime
The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.
01001100101011
11010101100110

**Fig 4.3.1: Visual overview of the Common Language Infrastructure (CLI)**

### 4.6.1 ER-Diagram:

In software engineering, an Entity-Relationship model is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion, Diagrams created using this process are called entity-relationship diagram, or ER diagram or ERDs of short.

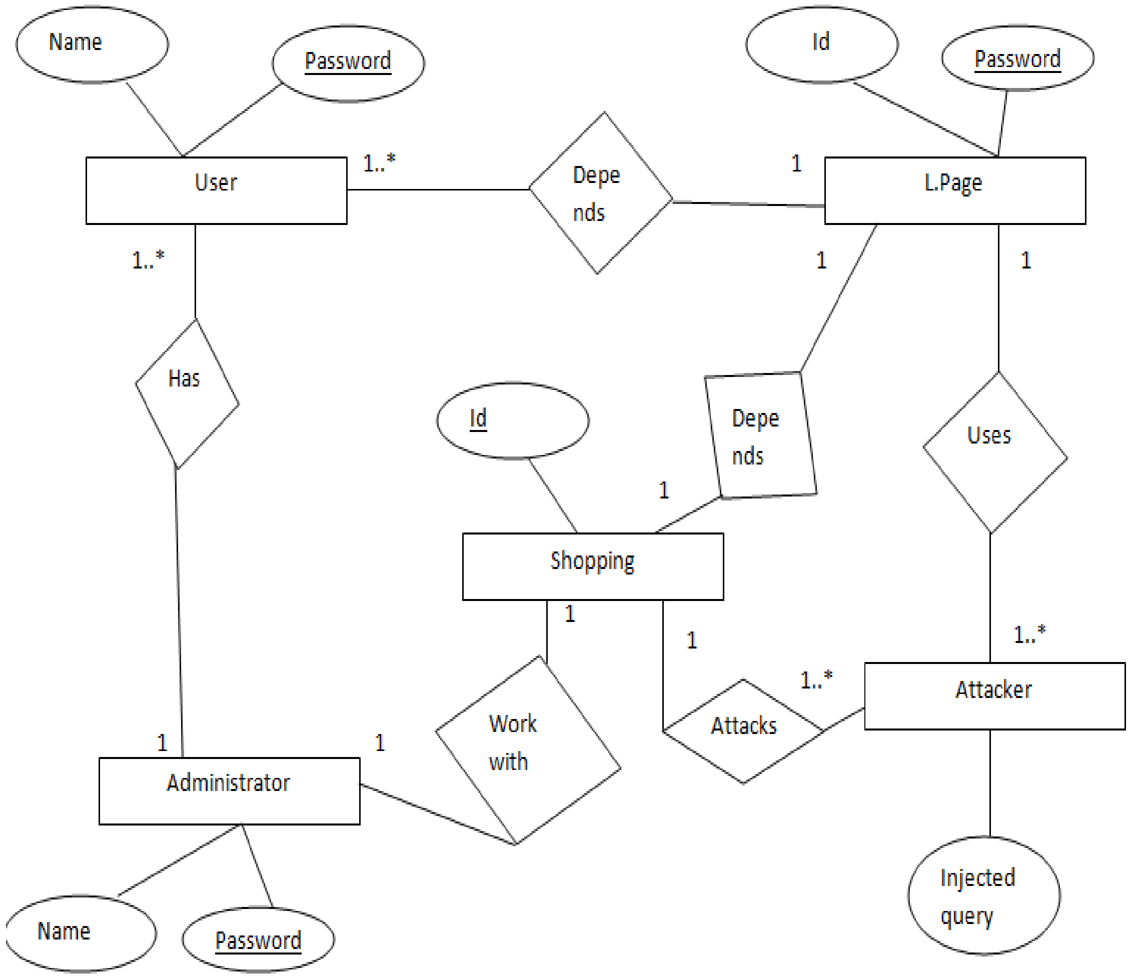
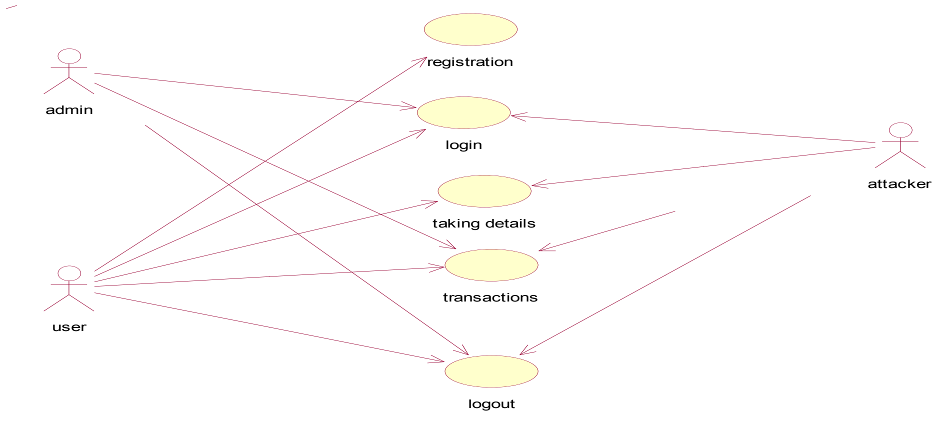


Fig 4.1: ER Diagram of the System

**Use case diagram on overview of system**



4.4.2.UseCasediagramforSQLinjection attacks

## 4.4.3 Sequence Diagrams:

The Sequence Diagrams emphasizes the Time ordering of messages between objects. It models collaboration of objects based on a time sequence. Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self-explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects.

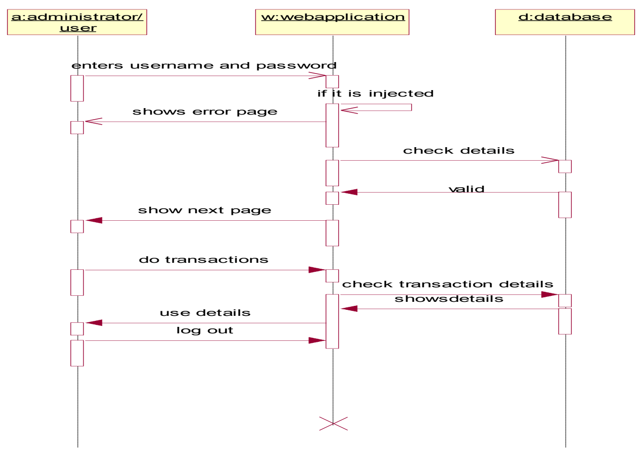
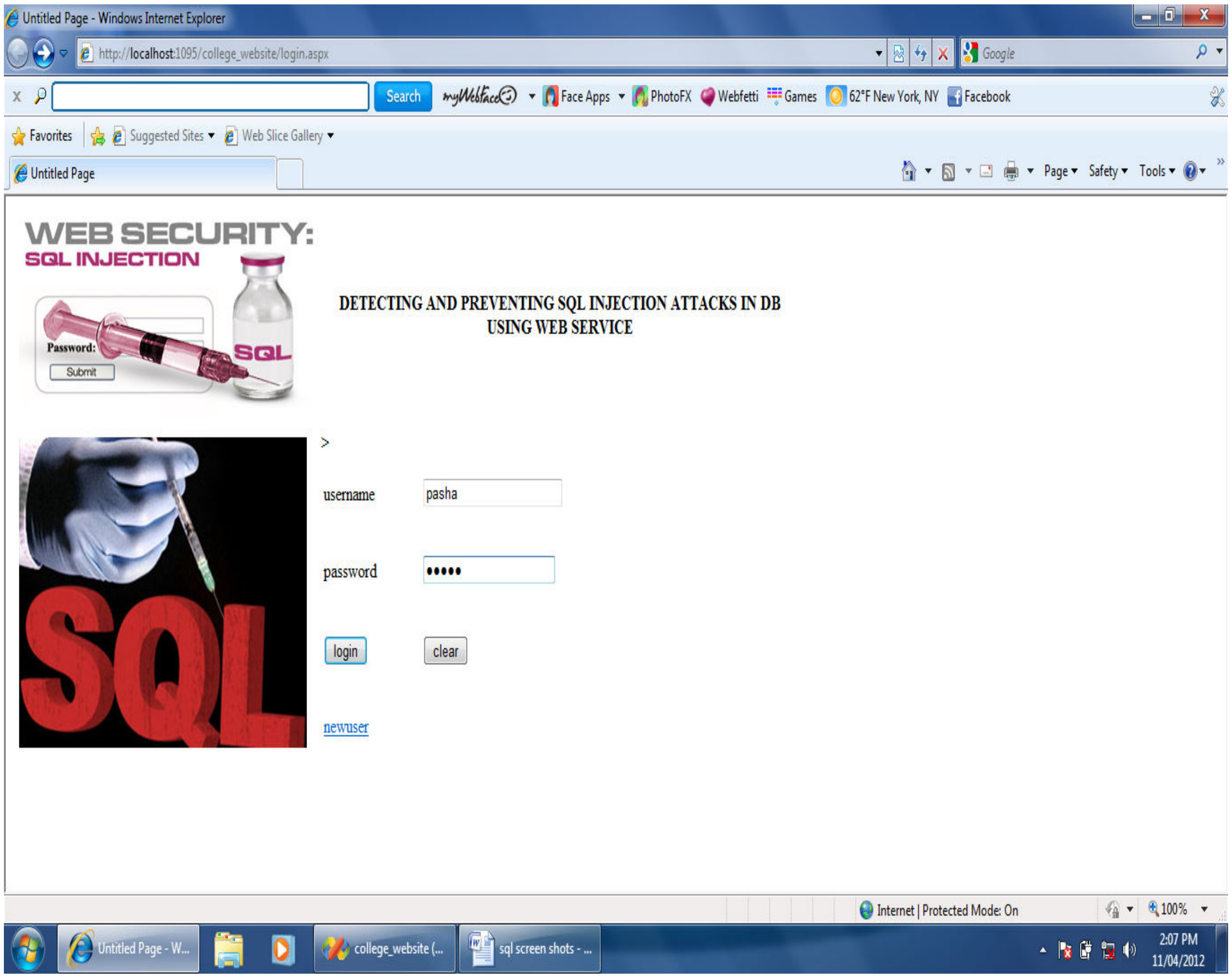


Fig 4.4.3. Sequence Diagram for SQL injection attacks


Untitled Page - Windows Internet Explorer
http://localhost:1095/college_website/login.aspx
WEB SECURITY:
SQL INJECTION
SQL

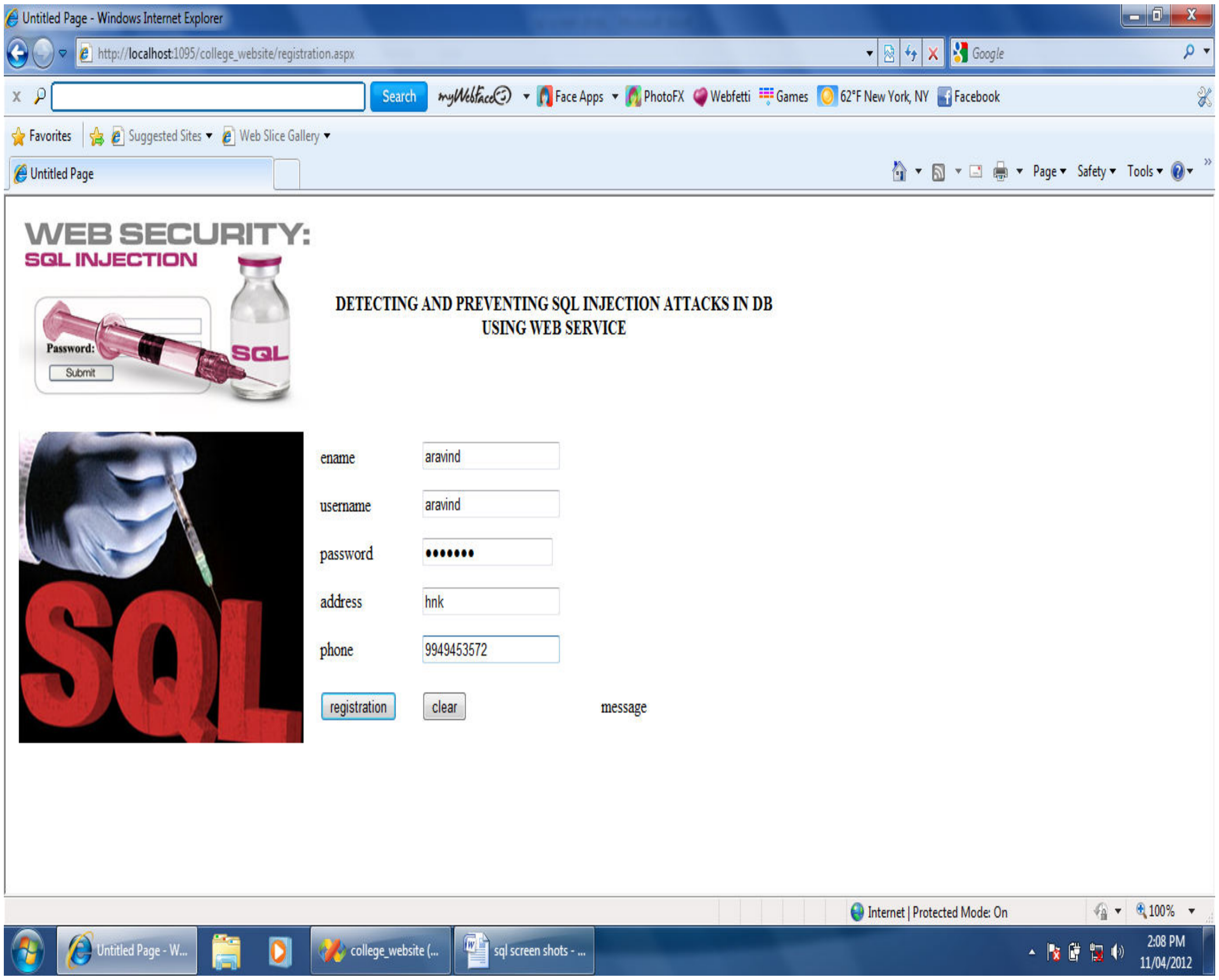
DETECTING AND PREVENTING SQL INJECTION ATTACKS IN DB
USING WEB SERVICE

username pasha

password •••••

login clear

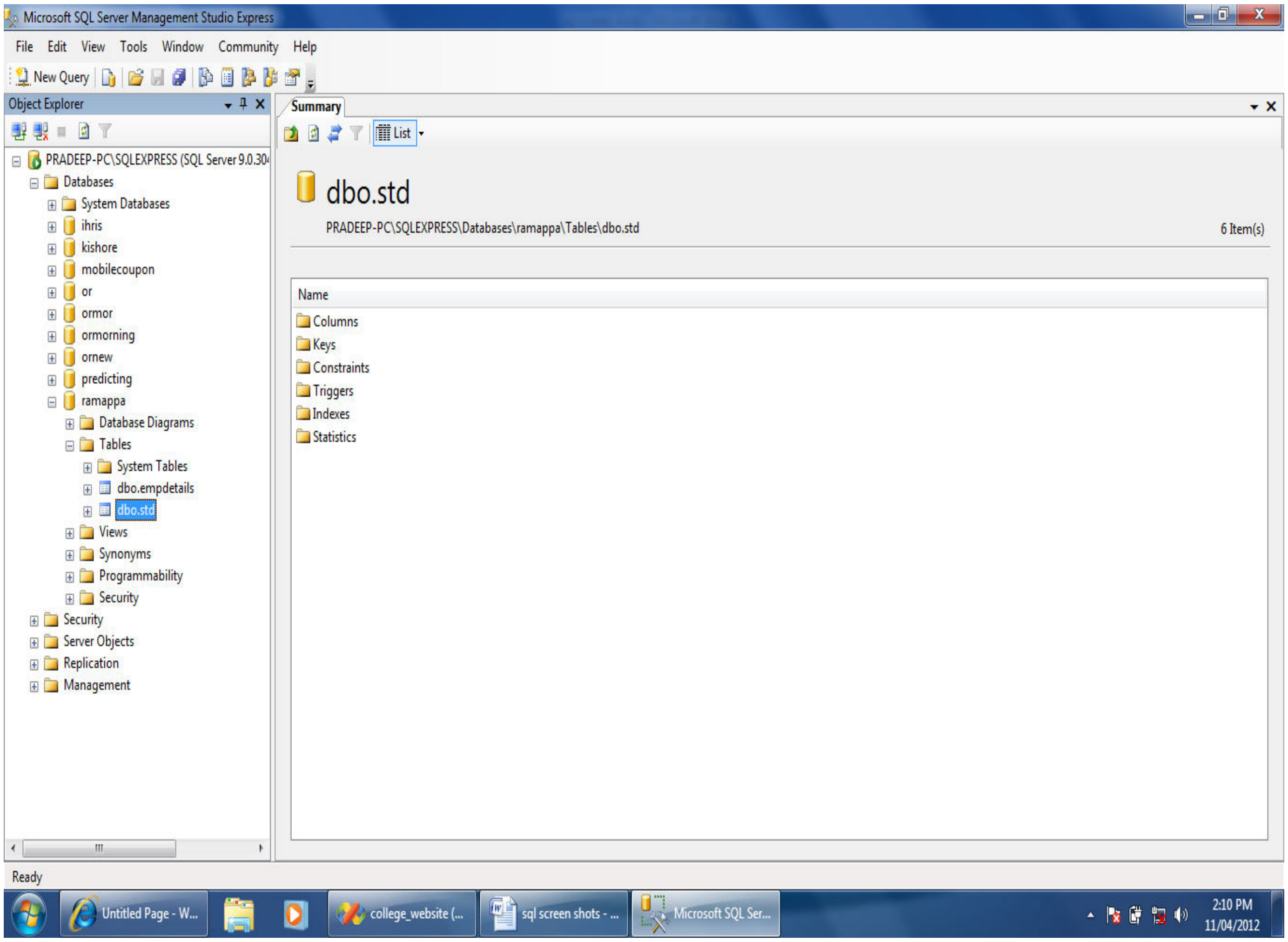
newuser


Untitled Page - Windows Internet Explorer
http://localhost:1095/college_website/registration.aspx
WEB SECURITY:
SQL INJECTION
SQL

DETECTING AND PREVENTING SQL INJECTION ATTACKS IN DB USING WEB SERVICE

| | |
|---|---|
| ename | aravind |
| username | aravind |
| password | ••••••• |
| address | hnk |
| phone | 9949453572 |

registration clear message


Microsoft SQL Server Management Studio Express
File Edit View Tools Window Community Help
New Query
Object Explorer
Summary
List

PRADEEP-PC\SQLEXPRESS (SQL Server 9.0.30
Databases
System Databases
ihris
kishore
mobilecoupon
or
ormor
ormorning
ornew
predicting
ramappa
Database Diagrams
Tables
System Tables
dbo.empdetails
dbo.std
Views
Synonyms
Programmability
Security
Security
Server Objects
Replication
Management

dbo.std
PRADEEP-PC\SQLEXPRESS\Databases\ramappa\Tables\dbo.std 6 Item(s)

Name
Columns
Keys
Constraints
Triggers
Indexes
Statistics

Ready

## CONCLUSION AND FUTURE SCOPE:

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in ASP.NET and C#.Net web based application and on some extent Windows Application and SQL Server, but also about all handling procedures related with **“Detecting and preventing SQL injection attacks in database using web service”.** In this project, Web Service Oriented XPATH Authentication Technique checks the user input with valid database and does not affect the database directly then the validated user input field is allowed to access the web application. This proposed technique was able to suitably classify the attacks that performed on the applications without blocking legitimate accesses to the database. These results show that our technique represents a promising approach to countering SQLIA’s.