# Palestine Polytechnic University


**College of Engineering & Technology**
**Computer & Electrical Engineering Department**


**Graduate Software Project**

Web Application Security Tester

**Project Team**


**Khalaf Idais** — **Tayseer Sweiti**
**Abd-Alrahman Abu-Sakour** — **Osama Jabari**

**Project Supervisor**
**Mr. Ibrahim Tuffaha**


**Hebron - Palestine**

**July 2005**

# Abstract

Web applications become more important in most aspects of our life. It serves a mission critical system that used to process millions of dollars in dailly online transaction. Web security vulnerabilities impact the risk of web site. Identifying web security vulnerabilities is crucial for web developers to develop a secure web application

We develop a windows application to identify vulnerabilities in web applications; also we introduce this document as a guide for building secure web application. Our software gives samples of web vulnerabilities, because identifying all of the web vulnerabilities is not practical, since new web vulnerabilities continually arise.

## TABLE OF CONTENTS

# List of Tables

# List of Figures

# List of Figures (cont.)

## List of Abbreviations

| Abbreviation | Description |
|---|---|
| ADO | ActiveX Data Objects |
| ASP | Active Server Page |
| CGI | Common Gateway Interface |
| CLR | Common Language Runtime |
| DBMS | Data Base Management System |
| DHTML | Dynamic Hyper Text Markup Language |
| DLL | Dynamic Link Library |
| DoS | Denial of service |
| ER | Entity Relation |
| HTML | Hyper Text Markup Language |
| HTTP | Hyper Text Transfer Protocol |
| HTTPS | Secure Hyper Text Transfer Protocol |
| IP | Internet Protocol |
| LAN | Local Area Network |
| OLTP | On-Line Transition Processing |
| OS | Operating System |
| SQL | Structured Query Language |
| SSI | Internet Information Services |
| SSL | Secure Socket Layer |
| TCP | Transmission Control Protocol |
| TSL | Transmitting Secure Socket Layer |
| URL | Universal Resource Locator |
| VB | Visual Basic |
| WAN | Wide Area Network |
| WWW | World Wide Web |
| XML | Extensible Markup Language |
| XSS | Cross Site Scripting |

## Problem Initiation

Web Application is a client/server software application that interacts with users or other systems using HTTP or other application layer protocols. A Web Service is a collection of functions that are packaged as single entity and published to the network for use by other programs Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide. Web application security is a theory and practice of information security relating to the World Wide Web, HTTP and web application software.

Designing a web application is an exercise in designing a system that meets a business need and not an exercise in building a system that is just secure for the sake of it. However, the application design and development stage is the ideal time to determine security needs and build assurance into the application. Prevention is better than cure, after all.

Testing the level of the security of a web application helps developers to build a reliable and secure web application. We will try to build a windows application which tests the web application for common vulnerabilities

# 1

# Chapter One

# Introduction

# Chapter One

# 1 Introduction

## 1.1-Preface

We all use web applications everyday whether we consciously know it or not. That is, all of us who browse the web. That is all of us, right? The ubiquity of web applications is not always apparent to the everyday web user. When you need to transfer money, search for a flight, check out arrival times or even the latest sports scores during work, you probably do it using a web application. Web Applications and Web Services (web applications that describe what they do to other web applications) are the major force behind the next generation Internet. The last two years have seen a significant surge in the amount of web application specific vulnerabilities that are disclosed to the public. To this day, not one web application technology has shown itself invulnerable to the inevitable discovery of vulnerabilities that affect its owners' and users' security and privacy.

For many organizations, web sites serve as mission critical systems that must operate smoothly to process millions of dollars in daily online transactions. Web security vulnerabilities continually impact the risk of a web site. When any web security vulnerability is identified, performing the attack requires using at least one of several application attack techniques.

## 1.2-Classes of website attacks

A well known website attacks are grouped into six basic classes:

### 1.2.1-Authentication

These attacks target a web site's method of validating the identity of a user, service or application. This class of attack includes:

**1- Brute Force[9]**

A Brute Force attack is an automated process of trial and error used to guess a person's username, password, credit-card number or cryptographic key. Many systems will allow the use of weak passwords or cryptographic keys, and users will often choose easy to guess passwords, possibly found in a dictionary. Given this scenario, an attacker would cycle though the dictionary word by word, generating thousands or potentially millions of incorrect guesses search for the valid password. When a guessed password allows to access to the system, the brute force attack has been successful and the attacker is able to access the account. The same trial and error technique is also applicable to guessing encryption keys. When a web site uses a weak or small key size, it's possible for an attacker to guess a correct key by testing all possible keys.

Essentially there are two types of brute force attacks, (normal) brute force and reverse brute force. A normal brute force attack uses a single username against many passwords.

While brute force techniques are highly popular and often successful, they can take hours, weeks or years to complete.

**2- Insufficient Authentication [4][9]**

Insufficient Authentication occurs when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate. Web-based administration tools are a good example of web sites providing access to sensitive functionality. Depending on the specific online resource, these web applications should not be directly accessible without the user required to properly verify their identity.

### 3- Weak Password Recovery Validation [5][9]

Weak Password Recovery Validation is when a web site permits an attacker to illegally obtain, change or recover another user's password. Conventional web site authentication methods require users to select and remember a password or pass phrase. The user should be the only person that knows the password and it must be remembered precisely.

Examples of automated password recovery processes include requiring the user to answer a "secret question" defined as part of the user registration process. This question can either be selected from a list of canned questions or supplied by the user. Another mechanism in use is having the user provide a "hint" during registration that will help the user remember his password. Other mechanisms require the user to provide several pieces of personal data such as their social security number, home address, zip code etc. to validate their identity. After the user has proven who they are, the recovery system will display or e-mail them a new password.

A web site is considered to have Weak Password Recovery Validation when an attacker is able to foil the recovery mechanism being used. This happens when the information required to validate a user's identity for recovery is either easily guessed or can be circumvented. Password recovery systems may be compromised through the use of brute force attacks.

## 1.2.2-Authorization[6][9]

These attacks target a web site's method of determining if a user, service, or application has the necessary permissions to perform a requested action. For example, many web sites should only allow certain users to access specific content or functionality. Other times a user's access to other resources might be restricted. Using various techniques, an attacker can fool a web site into increasing their privileges to protected areas. This class of attack includes:

**1- Credential/Session Prediction**[7][9]

Credential/Session Prediction is a method of hijacking or impersonating a web site user. Deducing or guessing the unique value that identifies a particular session or user accomplishes the attack. Also known as Session Hijacking, the consequences could allow attackers the ability to issue web site requests with the compromised user's privileges. Many web sites are designed to authenticate and track a user when communication is first established. To do this, users must prove their identity to the web site, typically by supplying a username/password (credentials) combination. Rather than passing these confidential credentials back and forth with each transaction, web sites will generate a unique "session ID" to identify the user session as authenticated. Subsequent communication between the user and the web site is tagged with the session ID as "proof" of the authenticated session. If an attacker is able predict or guess the session ID of another user, fraudulent activity is possible.

Example:

Many web sites attempt to generate session IDs using proprietary algorithms. These custom methodologies might generate session IDs by simply incrementing static numbers. Or there could be more complex procedures such as factoring in time and other computer specific variables. The session ID is then stored in a cookie, hidden form-field, or URL. If an attacker can determine the algorithm used to generate the session ID, an attack can be mounted as follows:

1) Attacker connects to the web application acquiring the current session ID.
2) Attacker calculates or Brute Forces the next session ID.
3) Attacker switches the current value in the cookie/hidden form field/ URL and assumes the identity of the next user.

**2- Insufficient Authorization** [7][9]

Insufficient Authorization is when a web site permits access to sensitive content or functionality that should require increased access control restrictions. When a user is authenticated to a web site, it does not necessarily mean that he should have full access to all content and that functionality should be granted arbitrarily.

Authorization procedures are performed after authentication, enforcing what a user, service or application is permitted to do. Thoughtful restrictions should govern particular web site activity according to policy. Sensitive portions of a web site may need to be restricted to everyone expect to perhaps an administrator.

**3- Insufficient Session Expiration** [5][9]

Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization. Insufficient Session Expiration increases a web site's exposure to attacks that steal or impersonate other users. Since HTTP is a stateless protocol, web sites commonly use session IDs to uniquely identify a user from request to request. Consequently, each session ID's confidentiality must be maintained in order to prevent multiple users from accessing the same account. A stolen session ID can be used to view another user's account or perform a fraudulent transaction. The lack of proper session expiration may improve the likely success of certain attacks. For example, an attacker may intercept a session ID, possibly via a network sniffer or Cross-site Scripting attack. Although short session expiration times do not help if a stolen token is immediately used, they will protect against ongoing replaying of the session ID. In another scenario, a user might access a web site from a shared computer (such as at a library, Internet cafe, or open work environment). Insufficient Session Expiration could allow an attacker to use the browser's back button to access web pages previously accessed by the victim.

A long expiration time increases an attacker's chance of successfully guessing a valid session ID. The long length of time increases the number of concurrent and open sessions, which enlarges the pool of numbers an attacker, might guess.

**4- Session Fixation** [6]

Session Fixation is an attack technique that forces a user's session ID to an explicit value. Depending on the functionality of the target web site, a number of techniques can be utilized to "fix" the session ID value. These techniques range from Cross-site Scripting exploits to peppering the web site with previously made HTTP requests. After a user's session ID has been fixed, the attacker will wait for them to login. Once the user does so, the attacker uses the predefined session ID value to assume their online identity. Without active protection against session fixation, the attack can be mounted against any web site using sessions to identify authenticated users. Web sites using sessions IDs are normally cookie-based, but URLs and hidden form-fields are used as well. Unfortunately, cookie-based sessions are the easiest to attack. Most of the currently identified attack methods are aimed toward the fixation of cookies.

### 1.2.3-Client-side Attacks [9]

The Client-side Attacks focus on the abuse or exploitation of a web site's users. When a user visits a web site, trust is established between the two parties both technologically and psychologically. A user expects web sites they visit to deliver valid content. A user also expects the web site not to attack them during their stay. By leveraging these trust relationship expectations, an attacker may employ several techniques to exploit the user. This class of attack includes:

**1- Content Spoofing** [5][9]

Content Spoofing is an attack technique used to trick a user into believing that certain content appearing on a web site is legitimate and not from an external source. This attack exploits the trust relationship established between the user and the web site. The technique has been used to create fake web pages including login forms, defacements, false press releases, etc.

**2- Cross-site Scripting [8]**

Cross-site Scripting (XSS) is an attack technique that forces a web site to echo attacker-supplied executable code, which loads in a user's browser. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

When an attacker gets a user's browser to execute his code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross site Scripting attacks essentially compromise the trust relationship between a user and the web site.

**1.2.4-Command Execution [9]**

The Command Execution attacks are designed to execute remote commands on the web site. All web sites utilize user-supplied input to fulfill requests. Often these user-supplied data are used to create construct commands resulting in dynamic web page content. If this process is done insecurely, an attacker could alter command execution.
This class of attack includes:

**1- Buffer Overflow [4][9]**

Buffer Overflow exploits are attacks that alter the flow of an application by overwriting parts of memory. Buffer Overflow is a common software flaw that results in an error condition. This error condition occurs when data written to memory exceed the allocated size of the buffer. As the buffer is overflowed, adjacent memory addresses are overwritten causing the software to fault or crash.

A Buffer Overflow can be used as a Denial of Service attack when memory is corrupted, resulting in software failure. Even more critical is the ability of a Buffer Overflow attack to alter application flow and force unintended actions. This scenario can occur in several ways. Buffer Overflow vulnerabilities have been used to overwrite stack pointers and redirect the program to execute malicious instructions. Buffer Overflows have also been used to change program variables.

**2- Format String Attack** [2][9]

Format String Attacks alter the flow of an application by using string formatting library features to access other memory space. Vulnerabilities occur when user-supplied data are used directly as formatting string input for certain C/C++ functions (e.g. fprintf, printf, sprintf, setproctitle, syslog, ...).

If an attacker passes a format string consisting of printf conversion characters (e.g. "%f", "%p", "%n", etc.) as parameter value to the web application, they may:

- . Execute arbitrary code on the server
- . Read values off the stack
- . Cause segmentation faults / software crashes

**3- OS Commanding** [2][9]

OS Commanding is an attack technique used to exploit web sites by executing Operating System commands through manipulation of application input. When a web application does not properly sanitize user-supplied input before using it within application code, it may be possible to trick the application into executing Operating System commands. The executed commands will run with the same permissions of the component that executed the command (e.g. Database server, Web application server, Web server, etc.).

**4- SQL Injection**[3][9]

SQL Injection is an attack technique used to exploit web sites that construct SQL statements from user-supplied input. Structured Query Language (SQL) is a specialized programming language for sending queries to databases. Most small and industrial strength database applications can be accessed using SQL statements. SQL is both an ANSI and an ISO standard. However, many database products supporting SQL do so with proprietary extensions to the standard language. Web applications may use user-supplied input to create custom SQL statements for dynamic web page requests.

When a web application fails to properly sanitize user-supplied input, it is possible for an attacker to alter the construction of backend SQL statements. When an attacker is able to modify a SQL statement, the process will run with the same permissions as the component that executed the command. (E.g. Database server, Web application server, Web server, etc.). The impact of this attack can allow attackers to gain total control of the database or even execute commands on the system.

**5- SSI Injection**[7][9]

SSI Injection (Server-side Include) is a server-side exploit technique that allows an attacker to send code into a web application, which will later be executed locally by the web server. SSI Injection exploits a web application's failure to sanitize user-supplied data before they are inserted into a server-side interpreted HTML file.

Before serving an HTML web page, a web server may parse and execute Server-side Include statements before providing it to the user. In some cases (e.g. message boards, guest books, or content management systems), a web application will insert user-supplied data into the source of a web page. If an attacker submits a Server-side Include statement, he may have the ability to execute arbitrary operating system commands, or include a restricted file's contents the next time the page is served.

Example:

The following SSI tag can allow an attacker to get the root directory listing on a

UNIX based system.

<!--#exec cmd="/bin/ls /" -->

The following SSI tag can allow an attacker to obtain database connection strings, or other sensitive data contained within a .NET configuration file.

<!--#INCLUDE VIRTUAL="/web.config"-->

### 1.2.5-Information Disclosure[8][9]

The Information Disclosure attacks are designed to acquire system specific information about a web site. System specific information includes the software distribution; version numbers, and patch levels. Or the information may contain the location of backup files and temporary files. Most web sites will reveal a certain amount of data, but it's best to limit the amount of data whenever possible. The more information about the web site an attacker learns, the easier the system becomes to compromise. This class of attack includes:

#### 1- Information Leakage[9]

Information Leakage is when a web site reveals sensitive data, such as developer comments or error messages, which may aid an attacker in exploiting the system. Sensitive information may be present within HTML comments, error messages, source code, or simply left in plain sight. There are many ways a web site can be coaxed into revealing this type of information. While leakage does not necessarily represent a breach in security, it does give an attacker useful guidance for future exploitation. Leakage of sensitive information may carry various levels of risk and should be limited whenever possible.

**2- Path Traversal**[9]

The Path Traversal attack technique forces access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP based interface is potentially vulnerable to Path Traversal. Most web sites restrict user access to a specific portion of the file system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executables necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-characters sequences.

The most basic Path Traversal attack uses the ".../" special character sequence to alter the resource location requested in the URL. Although most popular web servers will prevent this technique from escaping the web document root, alternate encodings of the "../" sequence may help bypass the security filters. These method variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af") of the forward slash character, backslash characters ("..\") on Windows-based servers, URL encoded characters ("%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character.

**3- Predictable Resource Location**[6][9]

Predictable Resource Location is an attack technique used to uncover hidden web site content and functionality. By making educated guesses, the attack is a brute force search looking for content that is not intended for public viewing. Temporary files, backup files, configuration files, and sample files are all examples of potentially leftover files. These brute force searches are easy because hidden files will often have common naming convention and reside in standard locations. These files may disclose sensitive information about web application internals, database information, passwords, machine names, file paths to other sensitive areas, or possibly contain vulnerabilities. Disclosure of this information is valuable to an attacker. Predictable Resource Location is also known as Forced Browsing, File Enumeration, Directory Enumeration, etc.

### 1.2.6-Logical Attacks[9]

The Logical Attacks focus on the abuse or exploitation of a web application's logic flow. Application logic is the expected procedural flow used in order to perform a certain action. Password recovery, account registration, auction bidding, and eCommerce purchases are all examples of application logic. A web site may require a user to correctly perform a specific multi-step process to complete a particular action. An attacker may be able to circumvent or misuse these features to harm a web site and its users. This class of attack includes:

#### 1- Abuse of Functionality[9]

Abuse of Functionality is an attack technique that uses a web site's own features and functionality to consume, defraud, or circumvents access controls mechanisms. Some functionality of a web site, possibly even security features, may be abused to cause unexpected behavior. When a piece of functionality is open to abuse, an attacker could potentially annoy other users or perhaps defraud the system entirely. The potential and level of abuse will vary from web site to web site and application to application.

#### 2- Denial of Service[9]

Denial of Service (DoS) is an attack technique with the intent of preventing a web site from serving normal user activity. DogS attacks, which are easily normally applied to the network layer, are also possible at the application layer. These malicious attacks can succeed by starving a system of critical resources, vulnerability exploit, or abuse of functionality.

Many times DoS attacks will attempt to consume all of a web site's available system resources such as: CPU, memory, disk space etc. When any one of these critical resources reaches full utilization, the web site will normally be inaccessible.

**3- Insufficient Anti-automation[8][9]**

Insufficient Anti-automation is when a web site permits an attacker to automate a process that should only be performed manually. Certain web site functionalities should be protected against automated attacks.

**4- Insufficient Process Validation[9]**

Insufficient Process Validation is when a web site permits an attacker to bypass or circumvent the intended flow control of an application. If the user state through a process is not verified and enforced, the web site could be vulnerable to exploitation or fraud. When a user performs a certain web site function, the application may expect the user to navigate through a specific order sequence. If the user performs certain steps incorrectly or out of order, a data integrity error occurs. Examples of multi-step processes include wire transfer, password recovery, purchase checkout, account signup, etc. These processes will likely require certain steps to be performed as expected.

## 1.3 Why do Programmers Write Insecure Code?

Many programmers don't intend to write insecure code - but do anyway. Here are a number of purported reasons for this.

- There is no curriculum that addresses computer security in most schools. Even when there is a computer security curriculum, they often don't discuss how to write secure programs as a whole. Many such curriculums only study certain areas such as cryptography or protocols. These are important, but they often fail to discuss common real-world issues such as buffer overflows, string formatting, and input checking. We believe this is one of the most important problems; even those programmers who go through colleges and universities are very unlikely to learn how to write secure programs, yet we depend on those very people to write secure programs.

- Programming books/classes do not teach secure/safe programming techniques. Indeed, until recently there were no books on how to write secure programs at all.
- No one uses formal verification methods.
- Programmers do not think ''multi-user.".
- Programmers are human, and humans are lazy. Thus, programmers will often use the ''easy" approach instead of a secure approach - and once it works, they often fail to fix it later.
- Most programmers are simply not good programmers.
- Most programmers are not security people; they simply don't often think like an attacker does.
- Most security people are not programmers.
- Most computer security models are terrible.
- There is lots of ''broken" legacy software. Fixing this software (to remove security faults or to make it work with more restrictive security policies) is difficult.
- Consumers don't care about security. (Personally, we have hope that consumers are beginning to care about security; a computer system that is constantly exploited is neither useful nor user-friendly. Also, many consumers are unaware that there's even a problem, assume that it can't happen to them, or think that that things cannot be made better.)
- Security costs extra development time.
- Security costs in terms of additional testing.

# 2

# *Chapter Two*

# *System Specification*

# Chapter Two

# 2 System Specifications

## 2.1 Introduction

In this section the system specifications will be explained and identified in more technical terms. This section will cover:

- Objectives: where the main purpose is to help developers to build more secure web applications.
- Project benefits for Users, development Team, and Society
- Functional and Non-Functional Requirements with a brief description of each one.
- Project constrains that mentioned all constrains and validation the system must contain.
- Cost for all system development requirements for HW, SW, Humans, Books, and Others.
- The project cost benefit analysis and the time needed to cover the cost ,then making a justification for each one if it tangible or non-tangible benefits.
- Project Feasibility Study by explained its alternatives (Economic Feasibility, Technical Feasibility, Legal Feasibility ....)
- Project Risk Evaluation: Talks about the possible risks, and the solution for each one.
- Scheduling for the entire project and give a period of time for each step.
- Finally will be the summery and recommendation.

## 2.2 System Objectives

These are the main objectives of the project:

- Enable developers to build more secure web applications.
- Building a program to test web applications for vulnerabilities.
- The program will analyze the HTML code of a web application and shows a list of vulnerabilities if exist (Black Box Test).
- The program will inject the website using different injection codes.
- The program will try to give a suggested solution to some of the security problems.
- If the same problem found more than one time in the same page it will reported only one time.
- Learning and implementing the web application programming security.

## 2.3 System Benefits

### 2.3.1 Benefits for user

- Protect user's personal information.
- Feeling safe when using websites in business.
- Make it more easy for users to get what they need using reliable websites.

### 2.3.2 Benefits for development team

- Create a background about the security concepts that must be considered when developing a web application.
- Know the risks of not considering the security concepts in programming.
- Encourage teaching the security concepts in programming languages courses.
- The project team can use this idea in their future studies (e.g. Master studies).

### 2.3.3 Benefits for society

- Open a new field of research and new ideas for future graduation projects.
- Provide developers with a way to test there programs for vulnerabilities.
- Increase the reliability for using websites by prevailing the security breaches in websites.

## 2.4 Functional Description

This project will test a web application for the following vulnerabilities:

**a) Input field test.:**

This project will test the input fields in the webpage ,This test will firstly checks if the input field size can be changed or not, and then checks request method POST or GET.

**b) Password tests**

The project will test the security of password input tags stored in the database, This function tests the length of the password, if it was less than six characters, an error message will be displayed. It also tests the encryption of the password. And also it tests the default passwords. If any test fails, a problem massage will be displayed.

**c) SSL test**

The project will test if the website uses SSL encryption or not, if so it will test the version of SSL.

**d) Information Leakage test**

The project will test the information leakage in the website, it will test if a web site reveals sensitive data, such as developer comments or error messages, which may aid an attacker in exploiting the system.

**e) Predictable folders location test**

The project will test the Predictable folders location in the website; it will test if the website uses a default and predictable root folders, by searching about these folders.

**f) SQL injection test**

The project will test the ability to inject SQL in the website database, it will Insert SQL query in the input field, then request the page and notify the response, if an error message appeared or not.

**g) Insufficient authentication test**

The project will test if all pages in a secure website are full authenticated, the test will request a page in a secure website, and checks if this request redirected to the main login page or accepted.

**h) Buffer overflow test**

The project will test input field against malicious code , it will insert malicious code in the input field, if it accepts that code without warning, it will be considered not secure against malicious code, else, it will be secure .

**i) Command Execution Test**

The project will test if the website accepts executing system commands on the web server.

**j) Cookies Test**

The project will test if the website store sensitive information using cookies.

**k) Cross Site Scripting Test**

The project will test if the website accepts injected malicious scripts.

## 2.5 Non-Functional Description

**1. Product Requirement**

- **Ease of use**: the system should provide a user-friendly interface for easy use.
- **Integrity**: the system should be integrated with the existing systems.

- **Accuracy**: the system should provide a good level of accuracy.
- **High efficiency**: The test results of the tests should be efficient.
- **High reliability**: the system should be reliable one and its results should be correct.
- **High portability:** The system will be portable since it uses HTML code for testing.
- **Errors**: The errors of the code should be minimized as possible.
- **Fast:** Obtaining the result in a short time as possible.
- **Simple (understandable)**: The system should be understandable.

2. **Process Requirements**

➢ The system and its documents should be delivered at scheduled date.

3. **External Requirements**

➢ **Ethical requirements**: the system developers are not responsible for unethical use of this software.

4. **Legislative Requirements**

➢ **Safety**: the system should not cause harms to the tested web application.

## 2.6 Project Constraints

This section lists the constraints that exist in the project and a description of each:

- The system should test only the links related to one web application.
- The project must be completed at the end of this semester.

## 2.7 Development Requirement and Cost

### 1. Hardware

| Item | Number of units | Unit Cost | Available | Subtotal |
|---|---|---|---|---|
| Desktop computers<br><br>P4, 2.8 GHz, 512 MB 256 MB RAM, 20G HD | 1 | $700 | Yes | $700 |
| Total | | | | $700 |

**Table 2.1:** Development Hardware Cost

### 2. Software

| Item | Number of units | Unit Cost | Available | Subtotal |
|---|---|---|---|---|
| Windows XP Professional | 1 | $350 | Yes | $350 |
| Visual Studio2003 | 1 | $1799 | Yes | $1799 |
| Microsoft Word2003 | 1 | $229 | Yes | $229 |
| PowerPoint | 1 | $229 | Yes | $229 |
| Total | | | | $2677 |

**Table 2.2:** Development software Cost

3. **Humans**

| Member | Number | Cost $ | Available | Subtotal |
|---|---|---|---|---|
| Computer Engineering Students | 4 | 2500 | Yes | 10000 |
| Supervisor | 1 | -- | Yes | -- |
| Total | | | | 10000 |

**Table 2.3**: Development Human Cost

4. **Others**

There is another 1000 NIS to cover other areas (internet, transportation, papers, pens, printing...etc). The following table summarizes the development cost.

| Components | Total |
|---|---|
| HW | $700 |
| SW | $2677 |
| Humans | $10000 |
| Others | $222 |
| Total | $13599 |

**Table 2.4**: Development Cost Summery

## 2.8 Costs-Benefit Analysis

This section contains the project cost benefit analysis and the time needed to cover the cost.

- The project has great impact in the web application field since it tests the quality of a website.
- It improves the security level of the websites.
- It provides a list of the vulnerabilities in the websites.

## 2.9 Feasibility Study

### 2.9.1 Economic Feasibility

As it pointed in the Development Requirements and Cost, the total cost of the project is $13599.The hardware, software, humans and other requirements are all available so the project could complete in satisfying way.

### 2.9.2 Technical Feasibility

The project required good programming capabilities and experience in dealing with algorithms, databases, indexing and good background of security concepts

The project team has good background in programming languages such as C, VB.NET and databases applications such oracle, access, SQL server2000, and study as many papers as possible in the field of web application security.

### 2.9.3 Legal Feasibility

This project take the nature of research type one which is open for any student .so there is no need to take a licensee from any except the university and there is no illegal issues.

## 2.10 Risk Evaluations

This section discusses the risks that may appear in the project and the possible solutions:

- **Hardware Failure**

To avoid this risk we will make a continuous daily backup of the project on flashes and other hard disks.

- **Shortage of development time**

The time given for us was not enough to compete all of the project objectives, so we worked very hardly to introduce the project in a required way.

# 3

# *Chapter Three*

# *Software Requirements Specification*

# Chapter 3

# 3 Software Requirements Specification

## 3.1 Introduction

We collect and analyze the data to determine the specifications of the required software. This chapter documents the functional detail description and details for each validation and constraints, the system data flow diagram, data dictionary, the database requirements and dictionary.

## 3.2 Functional Details Description

This section lists the major functions in the project and a description for each.

| | |
|---|---|
| **Function:** | Main crawler |
| **Type:** | Windows application |
| **Description:** | The main function that crawls and mines the whole website. |
| **Input:** | URL. |
| **Source:** | windows Form. |
| **Output:** | Two tables: table of all references and table of all input fields in the website. |
| **Destination:** | _____ |
| **Require:** | Valid URL |
| **Precondition:** | HTTP/HTTPS |
| **Post condition:** | _____. |
| **Procedure:** | This function will accept a valid URL from the user, and sends a request to the server, then it searches for all references and input tags in the website, and stores them in a database. |

| | |
|---|---|
| **Function:** | Crawl a webpage. |
| **Type:** | Windows application |
| **Description:** | This function will store all the hyperlinks founded in the web page. |
| **Input:** | URL. |
| **Source:** | Database. |
| **Output:** | List of all links in the site. |
| **Destination:** | Database |
| **Requires:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The resulted links can be used in mining. |
| **Procedure:** | This function identifies all the hyperlinks in the page and if they are not duplicated ones it adds them to the list of URLs to visit. the process either ended manually or after a certain number of links has been followed. The list of the URLs stored in the database will be used in the mining process. |
| **Validation:** | Every page must be crawled one time and no duplication of links. |

| | |
|---|---|
| **Function:** | Mine a page for input field. |
| **Type:** | Windows application |
| **Description:** | This function used to look for input tags in the WebPages. |
| **Input:** | URL. |
| **Source:** | Database. |
| **Output:** | WebPages contain the input tags. |
| **Destination:** | Database |
| **Requires:** | Webpage. |
| **Precondition:** | WebPages come out of a crawler. |
| **Post condition:** | The resulted WebPages can be added to the index. |
| **Procedure:** | This function finds all input fields in the WebPages, and if not duplicated it stores them in a database, for further tests. |
| **Validation:** | Every input field must be added to the database. |

| | |
|---|---|
| **Function:** | Input field test. |
| **Type:** | Windows application |
| **Description:** | This function used to test the input fields in the webpage. |
| **Input:** | Web page. |
| **Source:** | Database. |
| **Output:** | Test results. |
| **Destination:** | Database |
| **Requires:** | Input tags. |
| **Precondition:** | Input tags come out of a tag table. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function determines if the input field can be tested, if so, it determines its type, password or normal input field, and then make the necessary tests. |

| | |
|---|---|
| **Function:** | Normal Input field test. |
| **Type:** | Windows application |
| **Description:** | This function checks request method and the ability of changing input field properties. |
| **Input:** | Normal input tags. |
| **Source:** | Database. |
| **Output:** | Results of the test. |
| **Destination:** | Output text message. |
| **Requires:** | Normal input tags. |
| **Precondition:** | The tag should be in tag table. |
| **Post condition:** | Ability to make new tests. |
| **Procedure:** | This function will firstly checks if the input field size can be changed or not, and then checks request method POST or GET. |

| | |
|---|---|
| **Function:** | Password tests. |
| **Type:** | Windows application |
| **Description:** | This function uses to test the security of password input tags stored in the database. |
| **Input:** | Password input tags. |
| **Source:** | Database. |
| **Output:** | Results of the test. |
| **Destination:** | Output text message. |
| **Requires:** | Password input tags. |
| **Precondition:** | The tag should be in tag table. |
| **Post condition:** | Ability to make new tests. |
| **Procedure:** | This function tests the length of the password, if it was less than six characters, an error message will be displayed. It also tests the encryption of the password. And also it tests the default passwords. If any test fails, a problem massage will be displayed. |

| | |
|---|---|
| **Function:** | SSL tests. |
| **Type:** | Windows application. |
| **Description:** | This function make tests related to SSL. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Output text message. |
| **Requires:** | URL. |
| **Precondition:** | Valid URL. |
| **Post condition:** | If SSL used, the version will be determined. |
| **Procedure:** | This Function will test if the website uses SSL encryption or not, if so it will test the Version of SSL. |

| | |
|---|---|
| **Function:** | Information Leakage test. |
| **Type:** | Windows application |
| **Description:** | This function will test the information leakage in the website. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will test if a web site reveals sensitive data, such as developer comments or error messages, which may aid an attacker in exploiting the system. |

| | |
|---|---|
| **Function:** | Predictable folders location test. |
| **Type:** | Windows application |
| **Description:** | This function will test the Predictable folders location in the website. |
| **Input:** | Predictable folders of the website. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL, Predictable folders of the website. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will test if the website uses a default and predictable root folders, by searching about these folders. |

| | |
|---|---|
| **Function:** | SQL injection test. |
| **Type:** | Windows application |
| **Description:** | This function will test the ability to inject SQL in the website database. |
| **Input:** | Input tags. |
| **Source:** | Database. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Insert valid SQL query in the URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will Insert SQL query in the input field, then request the page and notify the response, if an error message appeared or not. |

| | |
|---|---|
| **Function:** | Insufficient authentication test. |
| **Type:** | Windows application |
| **Description:** | This function will test if all pages in a secure website are full authenticated. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | valid user into a webpage. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will request a page in a secure website, and checks if this request redirected to the main login page or accepted. |

| | |
|---|---|
| **Function:** | Buffer overflow test. |
| **Type:** | Windows application |
| **Description:** | This function will test URL against malicious code. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | Nothing. |
| **Procedure:** | This function will insert malicious code in the URL, if it accepts that code without warning, it will be considered not secure against malicious code, and else, it will be secure. |

| | |
|---|---|
| **Function:** | Cookies test. |
| **Type:** | Windows application |
| **Description:** | This function will display information about cookies. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will request URL cookies, and test if the cookies contain sensitive information that should not be prevailed to the user |

| | |
|---|---|
| **Function:** | Command Execution Test. |
| **Type:** | Windows application |
| **Description:** | This function will test website against Command Execution Vulnerability. |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will make many URL request with different OS commands, and test if the Website accepts these executing commands |

| | |
|---|---|
| **Function:** | Cross Site Scripting Test. |
| **Type:** | Windows application |
| **Description:** | This function will test website against malicious scripts |
| **Input:** | URL. |
| **Source:** | Windows Form. |
| **Output:** | Test results. |
| **Destination:** | Windows form. |
| **Require:** | Valid URL. |
| **Precondition:** | HTTP/HTTPS. |
| **Post condition:** | The results can be used for further tests. |
| **Procedure:** | This function will make many URL request with different scripts, and test if the Website accepts these scripts. |

## 3.3 Project Constraints

This section lists the constraints that exist in the project and a description of each:

- The crawling, mining and testing part are a windows application using VB.ENT.
- No redundancy in the references table.
- No redundancy in the input tags table.
- No redundancy in test results.
- The test result messages should be meaningful.
- Unethical use of this software should be prevented.
- We are not responsible for any bad use for this software.
- The project must be completed at the end of semester.

## 3.4 System processing Diagram

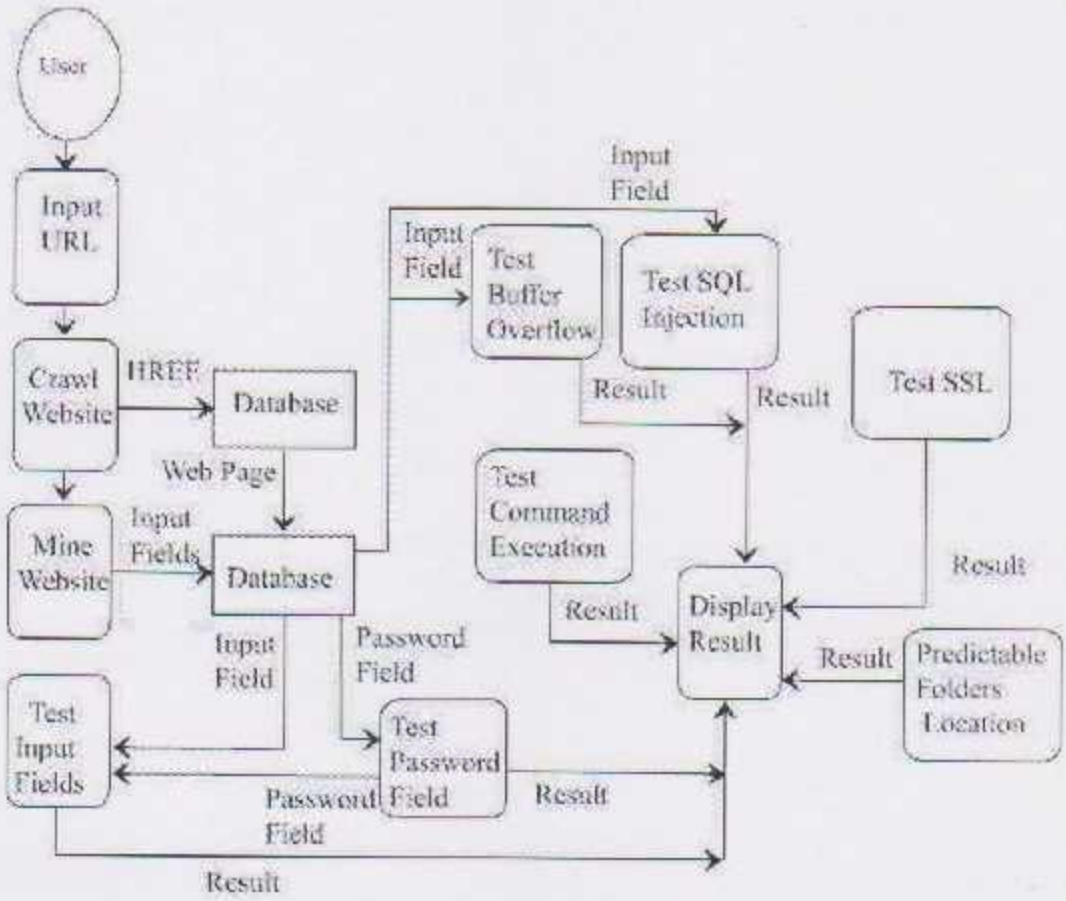


**Fig 3.1**: Processing diagram

| Entity name | Description |
|---|---|
| Crawl website | The project will store all the hyperlinks founded in the web page. |
| Mine website | The project will look for input tags in the WebPages. |
| Test input field | The project will checks request method and the ability Of changing input field properties. |
| Test password field | The project will test. |
| Test buffer overflow | The project will test input field against malicious code |
| Test SQL injection | The project will test the ability to inject SQL in the website database. |
| Test SSL | The project will test if the website uses SSL encryption or not. |
| Test command execution | The project will test website against Command Execution Vulnerability. |
| Predictable folders location | The project will test the Predictable folders location in the website. |
| Cookies Test | The project will display information about cookies |
| Cross Site Scripting | will test website against malicious scripts |

**Table3.1**: Data Dictionary.

| Data Item | Type |
|---|---|
| Reference ID | Integer |
| Hyperlink | VarChar |
| Page ID | Integer |
| Input Tag ID | Integer |
| Input Tag Value | VarChar |

**Table3.2:** Database Data Dictionary

## 3.5 Database Requirements

**References Table (Reference ID, Hyperlink)**

**Reference ID:** The number of the reference

**Hyperlink:** The URL of a webpage.

**Input Tags Table (Input Tag ID, Page ID, Input Tag Value)**

**Input Tag ID:** The number of the tag.

**Page ID:** The number of the page containing the input tag.

**Input Tag Value:** The value of the tag.

## 3.6 Summary and Recommendation

- **Summary**

  - Each function described in details: the function name, description, its input and output, source, destination, require, precondition, post condition, and its procedure.
  - The section lists the whole possible constraints that may exist, and also describes each one.
  - Those constraints represented in login and passwords for both administrator and power user.
  - Passwords must at least contain six characters (Alpha Numeric Mixture) but different from login to maintain the security and difficulty to be stolen.
  - Regular expressions explained in the help for the pattern section.
  - The data dictionary implies the whole entity names of the system with their type and description.
  - Starting websites needs the name and number of that websites to be crawled, also the status of a website either active to be crawled or not.
  - The database data dictionary table explains the type of each data item in the system either string, integer, or encoded.

- **Recommendation**

All requirement specifications are discussed in details, according to the project supervisor permission the team project could pass to the next chapter.

# 4

# *Chapter Four*

# *System Design*

# Chapter Four

# 4 System Design

## 4.1 Introduction

In this section the functions design will be implemented using functional oriented methodology, where each function will be designed accordingly.

This section will cover the following:

- **Input /output design:** a design for the input /output screens.
- **Database design:** a complete design for the database tables includes all tables and their fields.
- **Functions design:** where each function will be designed by using a flow chart, its interface, and constraints will be identified.

## 4.2 Input/Output Design

- **Main Screen**

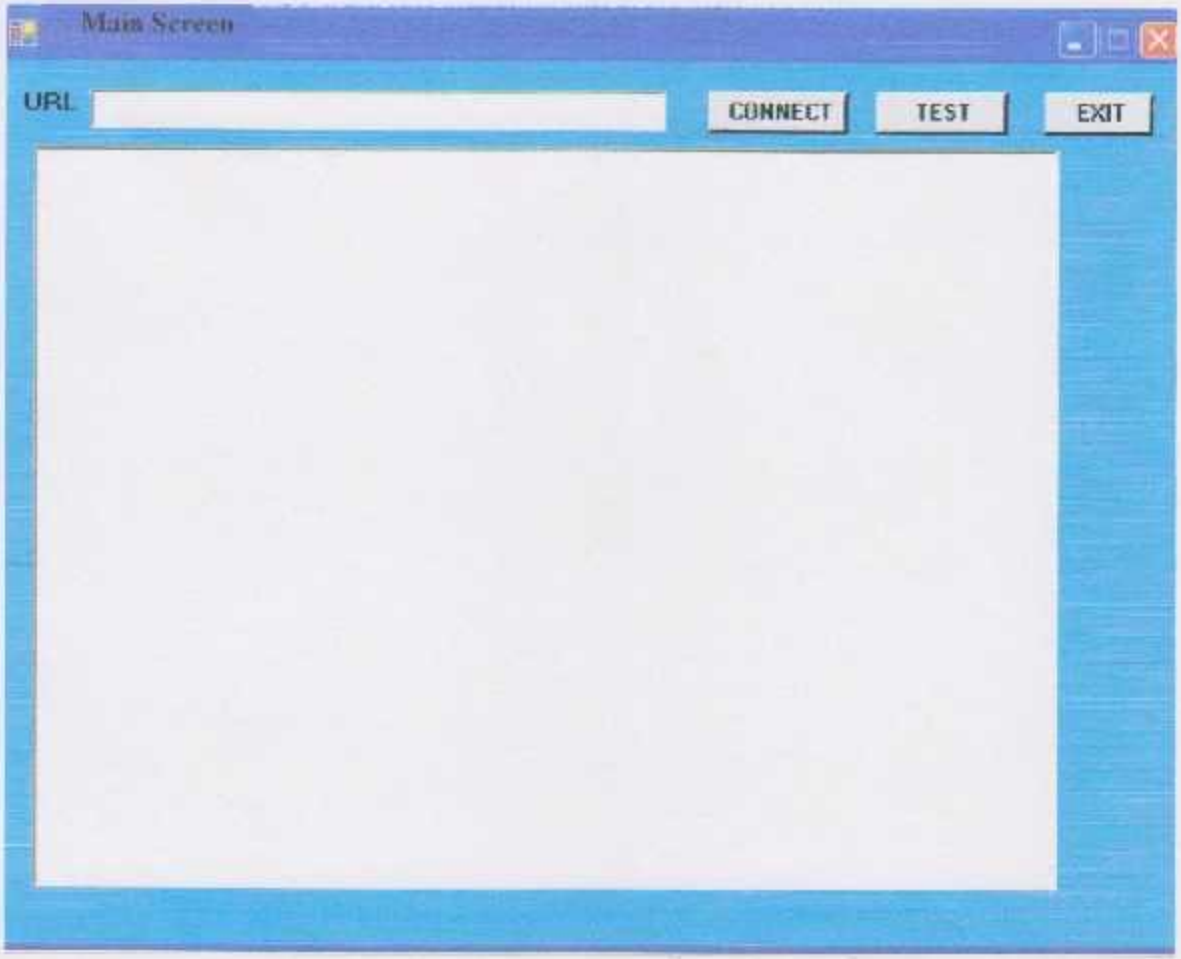


**Figure 4.1:** Main Screen

- **Tests Screen**

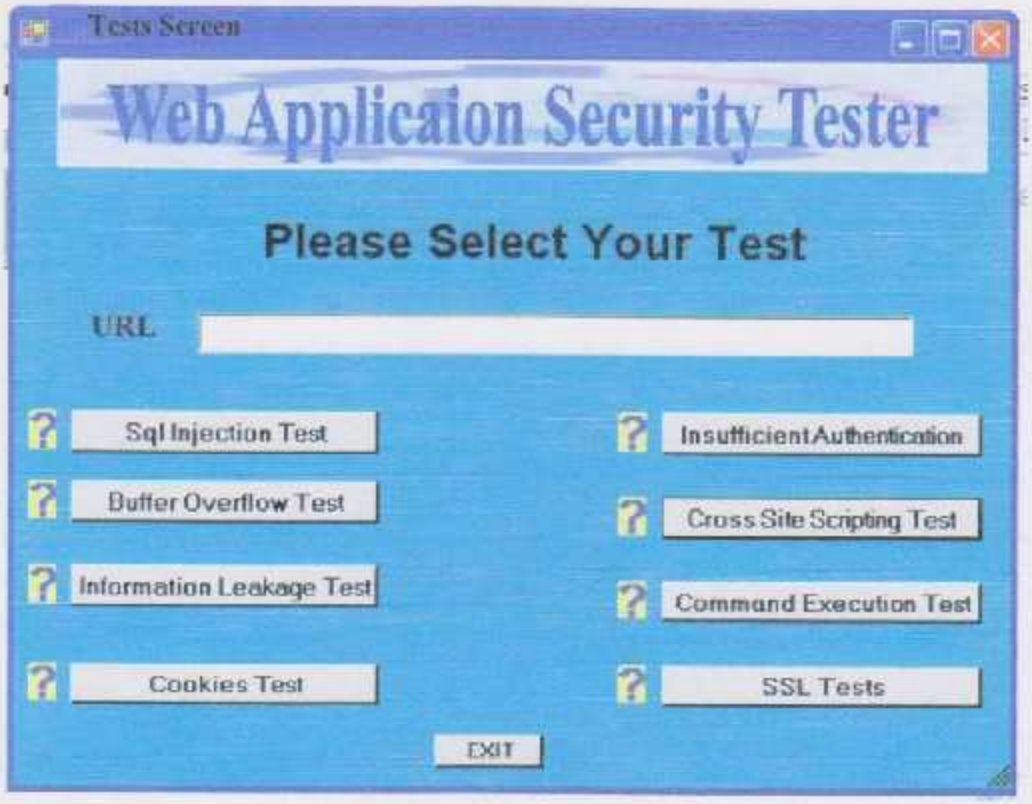


**Figure 4.2:** Tests Screen

- **SQL Injection Test Result Screen**

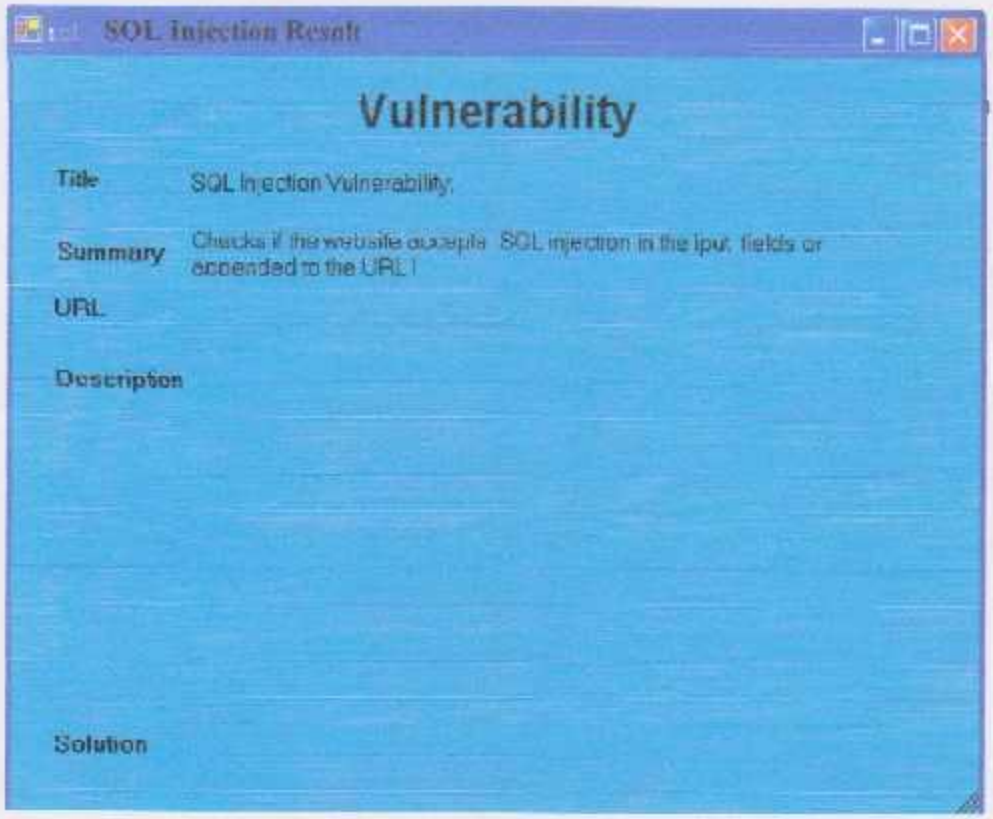


**Figure 4.3:** SQL Injection Test Result Screen

- **Buffer Overflow Test Result Screen**

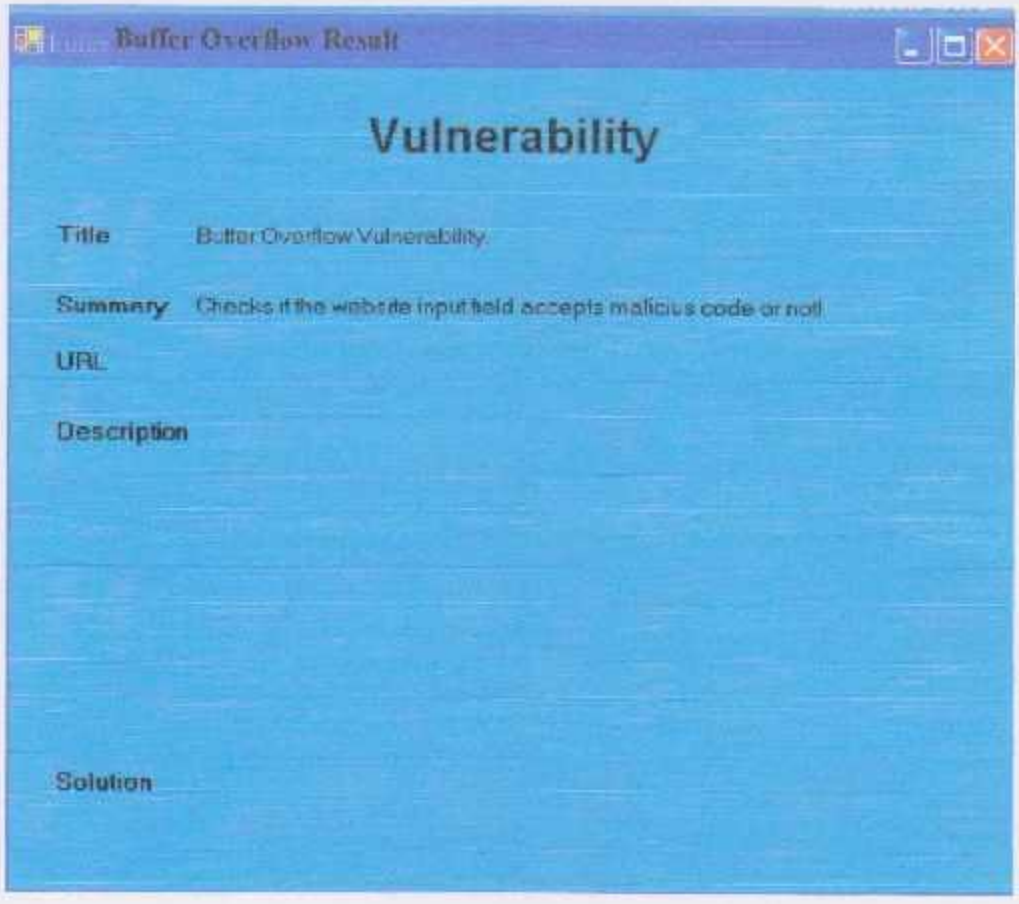


**Figure 4.4:** Buffer Overflow Test Result Screen

- **Information Leakage Test Result Screen**

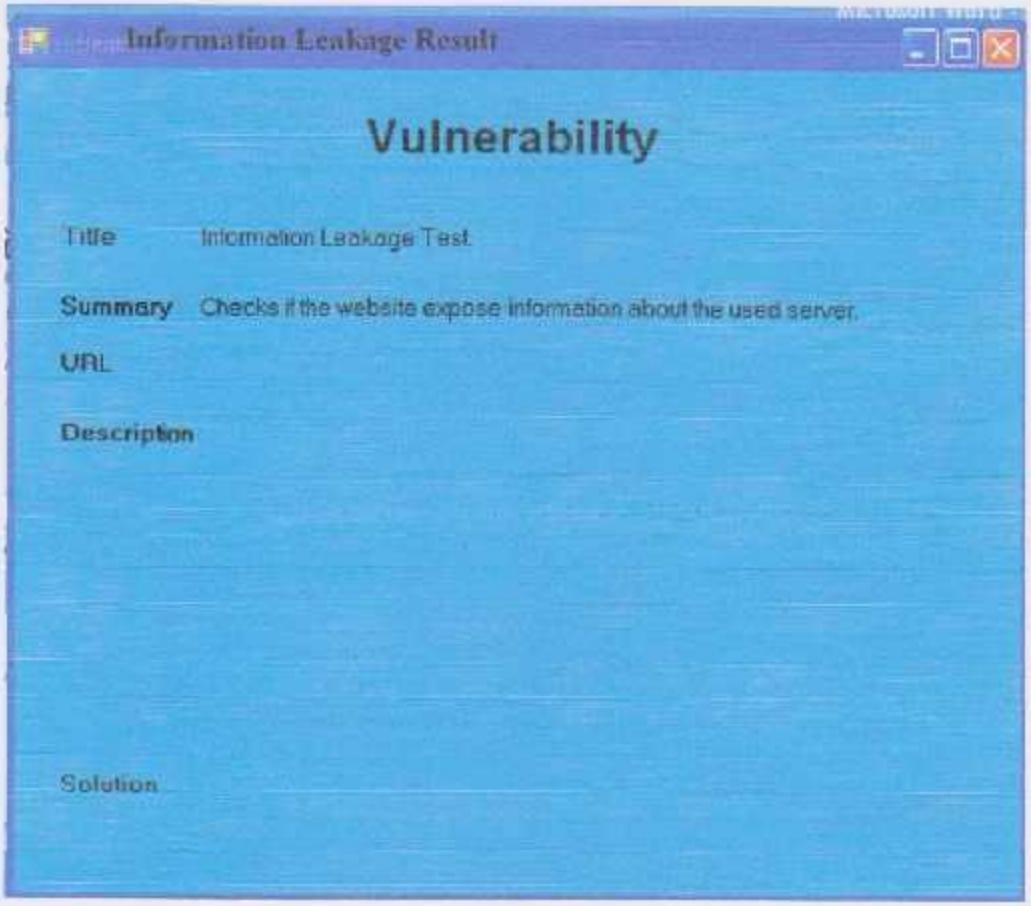


**Figure 4.5:** Information Leakage Test Result Screen

- **Cookies Test Result Screen**

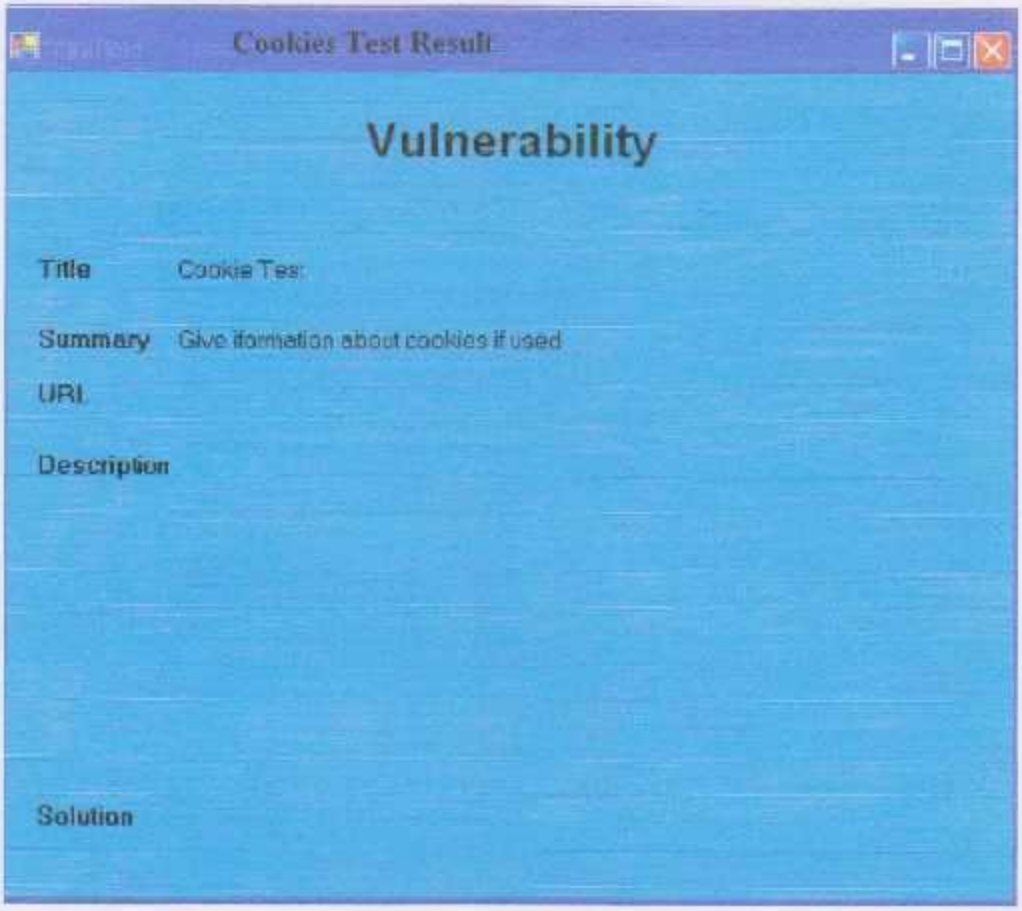


**Figure 4.6:** Cookies Test Result Screen

- **Insufficient Authentication Test Result Screen**

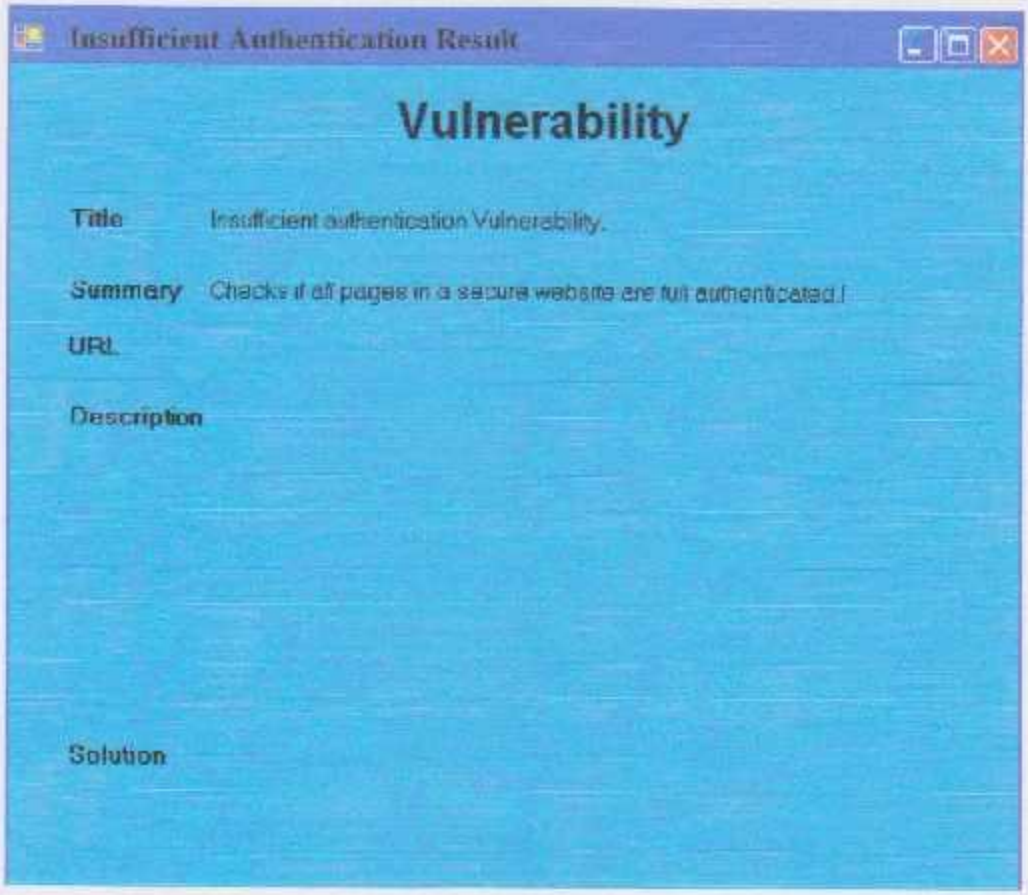


**Figure 4.7:** Insufficient Authentication Test Result Screen

- **Cross Site Scripting Result Screen**

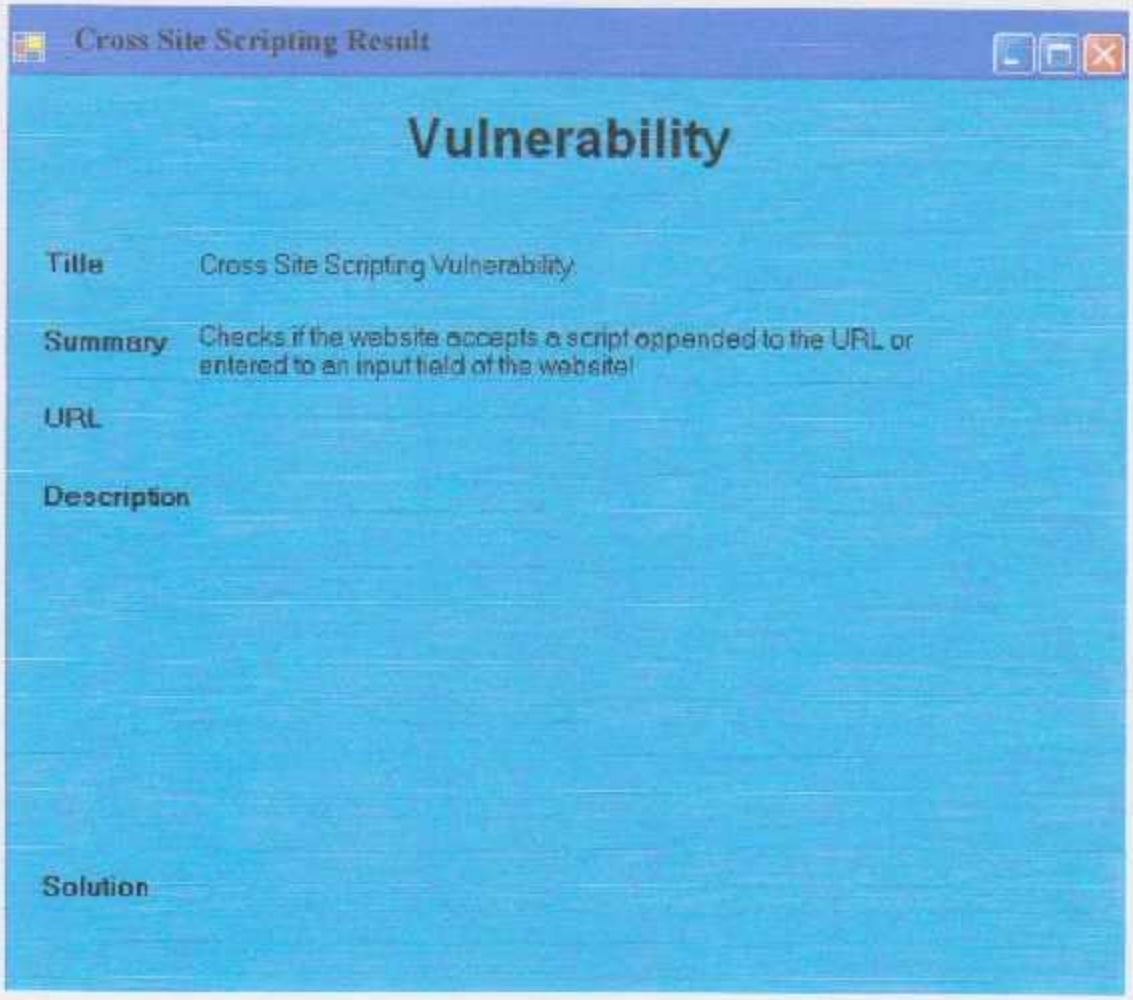


**Figure 4.8:** Cross Site Scripting Result Screen

- **Command Execution Test Result Screen**

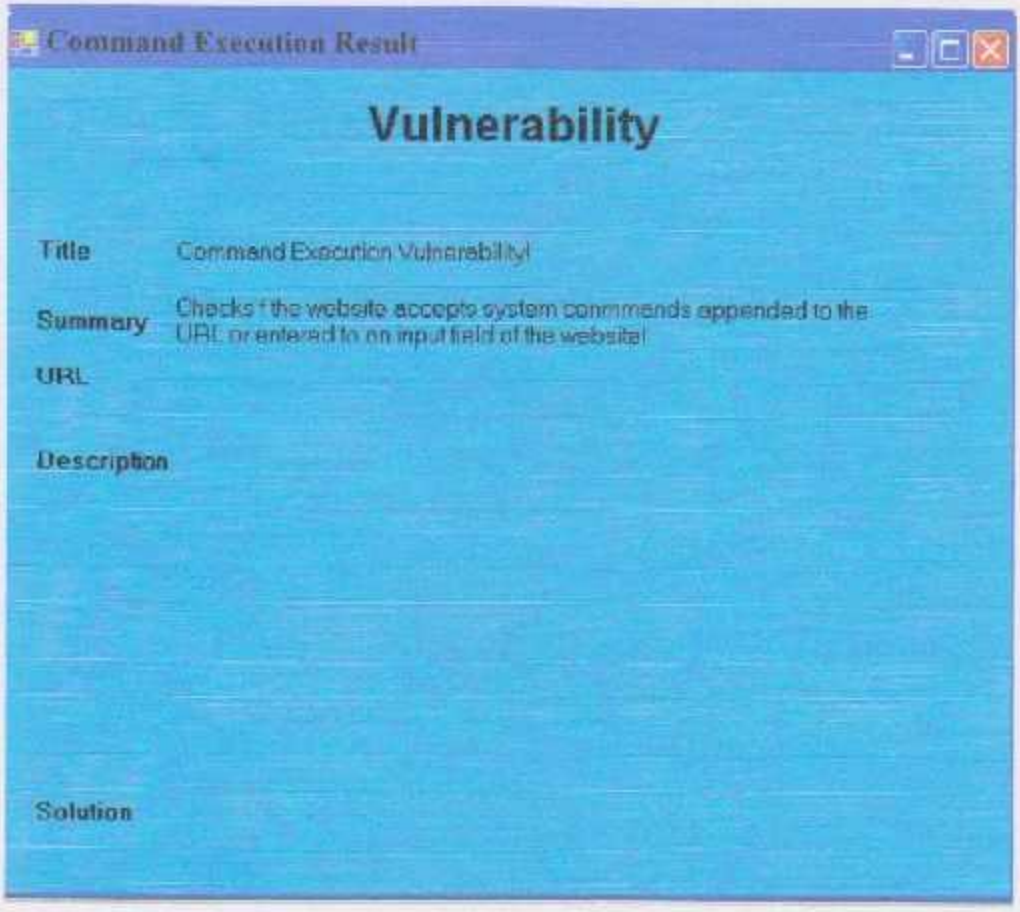


**Figure 4.9:** Command Execution Test Result Screen

- **SSL Test Result Screen**

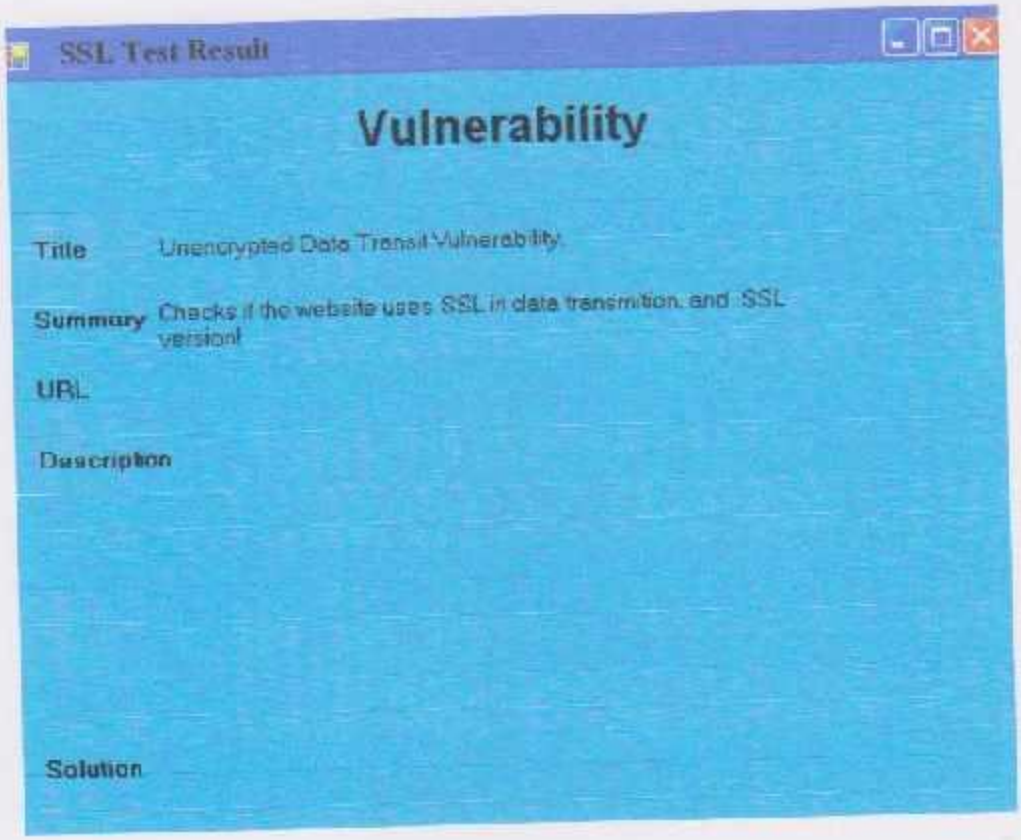


**Figure 4.10:** SSL Test Result Screen

## 4.3 Database Design

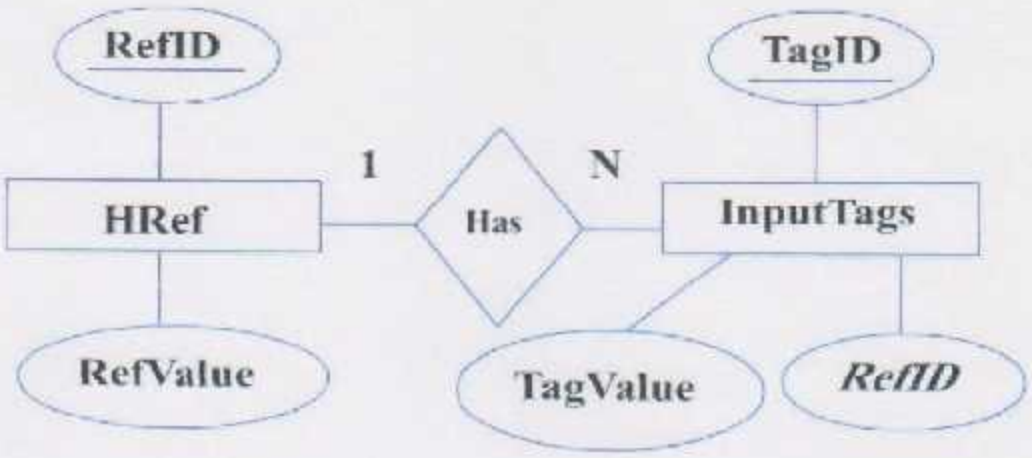


Figure 4.11: ER-Diagram

- **References Table**

| Field Name | Type | Size | Control |
|---|---|---|---|
| RefID | Int | 4 | PK |
| RefValue | Varchar | 100 | —— |

**Table 4.1:** References Table

- **Input Fields Table**

| Field Name | Type | Size | Control |
|---|---|---|---|
| TagID | Int | 4 | PK |
| TagValue | Varchar | 100 | —— |
| RefID | int | 4 | FK |

**Table 4.2:** Input Fields Table

## 4.4 Functional Design

- **Main Input Screen**

  The user will enter a valid URL in the URL text field, and then choose one of three functions:

  - CONNECT: Connect to the server, and display the HTML source code.
  - TEST: Shows the tests screen.
  - EXIT: Exit the application.

1. **Interface:**
   - **Input:** Valid URL
   - **Output:** Source code or screen contains all tests
2. **Constraints:**
   - The URL must begin with (http://).
   - The user should press CONNECT before TEST.
   - Login and password must be validated

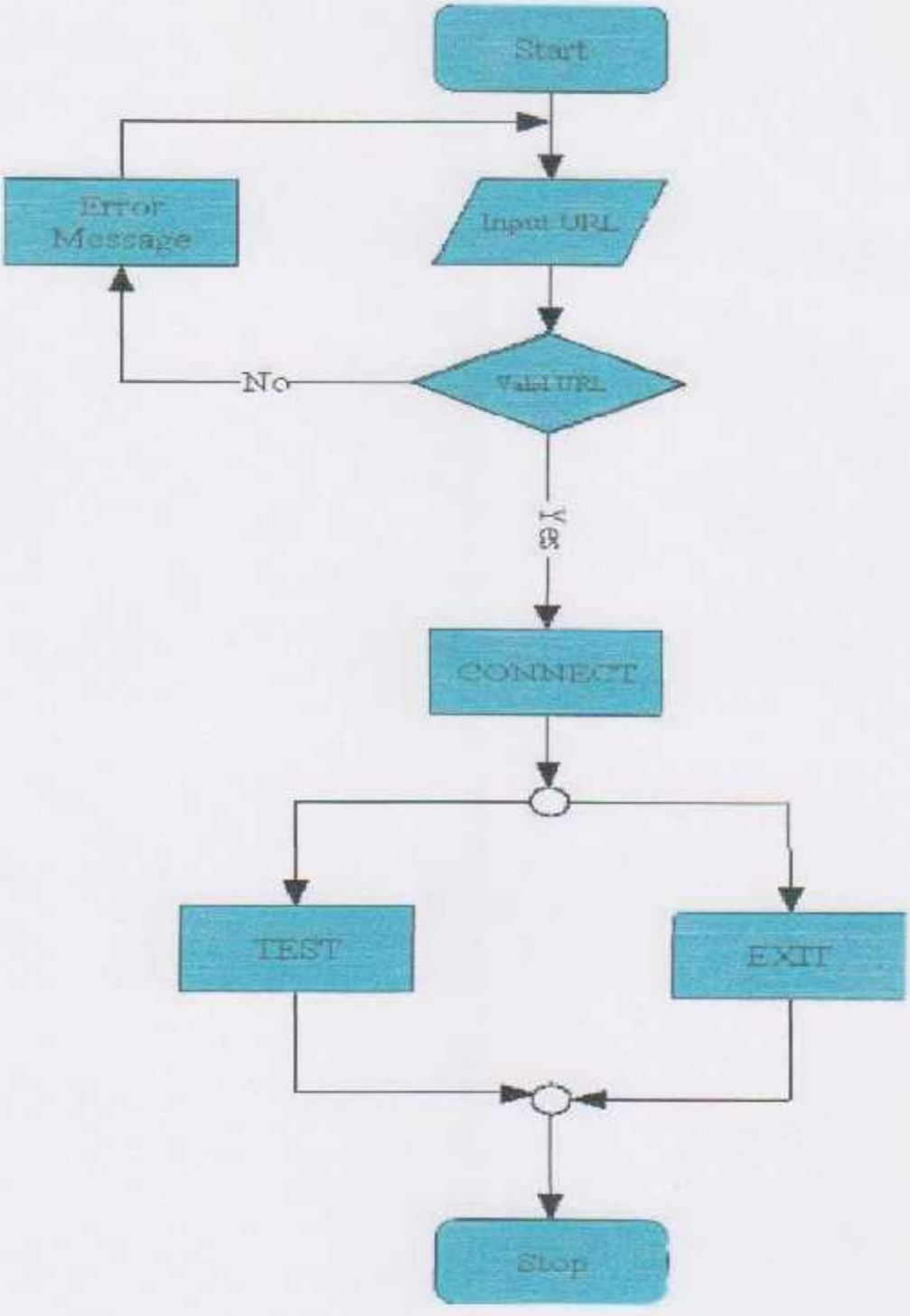


**Figure 4.12:** Main Input Screen flowchart

- **Test a Website**

This function enables the user to choose one of eight tests to test a website.

1. **Interface:**
    - **Input:** Choose one of the tests.
    - **Output:** Test Result.

2. **Constraints:**
    - One of tests should be chosen.
    - The user can't change the URL at this stage.
    - Information about each test displayed when a question mark is pressed.

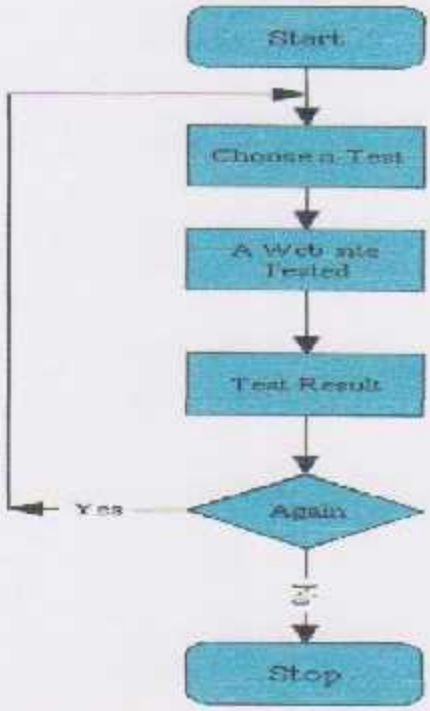


**Figure 4.13:** Test a Website Flowchart

- **Crawl a website**

  This function will store all the hyperlinks founded in the web page.

  1. **Interface:**
     - **Input:** URL.
     - **Output:** List of all links in the webpage and the WebPages contain the links.

  2. **Constraints:**
     - Every page must be crawled one time.
     - Only links related to the website will be stored.

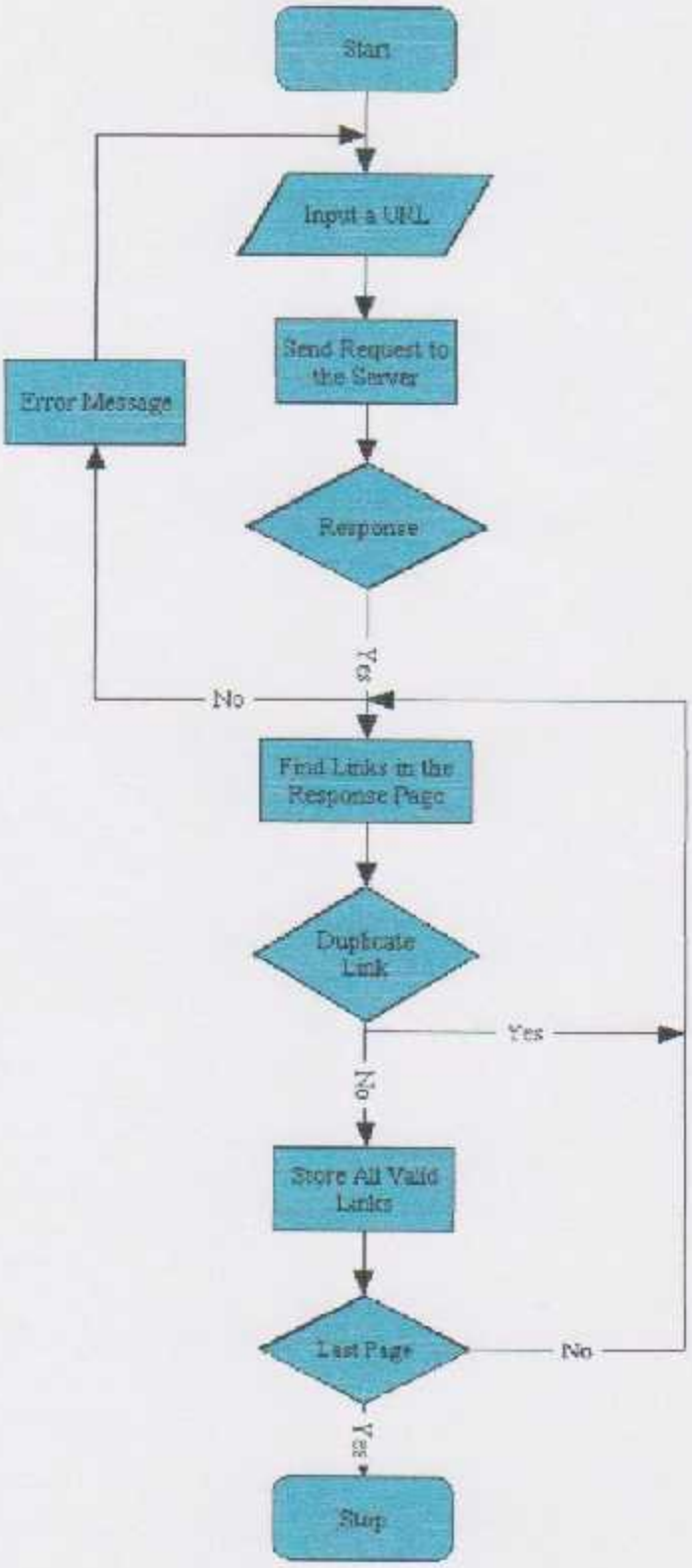


**Figure 4.14:** Crawl a Website Flowchart

- **Mine a page for input tags**

  This function used to extract all input tags from a webpage.

1. **Interface:**
   - **Input:** Web page.
   - **Output:** All input tags, and WebPages contain the input tags.

2. **Constraints:**
   - Every page contains input tags must be added to DB without redundancy.
   - Every input tag must be added to DB without redundancy

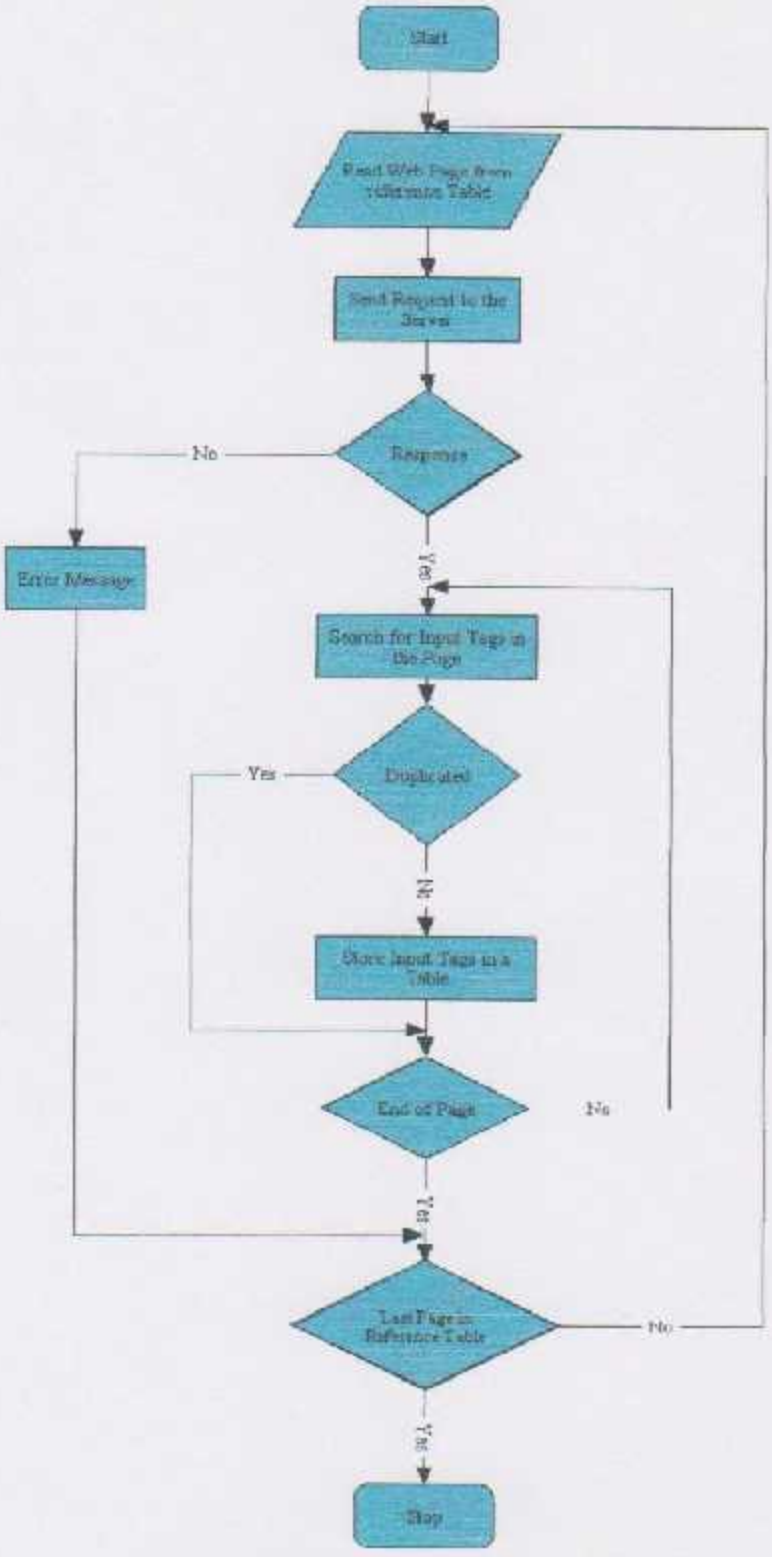


**Figure 4.15:** Mining for Input Tags Flowchart

- **SQL Injection Test**

This function used to test the website against SQL injection commands.

- **Interface:**
  - **Input:** URL, SQL injection code.
  - **Output:** Test result.
- **Constraints:**
  - URL must be entered before entering SQL injection code


**Figure 4.16:** SQL Injection Test Flowchart

- **Buffer Overflow Test**

This function used to test the website against malicious code.

1. **Interface:**
   - **Input:** URL, malicious code.
   - **Output:** Test result.

2. **Constraints:**
   - This test must be done if input field found.

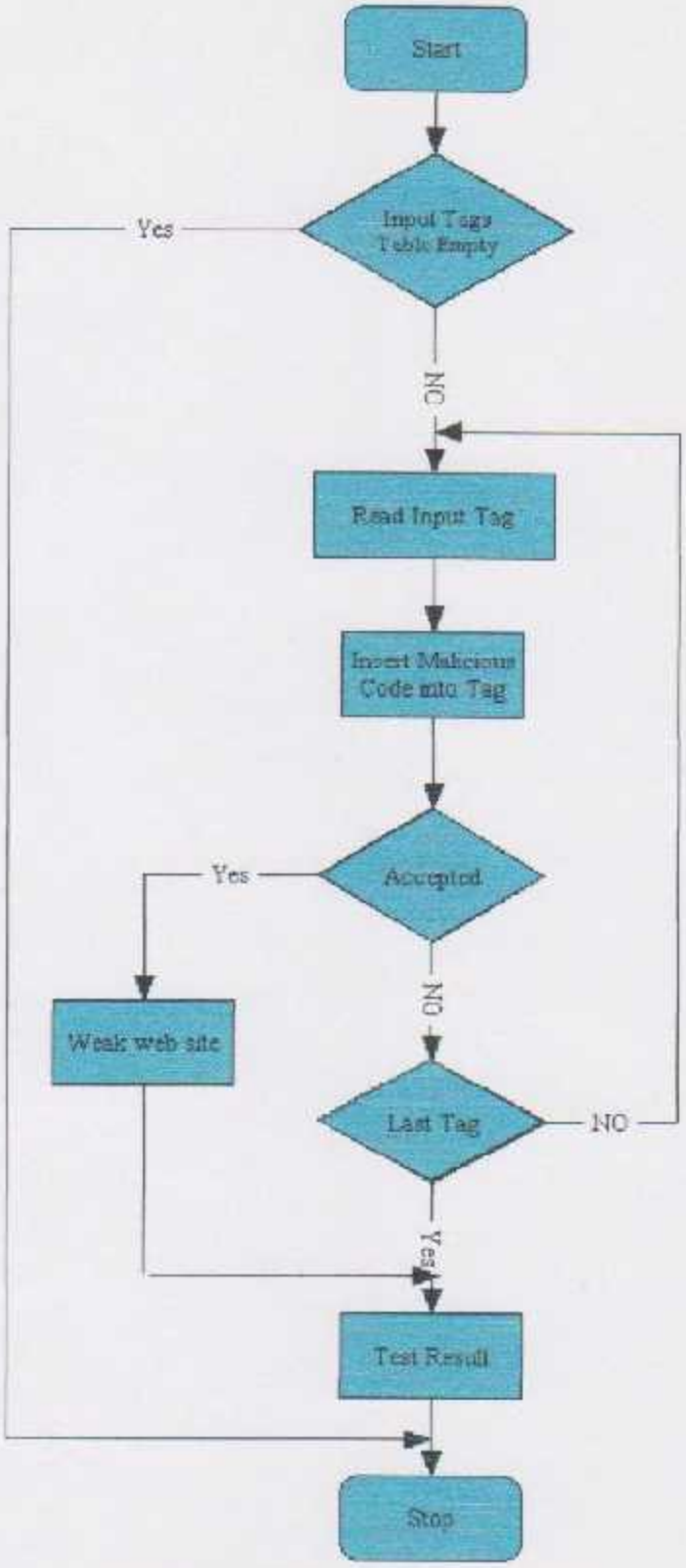


**Figure 4.17:** Buffer Overflow Test Flowchart

- **Information Leakage Test**

This function tests if the website exposes information about its server.

**1. Interface:**

- **Input:** Website.
- **Output:** Test result.

**2. Constraints:**

- Valid URL must be entered.
- These information should not be used for hacking purposes.

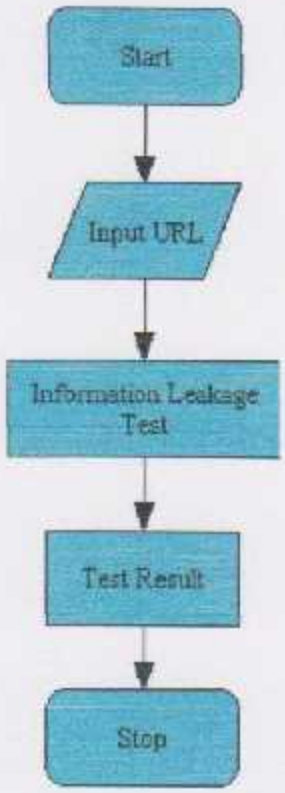


**Figure 4.18:** Information Leakage Test Flowchart

- **Cookies Test**

This function enables the user to obtain information about cookies in the website.

**1. Interface:**

- **Input:** Website.
- **Output:** Test result.

**2. Constraints:**

- Valid URL must be entered.
- The passed if the website uses cookies.

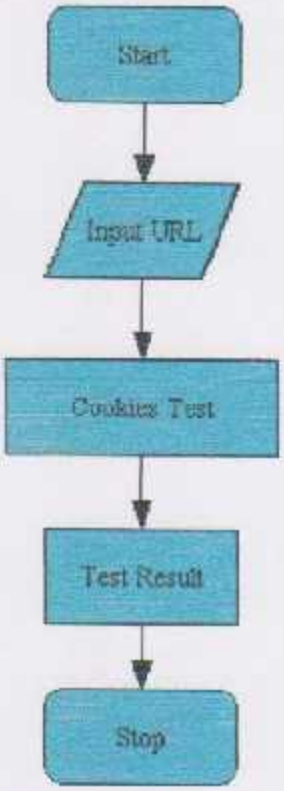


**Figure 4.19:** Cookies Test Flowchart

- **Insufficient Authentication Test**

This function tests if all pages in a secure website are forbidden for unauthenticated users.

**1. Interface**

- **Input:** Secure page in a secure website.
- **Output:** Test result

**2. Constraints:**

- The requested page should be a secure page in a secure website.

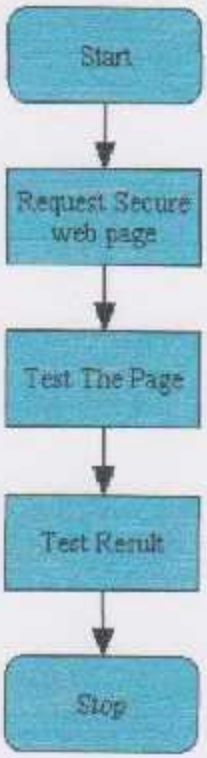


**Figure 4.20:** Insufficient Authentication Test Flowchart

- **Cross Site Scripting Test (XSS)**

This function enables user to test if the website accepts scripts in the URL or during input fields.

**1. Interface:**

- **Input:** Website, Script.
- **Output:** Test result.

**2. Constraints:**

- The scripts depend on the scripting language used in developing the website, and also on the operating system used.

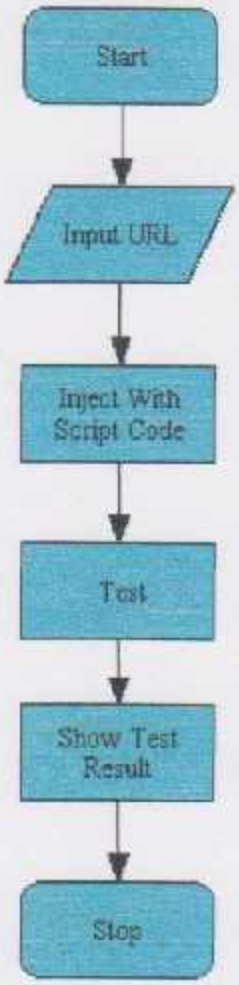


**Figure 4.21:** Cross Site Scripting Test Flowchart

- **Command Execution Test**

This function enables power user to search a specific web site using pattern .

**1. Interface:**

- **Input:** website, OS or language command.
- **Output:** Test result.

**2. Constraints:**

- The commands depend on the language used for developing the website, and also on the operating system used


**Figure 4.22:** Command Execution Test Flowchart

- **SSL Test**

This function enables user to test if the web site transmit data over a secure channel (SSL), and the version of SSL.

**1. Interface:**

- **Input:** Website
- **Output:** Test result.

**2. Constraints:**

- Data must be transmitted over secure channel on the network.
- SSL affected data only on the channel.

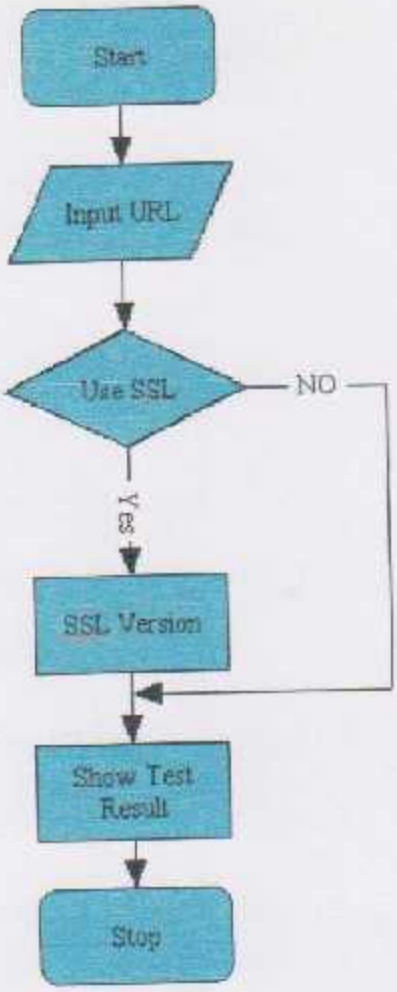


**Figure 4.23:** SSL Test Flowchart

- **Help**

This function enables user to get information about all the tests used.

1. **Interface:**
   - **Input:** Website, Tests form.
   - **Output:** Information Message.

2. **Constraints:**
   - The user should press on the question mark to get information abut the test.

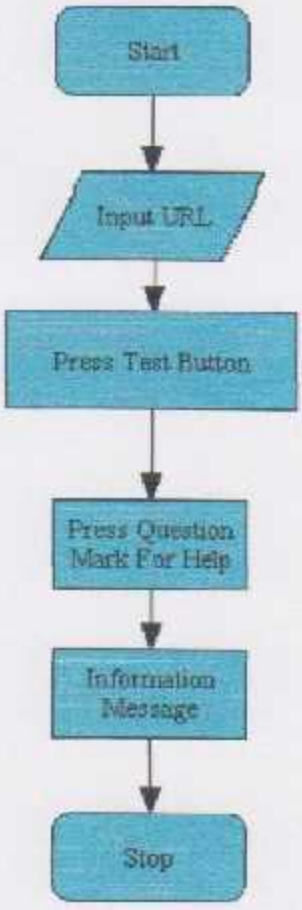


**Figure 4.24:** Get Help Flowchart

## 4.5 Summary and Recommendation:

- Each function is explained and designed accordingly. Using functional oriented methodology.
- Each function is designed for its input /output screens.
- There is a complete design for the database includes all tables and their fields.
- A flow chart for each function, also its interface and constraints is identified.
- All constraints are explained clearly in this section to omit any ambiguity.

All system design requirements are discussed in details with the supervisor. The supervisor gave the team permission to pursue work for implementation.

# 5

# Chapter Five

# Coding and Implementation

# Chapter Five

# Coding and Implementation

## 5.1 Introduction

This chapter covers the following subjects:

- Coding Programming Language.
- Databases System.
- Establishment of Development Environment.
- Database Creation and Configuration.
- Coding and Unit Testing.

## 5.2 Coding Programming Language

There are several programming languages that can be used in the system to provide what the programmers need , the most suitable one is Visual Studio.NET since it has different features especially the ability of dealing with web applications, web services, and database. It includes enhancements to Visual Basic, Visual C#, and Visual C++, as well as a new programming language.

The WEB APPLICATION SECURITY TESTER system chooses the ASP.Net and Visual Basic.Net as two environments of programming.

### 5.2.1 ASP.NET

ASP.NET, the next version of ASP, is a programming framework used to create enterprise-class Web Applications. These applications are accessible on a global basis leading to efficient information management. The advantages of ASP.NET offers are more than just the next version of ASP.

- **Why Using ASP.NET?**

Since 1995, Microsoft has been constantly working to shift it's focus from Windows-based platforms to the Internet. As a result, Microsoft introduced ASP (Active Server Pages) in November 1996. ASP offered the efficiency of ISAPI applications along with a new level of simplicity that made it easy to understand and use. However, ASP script was an interpreted script and consisted unstructured code and was difficult to debug and maintain. As the web consists of many different technologies, software integration for Web development was complicated and required to understand many different technologies. Also, as applications grew bigger in size and became more complex, the number of lines of source code in ASP applications increased dramatically and was hard to maintain. Therefore, an architecture was needed that would allow development of Web applications in a structured and consistent way.

The .NET Framework was introduced with a vision to create globally distributed software with Internet functionality and interoperability. The .NET Framework consists of many class libraries, includes multiple language support and a common execution platform. It's a very flexible foundation on which many different types of top class applications can be developed that do different things. Developing Internet applications with the .NET Framework is very easy. ASP.NET is built into this framework; we can create ASP.NET applications using any of the built-in languages.

Unlike ASP, ASP.NET uses the Common Language Runtime (CLR) provided by the .NET Framework. This CLR manages execution of the code we write. ASP.NET code is a compiled CLR code instead of interpreted code (ASP). CLR also allows objects written in different languages to interact with each other. The CLR makes development of Web applications simple.

- **Advantages Using ASP.NET**

  - ASP.NET drastically reduces the amount of code required to build large applications

  - ASP.NET makes development simpler and easier to maintain with an event-driven, server-side programming model

  - ASP.NET pages are easy to write and maintain because the source code and HTML are together

  - The source code is executed on the server. The pages have lots of power and flexibility by this approach

  - The source code is compiled the first time the page is requested. Execution is fast as the Web Server compiles the page the first time it is requested. The server saves the compiled version of the page for use next time the page is requested

  - The HTML produced by the ASP.NET page is sent back to the browser. The application source code you write is not sent and is not easily stolen

  - ASP.NET makes for easy deployment. There is no need to register components because the configuration information is built-in

  - The Web server continuously monitors the pages, components and applications running on it. If it notices memory leaks, infinite loops, other illegal software or activities, it seamlessly kills those activities and restarts itself

  - ASP.NET validates information (validation controls) entered by the user without writing a single line of code

  - ASP.NET easily works with ADO .NET using data-binding and page formatting features

  - ASP.NET applications run faster and counters large volumes of users without performance problems

- **Differences between ASP.NET and Client-Side Technologies**

Client-side refers to the browser and the machine running the browser. Server-side on the other hand refers to a Web server.

- **Client-Side Scripting**

JavaScript and VBScript and generally used for Client-side scripting. Client-side scripting executes in the browser after the page is loaded. Using client-side scripting you can add some cool features to your page. Both, HTML and the script are together in the same file and the script is downloading as part of the page which anyone can view. A client-side script runs only on a browser that supports scripting and specifically the scripting language that is used. Since the script is in the same file as the HTML and as it executes on the machine you use, the page may take longer time to download.

- **Server-Side Scripting**

ASP.NET is purely server-side technology. ASP.NET code executes on the server before it is sent to the browser. The code that is sent back to the browser is pure HTML and not ASP.NET code. Like client-side scripting, ASP.NET code is similar in a way that it allows you to write your code alongside HTML. Unlike client-side scripting, ASP.NET code is executed on the server and not in the browser. The script that you write alongside your HTML is not sent back to the browser and that prevents others from stealing the code you developed.

### 5.2.2 Visual Basic.NET

Visual Basic.NET is an extension of Visual Basic programming language with many new features in it. The changes from VB to VB .NET are huge, ranging from the change in syntax of the language to the types of projects we can create now and the way we design applications. Visual Basic .NET was designed to take advantage of the .NET

Framework base classes and runtime environment. It comes with power packed features that simplify application development.

The biggest change from VB to VB .NET is VB .NET is Object-Oriented now. VB .NET now supports all the key OOP features like Inheritance, Polymorphism, Abstraction and Encapsulation. We can now create classes and objects, derive classes from other classes and so on. The major advantage of OOP is code reusability. VB .NET now uses ADO .NET; a new data handling model to communicate with databases on local machines or on a network and also it makes handling of data on the Internet easy. All the data in ADO .NET is represented in XML format and is exchanged in the same format. Representing data in XML format allows us for sending large amounts of data on the Internet and it also reduces network traffic when communicating with the database VB .NET now supports Multithreading. A threaded application allows to do number of different things at once, running different execution threads allowing using system resources. Web Development is now an integral part of VB .NET making Web Forms and Web Services two major types of applications. VB .NET is strongly typed which means that we need to declare all the variables by default before using them, and supports structured exception handling using Try...Catch...Finally syntax

## 5.3 Databases System

### 5.3.1 Database

The database consists of a collection of tables that contain data and other objects, such as views, indexes, stored procedures, and triggers, defined to support activities performed with the data. The data stored in a database is usually related to a particular subject or process. Database can store either interrelated or unrelated data from other databases. For example, a server can have one database that stores personnel data and another that stores product-related data. Alternatively, one database can store current customer order data, and another related database can store historical customer orders used for yearly reporting.

### 5.3.2 Microsoft SQL Server 2000

Microsoft SQL Server 2000 extends the performance, reliability, quality, and ease-of-use of Microsoft SQL Server version 7.0. Microsoft SQL Server 2000 includes several new features that make it an excellent database platform for large-scale online transactional processing (OLTP), data warehousing, and e-commerce applications.

The OLTP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component.

The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services.

### 5.3.3 Authentication Mode

There are two types of authentication mode, each one deals in a different way with users account.

1. **Windows Authentication Mode:** When a user connects through a Microsoft Windows user account, SQL Server validates the account name and password using information in the Windows operating system. It uses a single account for all users called ASPNET user.

2. **Mixed Authentication Mode**: Allows users to connect using Windows Authentication or SQL Server Authentication. Users who connect through a Microsoft Windows user account can make use of trusted connections (connections validated by Windows) in either Windows Authentication Mode or Mixed Mode.

## 5.4 Establishment of Development Environment

1. Microsoft Windows server family or XP.
2. Internet Information Service (IIS).
3. .NET Framework.
4. SQL Server 2000.

## 5.5 Database Creation and Configuration

### 5.5.1 Database Creation

Right click on databases, new database then type the name of database WEB APPLICATION SECURITY TESTER project, database name is WEB APPLICATION SECURITY TESTER.

### 5.5.2 Tables Creation

Right click on tables, new table then type the name of your table and may create more than table as needed, then type all columns name and its type if it is integer, char, text. Each with its limiting length .The last field is very important that used to allowing to this column to be null or not. Before saving it, primary key field have to be chosen and if there is relation with another table must be justify. WEB APPLICATION SECURITY TESTER project has a multiple of tables such as References Table, and Input Field Table. All of these fields mentioned in chapter four.

The following table shows the whole tables that created in the system database.

| Name | Primarykey | Foreignkey | Description |
|---|---|---|---|
| InputTags Table | InputTagid | Referenceid | Stores the input tags in the website |
| References Table | Referenceid | ---------- | Stores the pages in the website. |

Table 5.1 : Database System Tables

### 5.5.3 Database Diagram

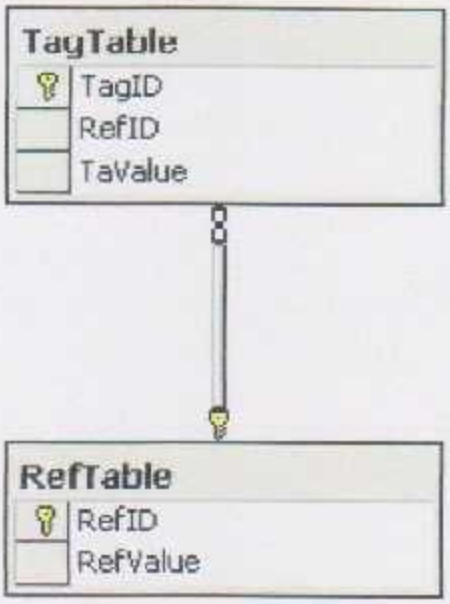


Figure 5.1: Database Diagram

## 5.6 Coding and Unit Testing

All system units and modules were tested against its specifications, the test ensured that the units and modules performed as expected. The following table shows some of screens that had been tested successfully.

| Screen | Function | Status | Figure |
|---|---|---|---|
| Main Screen.vb | To enter URL, connect, and test | Done | 5.1 |
| Tests Screen.vb | To choose one of tests | Done | 5.2 |
| SQL Injection Result.vb | To test SQL injection | Done | 53 |
| Buffer Overflow Result.vb | To test the overflow | Done | 5.4 |
| Information Leakage Result | To test if can get information about the server | Done | 5.5 |
| Cookies Test Result.vb | To get information about cookies | Done | 5.6 |
| Insufficient Authentication Result .vb | To test authenticated website | Done | 5.7 |
| Cross Site Scripting help.vb | To test XSS help | Done | 5.8 |

**Table 5.2** Coding and Unit Testing

**1- Main Screen**

- URL input text field: Enables the user to input HTTP/HTTPS webpage.
- Connect Button :
    - Sends web request to the server.
    - If response code 2XX then.
    - Read the contents of the response into an input stream.
    - Out the contents of the stream to a RichTextBox.
    - Call crawl function from a Func Class.
    - Call mine function Func Class.
    - If response status is 4XX, then show error message.
- TEST Button: Shows AllTests form.
- EXIT Button: terminates application.

**2- All Tests Screen.**

- SQL Injection Test Button: Shows SQL Injection Test form.
- Buffer Overflow Test Button: Shows Buffer Overflow Test form.
- Information Leakage Test Button: Shows Information Leakage form.
- Cookies Test Button: Shows Cookies Test form.
- Insufficient Authentication Test Button: Shows Insufficient Authentication Test form.
- Cross Site Scripting Test Button: Shows Cross Site Scripting Test form.
- Command Execution Test Button: Shows Command Execution Test form.
- SSL Tests Button: Shows Command Execution Test form.
- Question marks: shows Help message to the user.
- RETURN Button: Shows Main Screen form.
- EXIT Button: Terminates application.

### 3- SQL Injection Test Form

- TEST Button
    - Initialize a five SQL injection codes and the Progress Bar.
    - Send malicious HTTP requests by appending a code to the URL of the supplied webpage. Each malicious request is sent separately
    - For each input field in the web site submit SQL injection code values
    - Submitting a large input data to the input filed will affect the server
    - Each request is between TRY and CATCH.
    - The Progress Bar is increased by one after each request.
    - For each request if a response code of 2XX is received show test success result to a RichTextBox.
    - For each request if a response code of 4XX is received show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

**4- Buffer Overflow Test**

- TEST Button
    - Initialize a large size one line text file and the Progress Bar.
    - Send malicious HTTP requests by appending the code to the URL of the supplied webpage.
    - For each input field in the web site check the following:
        - Is there is a max value to the input filed?
        - Is the max length of the input filed can be changed?
        - Submitting a large input data to the input filed will affect the server?
    - Each request is between TRY and CATCH.
    - The Progress Bar is increased by one after each request.
    - For each request if a response code of 2XX is received show test success result to a RichTextBox.
    - For each request if a response code of 4XX is received show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

**5- Information Leakage Form**

- TEST Button
    - Send malicious HTTP requests by using the URL of the supplied webpage.
    - Each request is between TRY and CATCH.
    - If response contains the server version or other special information, show a test success message.
    - If response doesn't contain the server version or other special information, show a test fail message.

- RETURN Button: Shows All Tests form.

**6- Cookies Test**

- TEST Button
    - Create CookieContainer object
    - Send malicious HTTP requests by the URL of the supplied webpage.
    - For each cookie in the response check the following:
        - Is the port number is supplied within the cookie?
        - Is it a secure cookie?
        - Is the cookie value contains password data?
    - Each request is between TRY and CATCH.
    - The Progress Bar is increased by one after each request.

- RETURN Button: Shows All Tests form.

**7- Insufficient Authentication Test**

- TEST Button
    - Initialize paths of the names of login and passwords files and other secrete files.
    - Send malicious HTTP requests by appending the paths to the URL of the supplied webpage.
    - For each password input field in the web site check the following:
        - Is special characters are acceptable?
        - Is the maximum length of the password filed can be changed?
        - Is there is a maximum and minimum length to the password filed?
    - Each request is between TRY and CATCH.
    - The Progress Bar is increased by one after each request.
    - For each request if a response code of 2XX is received show test success result to a RichTextBox.
    - For each request if a response code of 4XX is received show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

**8- Cross Site Scripting Test**

- TEST Button
    - Initialize 15 scripts injection codes and the Progress Bar.
    - Send malicious HTTP requests by appending a code to the URL of the supplied webpage. Each malicious request is sent separately
    - Each request is between TRY and CATCH.
    - The Progress Bar is increased by one after each request.
    - For each request if a response code of 2XX is received show test success result to a RichTextBox.
    - For each request if a response code of 4XX is received show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

**9- Command Execution Test**

- TEST Button
    - Initialize 10 system command s an OS commands injection codes.
    - Send malicious HTTP requests by appending command to the URL of the supplied webpage. Each malicious request is sent separately
    - Each request is between TRY and CATCH.
    - For each request if a response code of 2XX is received show test success result to a RichTextBox.
    - For each request if a response code of 4XX is received show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

10- SSL Test

- TEST Button
    - Send malicious HTTP requests by appending a code to the URL of the supplied webpage. Each malicious request is sent separately
    - The request is between TRY and CATCH
    - Check if the response page use SSL and the version of the SSL
    - Check the version of the server and the cookie if it contain information a bout the version of SSL.
    - If the version of SSL is lower than 3.0 received show test success result to a RichTextBox.
    - If the version of SSL is 3.0 received or the version can't be determined show test failed result to a RichTextBox.
- RETURN Button: Shows All Tests form.

## 5.7 Summary and Recommendation

- The designers use the visual studio.net as a programming language to build the system.
- Visual Basic.NET used by the developers to manage the whole system.
- SQL server is used to link the .NET with Database environment as a method of building a complete program.

The project's supervisor puts his signature on the chapter and gives his permission to the team to begin the next chapter

# 6

# *Chapter Six*

# *Testing*

# Chapter Six

# 6 Testing

## 6.1 Introduction

The WEB APPLICATION SECURITY TESTER system must be tested after the coding and implementation stage to ensure that every part of the system performs as expected to be, and to check its functionality works properly.

## 6.2 Testing Plan

### 6.2.1 Unit Testing

WEB APPLICATION SECURITY TESTER system has two main types of screen results used to integrate the system process. These parts are tested separately to ensure the functionality of each component. This section will show the forms before testing and the results of the test.
The two main parts are:

1. **Testing Result Screens Part**

This part shows some test results screen, we used different websites for testing, like:
http://www.ppu.edu
http://www.msn.com
and others. Some of result screens shown in chapter 5.

The next table shows the windows-screens that has been tested

| Screen Name | Test Result |
|---|---|
| CONNECT | Done successfully |
| TEST | Done successfully |
| Cross Site Scripting Test | Done successfully |
| Command Execution Test | Done successfully |
| SSL Test | Done successfully |

The following figures are some windows samples after testing.

- **After Pressing CONNECT Button:**

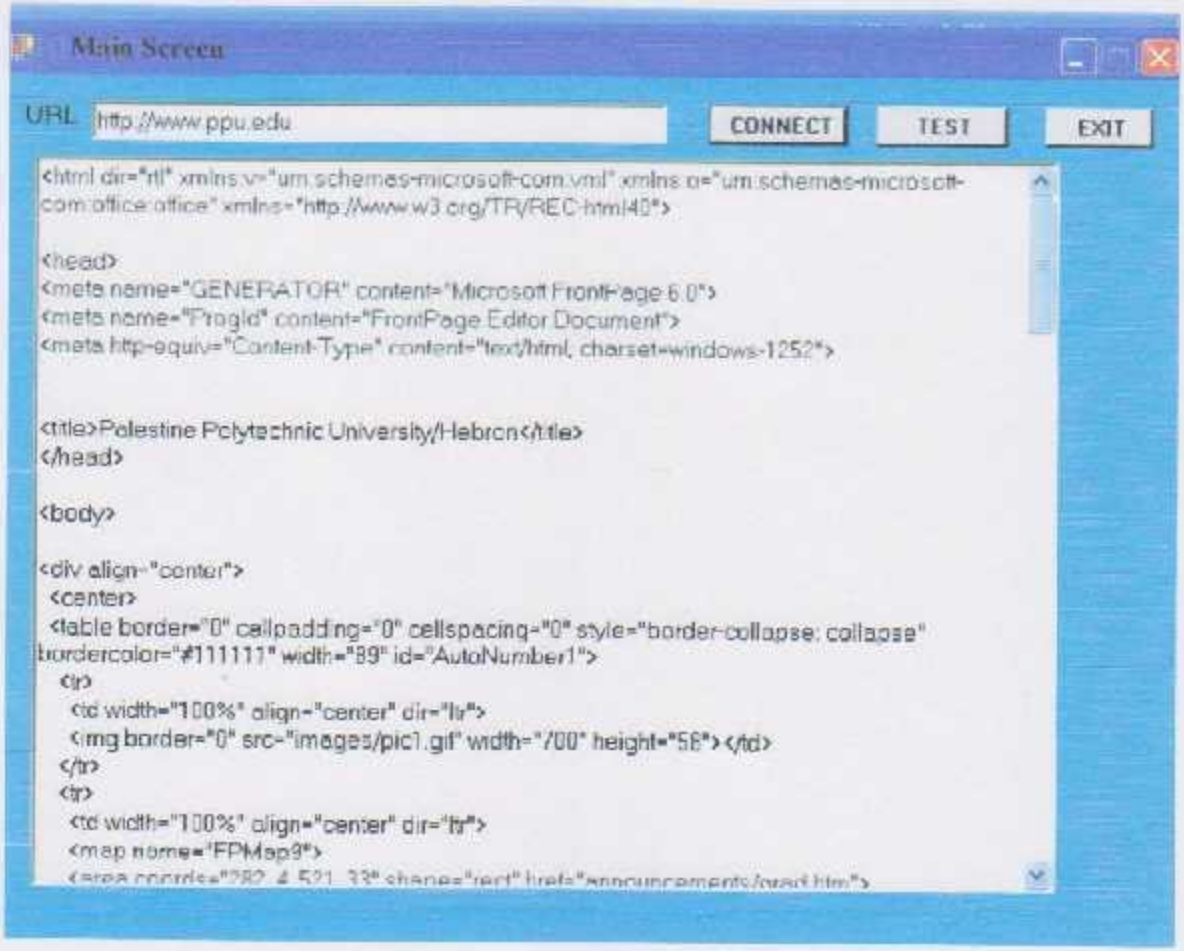


**Figure 6.1:** CONNECT Testing

- **After Pressing TEST Button**

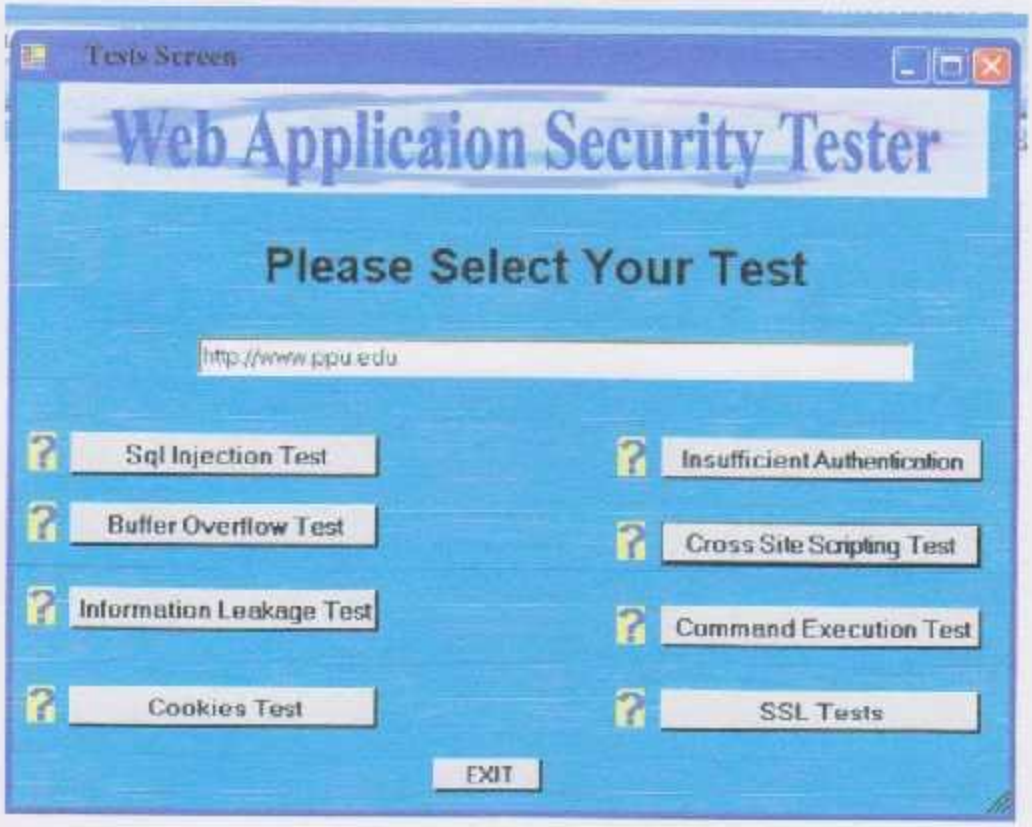


**Figure 6.2** TEST Button Testing

- **Cross Site Script Test on http://www.ppu.edu**

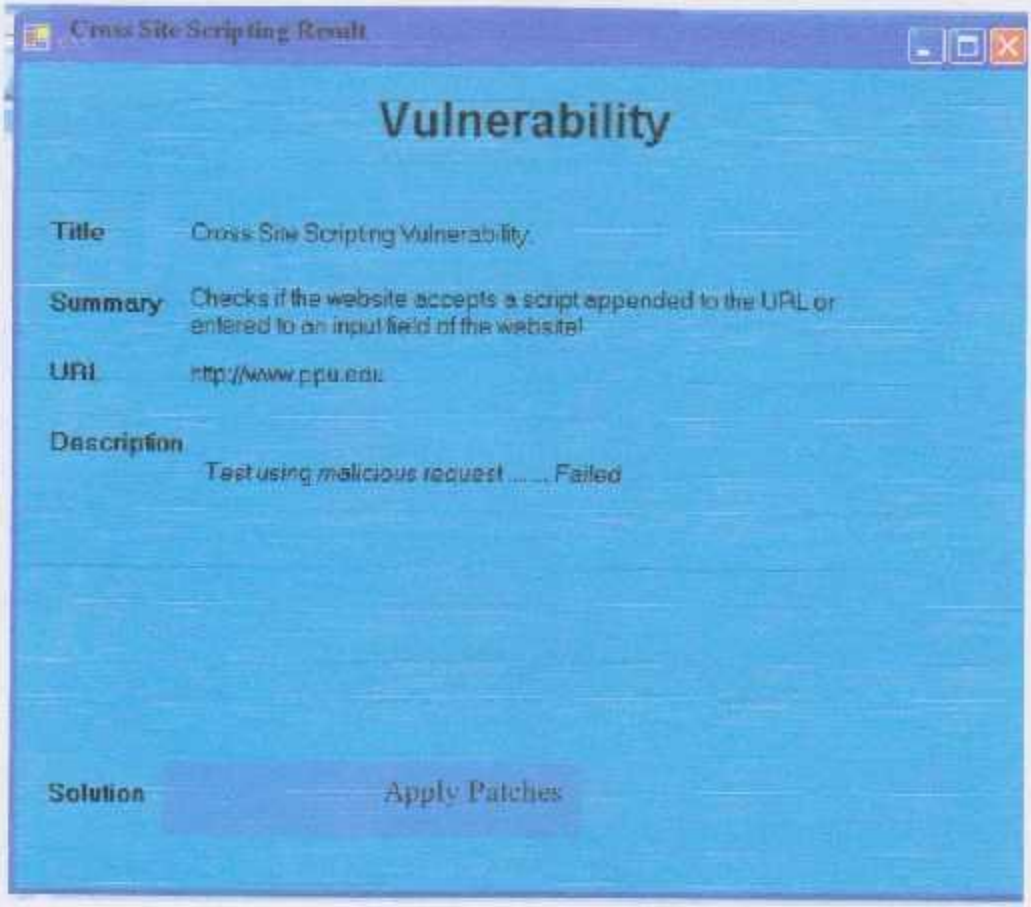


**Figure 6.3:** Cross Site Scripting Test

- **Command Execution Test on http://www.ppu.edu**

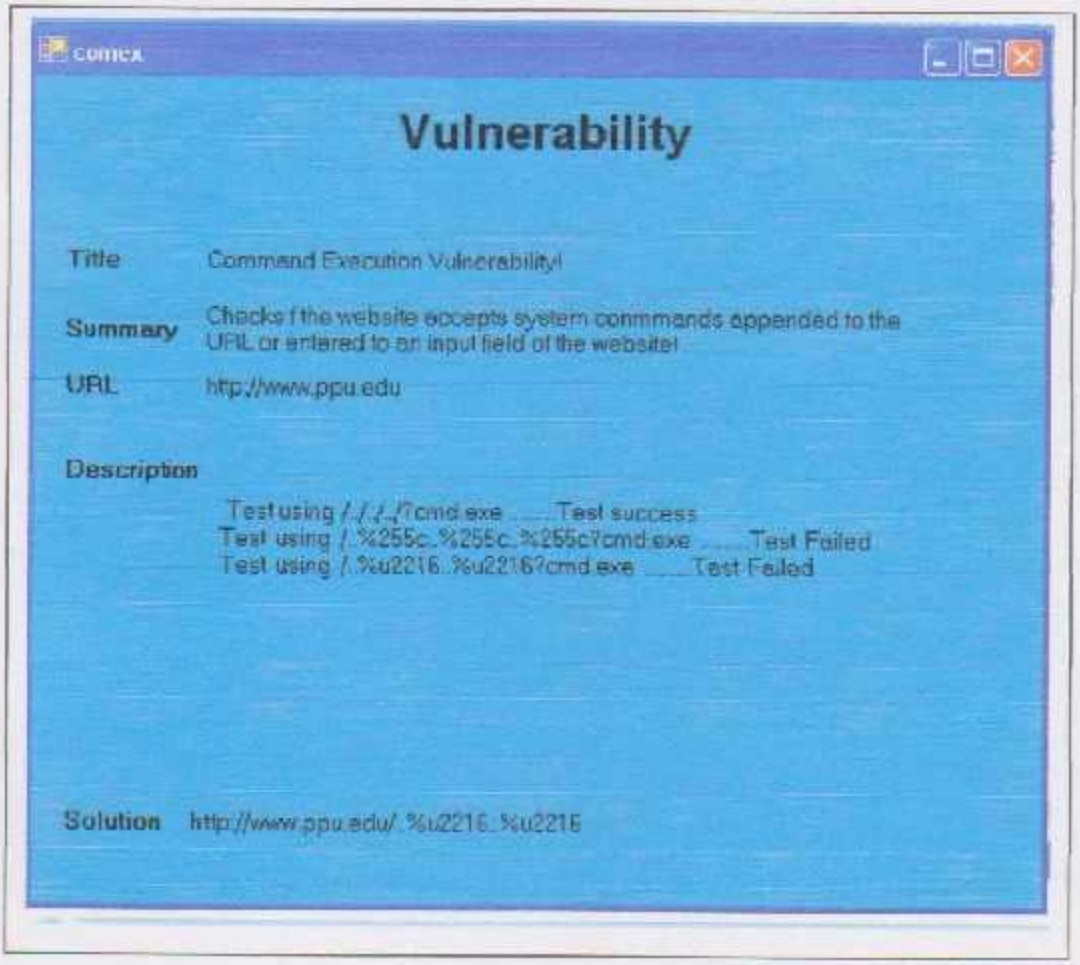


**Figure 6.4:** Command Execution Test

- **SSL Test on http://www.ppu.edu**

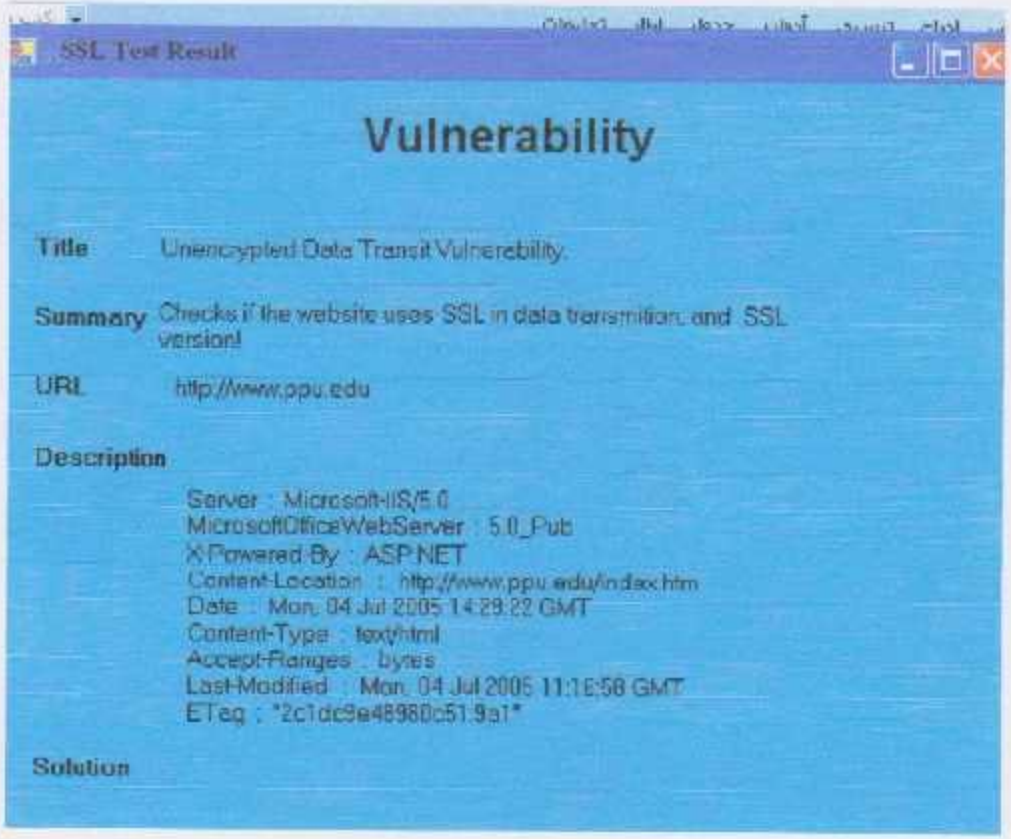


**Figure 6.5:** SSL Test

**2. Help Screen Part**

This part shows some help screens, after clicking the Question mark besides each test

| Screen Name | Test Result |
|---|---|
| Buffer Overflow Help | Done successfully |
| Information Leakage Help | Done successfully |
| Cookies Help | Done successfully |
| Insufficient Authentication Help | Done successfully |
| Cross Site Scripting Help | Done successfully |
| Command Execution Help | Done successfully |
| SSL Help | Done successfully |

The following are some web samples before and after testing.

- **Buffer Overflow Help Screen**

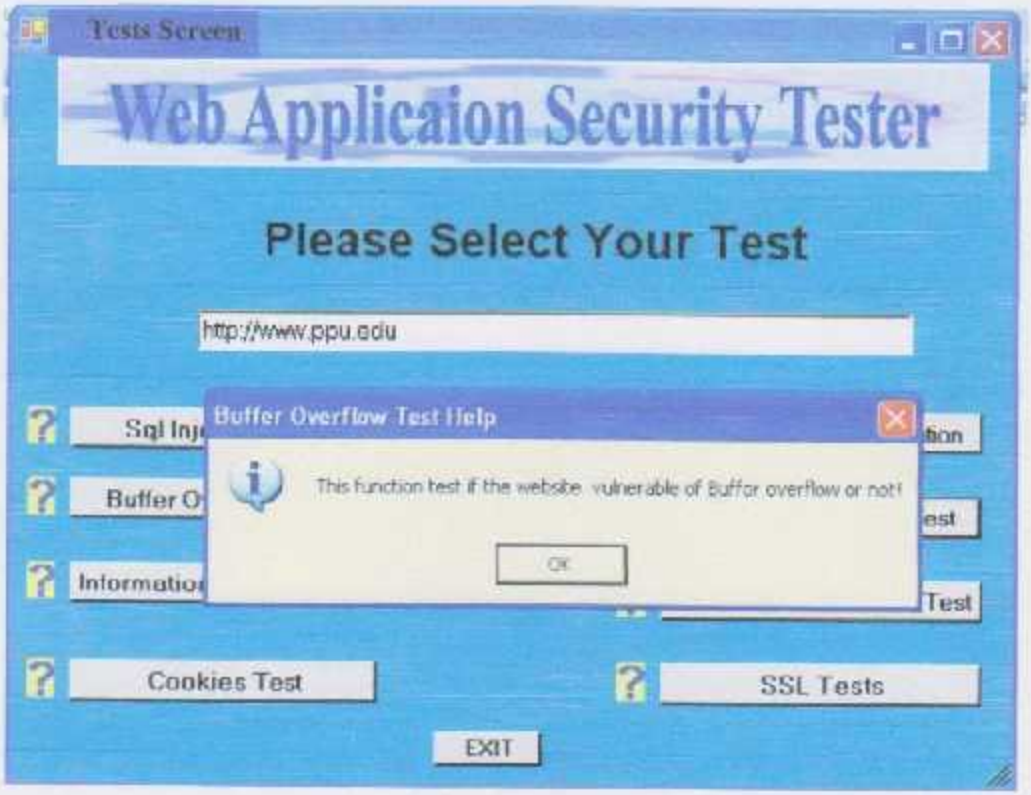


**Figure 6.6:** Buffer Overflow Help

- **Information Leakage Help Screen**

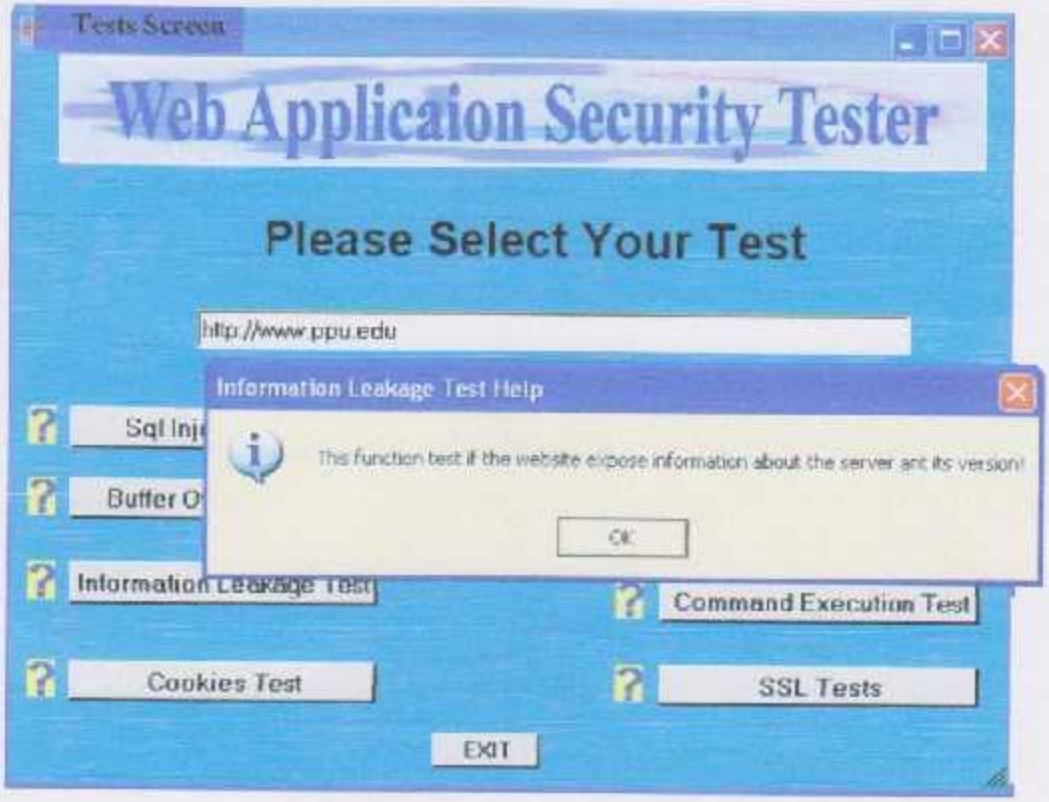


**Figure 6.7:** Information Leakage Help

- **Cookies Help Screen**


**Figure 6.8:** Cookies Help

- **Insufficient Authentication Help Screen**


**Figure 6.9**: Insufficient Authentication Help

- **Cross Site Scripting Help Screen**


**Figure 6.10:** Cross Site Scripting Help

- **Command Execution Help Screen**


**Figure 6.11:** Command Execution Help

- **SSL Help Screen**

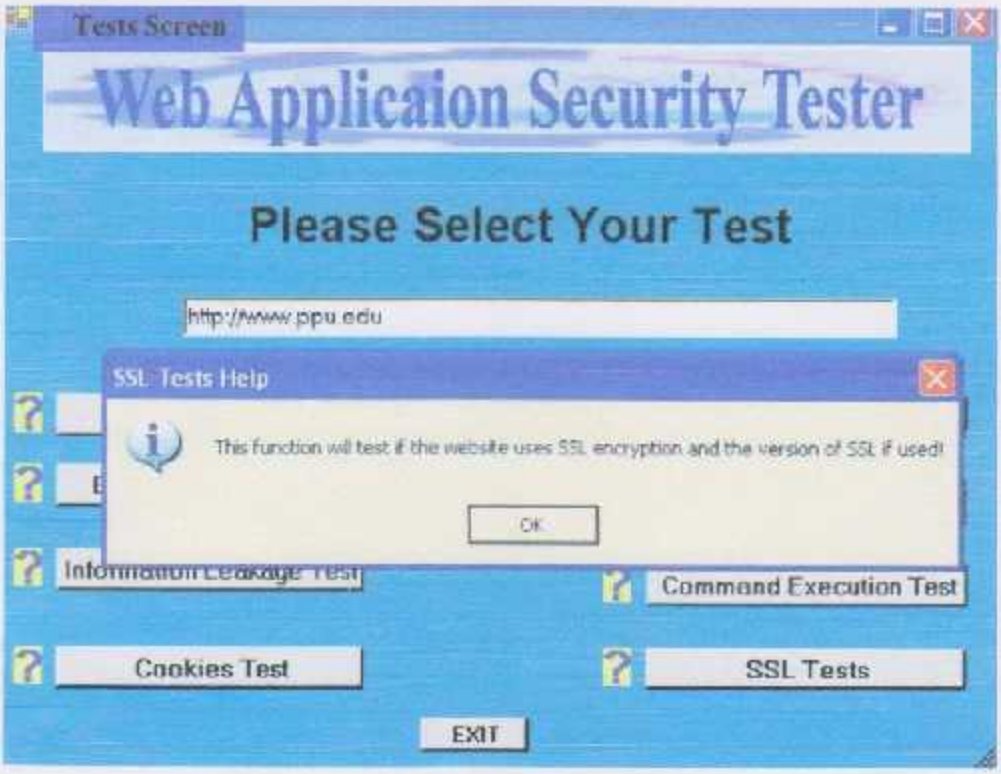


**Figure 6.12:** *SSL Help*

### 6.2.2 System Integration Testing

In (VB.NET) screens part each screen has been built and tested separately, and when it integrated with other screens in the whole application to ensure it is operated successfully.

Similarly, each screen in the Help screens part has been tested separately and then integrated with the complete application to verify its operation.
The integration of all objects was tested to ensure that the whole system performs as expected.

## 6.3 Testing Plan Results

The result of each screen performs as expected when tested separately, also the whole screens operate as expected when the application operated as a unit and process the developer needs.

The forms of each part perform as expected when each one tested separately. They give its results correctly when operated as an integrated environment.

The testing of the system integration indicated that the system performs as expected.

## 6.4 Summary and Recommendations

- Each operation is tested separately to ensure that it operates as expected.
- The integration of all objects is tested to ensure that the whole system performs as expected.
- The testing results indicate that the system works correctly.
- The results before and after testing show the process for the user and in the database system.

- The system operation as a unit ensures that the whole WEB APPLICATION SECURITY TESTER system performs as expected to be.

All system testing requirements are discussed in details with the supervisor. The supervisor gave his permission to pursue work for Maintenance.

# 7

# *Chapter Seven*

# *Maintenance*

# Chapter Seven

# 7 Maintenance

## 7.1 Introduction

The errors and omissions in the design requirements are discovered during the operation time of the program. To handle these errors, the system builds the suitable functions for catching the errors and maintaining the drop points to succeed the program perfectly. The system must therefore evolve to remain useful under problem resolution, enhancements, and interface modifications.

This section aims to maintain the program from any identified change in order to keep its services done correctly.

## 7.2 Implementation Plan

To design a system, it is necessary to decide the situation of the system with other projects that may or may not have a relation with it .The relation determined wither to update a specific system or completely design a new one. This can be performed by three methods of planning, they are:

1. **Direct Plan**

Replacing the old system completely with the new one, or when there is no old system to be replaced.

2. **Pilot Plan**

Using parts of the new system and keeping parts of the old system and run them together for sometime.

**3. Parallel Plan**

Running the old system and the new system in parallel at the same time for sometime, this used to test and verify the new system.

The WEB APPLICATION SECURITY RESTER system used the direct method since it is a new system.

## 7.3 Establishment of Production Environment

The following steps show the prerequisites that needed for the operating environment to help the administrator configure and operate the system:

1. Microsoft Windows server family or XP.
2. Microsoft office Family.
3. Internet Information Service (IIS).
4. .NET Framework.
5. SQL Server 2000.

## 7.4 Migration and Deployment Plan

### 7.4.1 WEB APPLICATION SECURITY RESTER System Production

**1. ASP.NET Part Production**

To put the WEB APPLICATION SECURITY RESTER system in production, the following steps must be done:

1. Build the application (BIN directory and DLL file will be created).
2. Copy the following files to the production server (BIN directory, web forms(.aspx), user controls (.ascx), XML files (.asmx), web.config, global.asax and changes of machine.config ).

3. Create a virtual directory using IIS.
4. Register your domain name with one of the companies.
5. While site is up and running, you can replace (update) files with new versions.

**2. VB.NET Part Production**

All the code behind page implemented using VB.NET

**3. Database Production**

- In the SQL server Enterprise Manager, right click on the database name(WEB APPLICATION SECURITY RESTER) ,choose all task and then select Detach database
- The detached database will be stored in the following directory……..\Microsoft SQL Server\MSSQL\Data\ WEBAPPLICATIONSECURITYRESTER.
- To setup database on the production server ,right click on the databases on the SQL server ,choose attach database
- In the attach database dialog box browse for the location of the database, and press Ok.

### 7.4.2 WEB APPLICATION SECURITY RESTER system management

The system used only by developers; to prevent unethical use for this software. Developers have full control over the system, execute, change, and update….etc

### 7.4.3 Security guidelines

1- This software is not free software.
2- Using this software should be ethical.
3- Developers are not responsible for unethical use of this software.

### 7.4.4 System Updating

The possibility of changing (adding or removing) some of the program requirements may be appeared in some cases. If the required update needed within the state study of the team, the maintenance is the responsibility of the programmer's team. Otherwise, the update cost is the responsibility of the program owner (purchaser). When the purchaser needs to update a requirement or add new one, he should call the programmers team and tell what does the updating points he need.

The programmer then does the request and resends the update, and after any changing, the documentation must be updated, and unit testing, integrated testing and system testing must be done. Then display the new functionality to the purchaser to make the acceptance testing in which we ensure that what he wants.

### 7.4.5 Error handling

During the implementation of the system, if an error occurs, the error message and its description will be displayed on the screen, and then the client must deal with the message in a way telling the WEB APPLICATION SECURITY RESTER developers about the error.

After the vendor completes repairing the error, he should make unit testing, integrating and system testing to ensure that the system work in a correct way, without damaging other functions.

## 7.6 Conclusion and Recommendation

- A good general security principle is ''defense in depth"; you should have numerous defense mechanisms (''layers") in place, designed so that an attacker has to defeat multiple mechanisms to perform a successful attack.
- Doesn't trust user input and don't trust anything comes across the network.
- Don't Store secrets in code or config files.
- Encrypt secrets, and store them in a safe place.
- Use principle of least privilege: Access data with the lowest possible permission.
- Pay attention to sensitive information.
- Don't depend on inheriting secure defaults information.
- "All input is evil, until proven otherwise".
- Never make a decision based on the name of something.
- Validate for correctness, reject otherwise.
- Security remains a major roadblock to universal acceptance of the Web for many kinds of transactions, especially since the recent sharp increase in remotely exploitable vulnerabilities have been attributed to Web application bugs.
- Zero risk is not practical
- There are several ways to mitigate risk
- Don't spend a million bucks to protect a dime .
- There are three types of implementation plan: Direct, Pilot, Parallel.
- The system used the Direct Plan since it is a new system.
- The programmer must prepare the suitable environment to deal with the system.
- The administrator must keep database access as secure as possible.
- The designers put reports for handling the errors or updating the system if needed.

All system maintenance requirements are discussed in details with the supervisor. The supervisor gave his acceptance.

# References


1. John Alexander & Billy Hollis_ "Developing Web Applications with Visual Basic .NET and ASP.NET"_ John Wiley & Sons, Inc_ New York_2002

2. Tamara Dean_"Network+ Guide to Networks"_ Third Edition_ Course Technology_ Canada_2004

3. Microsoft Inc.: http://www.microsoft.com

4. "Brute Force Attack", Imperva Glossary:
   http://www.imperva.com/application_defense_center/glossary/bruteforce.html

5. "Dos and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Compute Science:
   http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf

6. "Session Fixation Vulnerability in Web-based Applications", By Mitja Kolsek - Acros Security:
   http://www.acrossecurity.com/papers/session_fixation.pdf

7. "A new spoof: all frames-based sites are vulnerable" – SecureXpert Labs:
   http://tbtf.com/archive/11-17-98.html#s02

8. "HTML Code Injection and Cross-site Scripting", By Gunter Ollmann: http://www.technicalinfo.net/papers/CSS.html

9. Web Application Security Consortium: http://www.weapsec.com

# Appendices

Appendix A

Appendix B

# Appendix A

Source Code

# Appendix B

## Web Security Glossary