

CHAPTER 1

INTRODUCTION

1.1 Domain Introduction

Database security concerns the use of a broad range of information security controls to protect databases. It is potentially protecting the data, the database applications or stored functions, the database systems, the database servers and the associated network links against compromises of their confidentiality, integrity and availability. Database security entails allowing or disallowing user actions on the database and the objects within it.

Oracle uses schemas and security domains to control access to data and to restrict the use of various database resources. It is Mechanism that protects the database against intentional or accidental threats. There is a need for a technical and procedural standard for the protection of database systems, which lies at the heart of information systems. Such a standard shall serve as a guide to the setting up and operation of; systems that provide and maintain a safe and secure environment. Database security allows or disallows user actions on the database and objects within it. It protects a database from unintended activity. Unintended activity can be categorized as authenticated misuse, malicious attacks or inadvertent mistakes made by authorized individuals.

Database security is a term that is used to describe the safety invested into a single database, or storage of numerous information's and confidential on a computer. Database security can be defined as a system or process by which the "Confidentiality, Integrity, and Availability," or CIA, of the database can be protected.

1.2 Problem Definition

A SQL injection attack involves the alteration of SQL statements that are used within a web application through the use of attacker-supplied data. Insufficient input validation and improper construction of SQL statements in web applications can expose them to SQL injection attacks. The hacker can retrieve the information of a user using various SQL injections such as tautology, piggybacked queries, union query, alternate encodings, malformed queries, inference, leveraging stored procedures, etc.

1.3 Project Description

The main objective is to keep database in a well secured manner under serious SQL injection attacks and to analyze the principle of SQL attacks, since it is considered to be a serious attack on a database. It provides safety methods to both users as well as administrators. It also avoids the administrators bypassing the user accounts. Various Illegal functions such as deleting records from the database, adding unwanted information to the database, running the innermost function in the database are also can be eradicated using various counter measures that were implemented in project.

CHAPTER 2

LITERATURE SURVEY

Numerous Scenarios of data leaks in the recent years have made the Clients to place their valuable data's in a third party provider without the assurance of providing the privacy and data confidentiality [17],[6]. Data privacy means the data owned by an individual will never be disclosed to anyone else. Whereas, the data confidentiality refers to the ability to share the sensitive data among a community of users [10],[5]. Outsourced database prototype that allows clients to execute SQL queries [2], with privacy and under the regulatory compliance of constraints by leveraging the server-hosted and tamper-proof trusted hardware in the critical stages of query processing [2]. This provides the safety of the database from the hackers (intruders, insiders and administrators)[3].privacy is always a bit easier to implement rather than implementing the data confidentiality[5]. It is a tough task to implement both the security attributes to the database. There occur several problems with the modern day database that the hackers can hack a database by Bypass login [13]. We have introduced the concept of placing the SCPU (secure co-processor)[6],outside the main database thus it doesn't have any limitation to the memory space of the database along with which the Concept of paging module is implemented for the ease of the database query parsing[3].

Hackers mainly target the code of the database where the username or the password of an user can be easily hacked this can degrade the database security and it can be rectified by saving the username and password not as such instead they are stored in the converted manner[9]. At the position of providing the information to a database there is a possibility of modifying the database or deleting its data's by code injection. This Possibility of modifying the database by

SQL code injection can also be avoided by using the Bind variable method [13]. Whereas, by the usage of DBMS-Assert, Each Query Is Processed Word By Word Which Avoids Running Various Function. Recent problems with the administrator viewing the data and compressing the privacy can be rectified by using the concept of wrapping [15]. If the data in the SQL database is encrypted, and the encryption input is session anywhere else, that's a form of distributed security. Part of this issue is that SQL injection attacks [8]. A secure co-processor is a dedicated chip which at earlier stages was implemented as a slave along with a processor, now a day they are used as a single processor that's acts as a master of This SCPU [6].

There are several types attackers involved in attacking the database which damages the integrity of the database [11] the MD5 function has slightly decelerated in contrast to the MD4 function [1]. This raw data can be input to a computer program and also used in manual procedures such as analyzing the statistics from a survey [17]. The PL/SQL Wrapper process converts n PL/SQL source code into an intermediate form of the object code. Thus by hiding the application internals, this Wrapper prevents the Misuse of the application by other developers. Exposures of the algorithms to business competitors are also avoided. Wrapped code is as portable as source code [7]. The attributes in the database are always classified into public and private [10], where the public attributes are kept for the user purpose and the private attributes are encrypted and they can be decrypted only by the SCPU [9]. MD-5 algorithm is used for this purpose and this provides the output where the secondary phase of encryption is done at this position where this output is then converted into an alphanumeric value by using the raw data conversion methodology [9][13].

Following the authentication the users enters the main page where there is a possibility of SQL Code injection or function call injection can be performed. But this can be avoided by using the bind variable method where incase of code injection it will return a statement of invalid approach. Even though if they are a bit difficult to break this protection in case of exceptions this is overcome by using the concept of DBMS Assert[18], which provides and check up for the words using a word by word approach. Thus in case of finding any SQL queries in this, it will reply with an invalid statement. It is very much difficult to attack this database as we have the data wrapped [15]. Bind variables are one of those Oracle Concepts that experts frequently cite as being key to Application Performance. Once a query is submitted the Oracle first checks in the shared pool to see whether the statement has been submitted before. If already executed the execution plan is retrieved and the SQL is executed [11],[14]. The MD5 algorithm is obviously an extension of MD4, where the critical review is found to be fast,[6] but not possibly they are absolutely secure. In comparison with others, MD5 is not quite as fast as the MD4 algorithm, but they offer much more assurance of data security [2]. They are less power consuming and here the multiple cards can be used in parallel where their hardware chips provide fast hardware implementation of the algorithms [3][8]. The paging module is used for the process of navigation of the page from the home page.

This paging module is used to drag the pages from the database once the query is provided. Thus this enables us to place the SCPU outside the database and this also causes us to increase the data confidentiality to the data [1][10]. If the data in the SQL database is encrypted, and the encryption input is session anywhere else, that's a form of distributed security. Part of this issue is that SQL injection attacks [12][17]. MD5 is a hash function which was developed by Rivest [12]; it

generates a 128 bit long message and followed by a hash value. It is based on the principle of Merkle-Damgard construction. This MD5 is not, like the MD-4 which is the forerunner of MD5,[4] which is an iterating three-round hash function, but this has been expanded by a round among all other things. Thereby it said that the MD5 function has slightly decelerated in contrast to the MD4 function [13]. MD5 is currently a standard for conversion, Internet Engineering Task Force (IETF) Request for Comments (RFC)[11],[17]. This raw data can be input to a computer program and also used in manual procedures such as analyzing the statistics from a survey [1]. The PL/SQL Wrapper process converts n PL/SQL source code into an intermediate form of the object code. Thus by hiding the application internals, this Wrapper prevents the Misuse of the application by other developers [6][17]. . Raw data which has undergone processing are referred to as "cooked" data. Although these raw data has the potential to become "information," [4], [7]. The extraction, organization, and sometimes analysis and also the formatting for presentation are required for this to occur [11]. Paging module is used to pull the pages which are demanded by the SCPU for Query processing. Where Initially the DB is stored with data [1][4].

CHAPTER 3

SYSTEM STUDY

3.1 Existing System

Network security practitioners put more resources and efforts into defending against SQL INJECTION ATTACKS. But hackers will develop and deploy the next generation of SQLIA bonnets with different control architecture. A SQLIA is an attack that is aimed at subverting the original intent of the application. Through this hacker can easily enter into a user account and access their own information.

3.1.1 Limitations of Existing System

- Bypasses the login authentication.
- Selects secure information from database tables
- Attempts to add additional SQL statements to the existing SQL statement
- Execute oracle function or custom function from the select statement.
- It can easily execute oracle function or custom function from the select statement.
- If a DBA knows the password, he can easily access user account without user permission

3.2 Proposed System

The proposed system developed, eradicates the drawbacks of the existing system. The security level very much enhanced from its actual level. The DBA cannot view the user details in its original form.

The hacker cannot enter the user login by using the tricky queries, cannot run the inbuilt function in it etc. Usage of secure co-processor (SCPU) provides way for the extended safety and security of the data.

3.2.1 Advantages of Proposed System

- Avoid unauthorized access to the application
- User cannot get secure information from database
- User cannot add SQL queries to the existing SQL statement
- User cannot call oracle function or custom function.

3.3 Feasibility Study

The objective of feasibility study is not only to solve the problem but also to acquire a sense of its scope. During the study, the problem definition was crystallized and aspects of the problem to be included in the system are determined. Consequently benefits are estimated with greater accuracy at this stage. The key considerations are:

- ✓ Economic feasibility
- ✓ Technical feasibility
- ✓ Operational feasibility

3.3.1 Economic Feasibility

Economic feasibility studies not only the cost of hardware, software is included but also the benefits in the form of reduced costs are considered here. This project, if installed will certainly be beneficial since there will be reduction in manual work and increase in the speed of work.

3.3.2 Technical Feasibility

Technical feasibility evaluates the hardware requirements, software technology, available personnel etc., as per the requirements it provides sufficient memory to hold and process.

3.3.3 Operational Feasibility

This is the most important step of the feasibility study this study helps us predict the operational ability of the system that is being developed. This study also helps us analyze the approach towards which the system must be developed by which development effort is reduced. Proposed system is beneficial only if they can be turned into information systems, That will meet the organization requirements. This system supports in producing good results and reduces manual work. Only by spending time to evaluate the feasibility, do we reduce the chances from extreme embarrassments at larger stager of the project. Effort spend on a feasibility analysis that results in the cancellation of a proposed project is not a wasted effort.

CHAPTER 4

SYSTEM DESIGN

4.1 Architecture Diagram

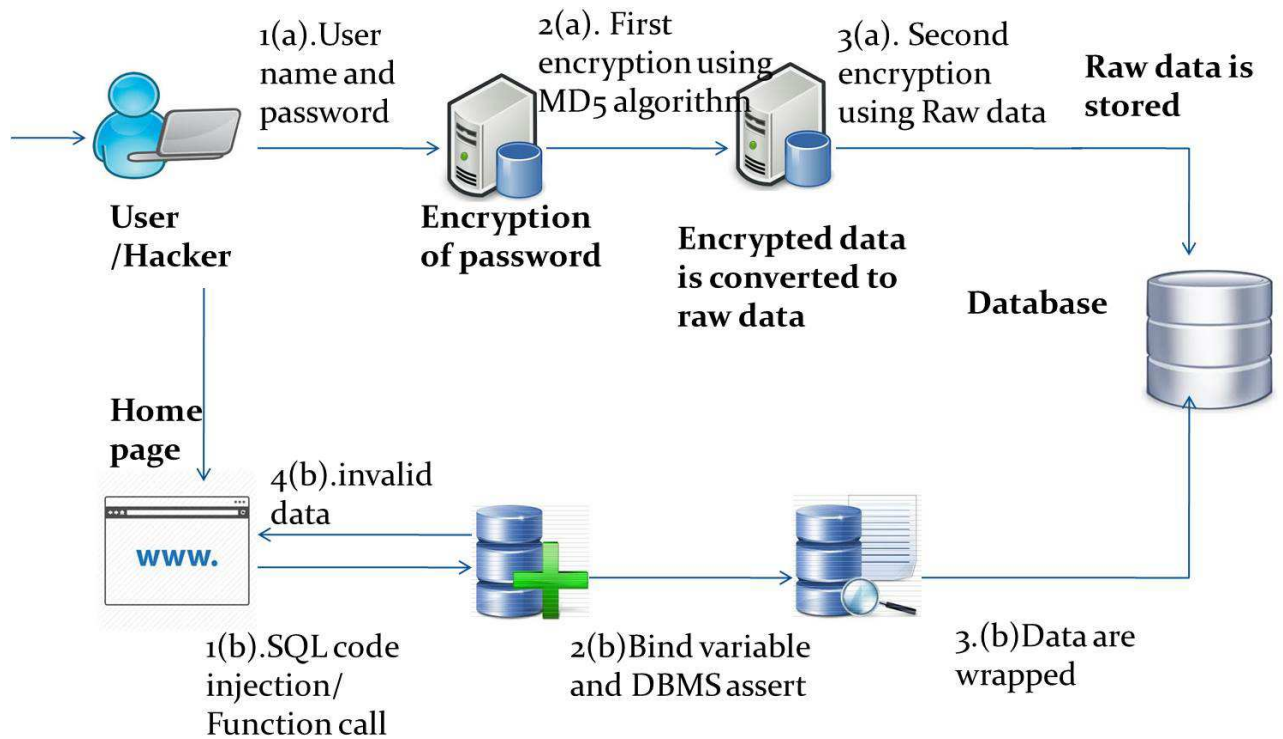
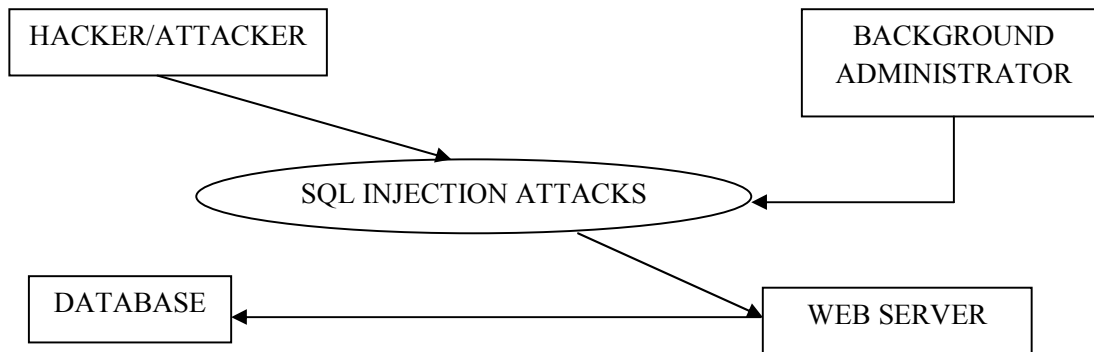


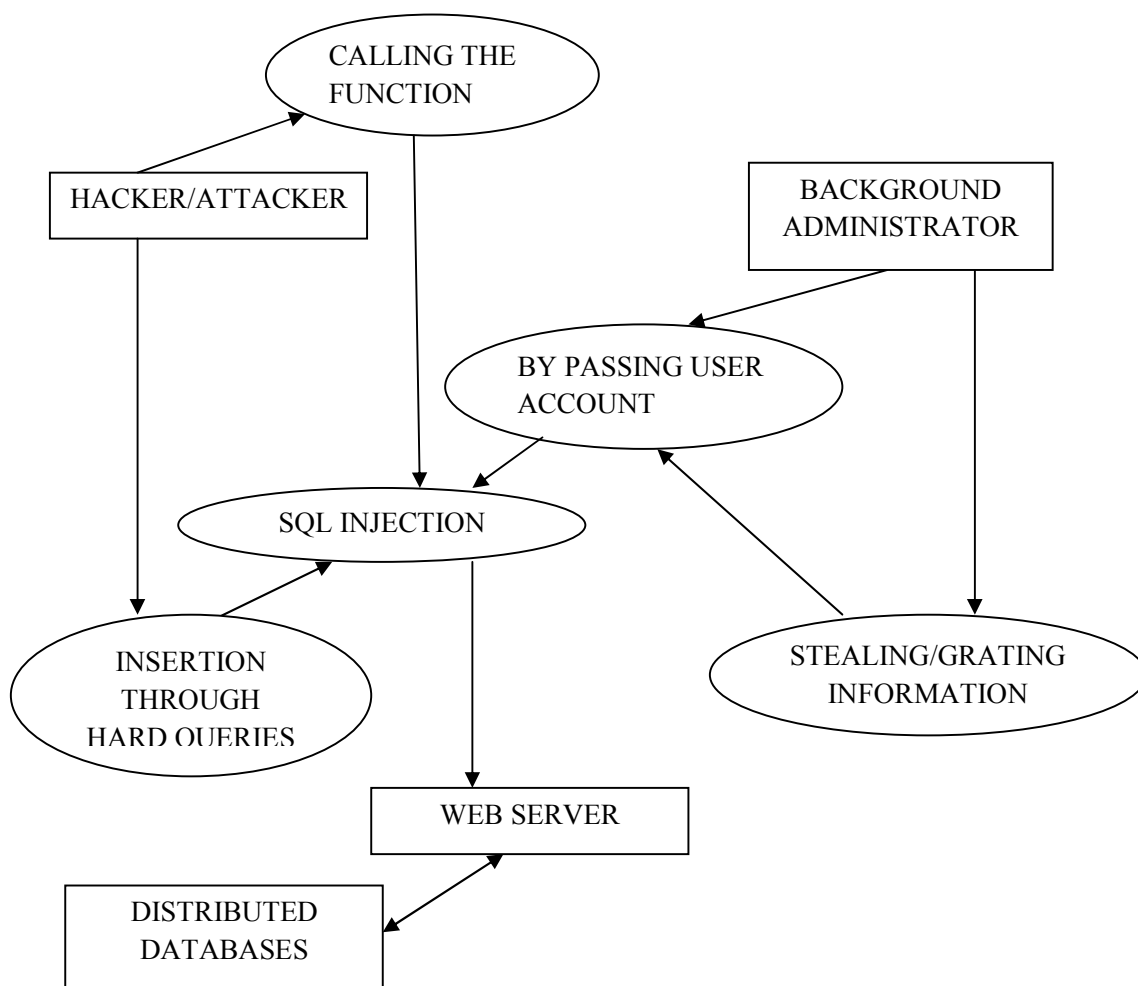
FIG: 4.1 Architecture Diagram for SQL Attack Free

4.2 Dataflow Diagram

4.2.1 Level 0



4.2.2 Level 1



4.2.3 Level 2

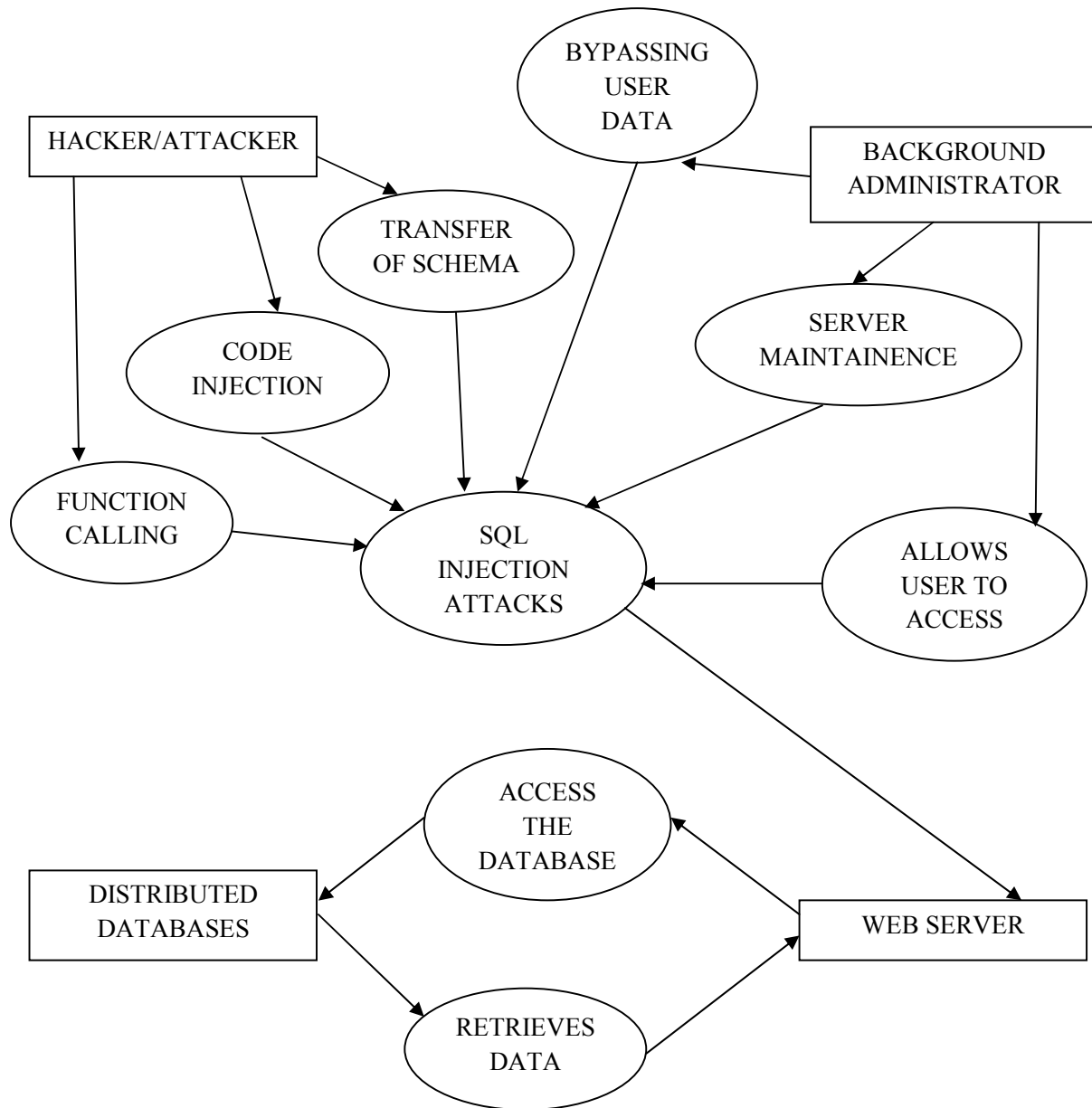


FIG: 4.2 Data Flow Diagram for SQL Attack Free

4.3 UML Diagrams

4.3.1 Use Case Diagram

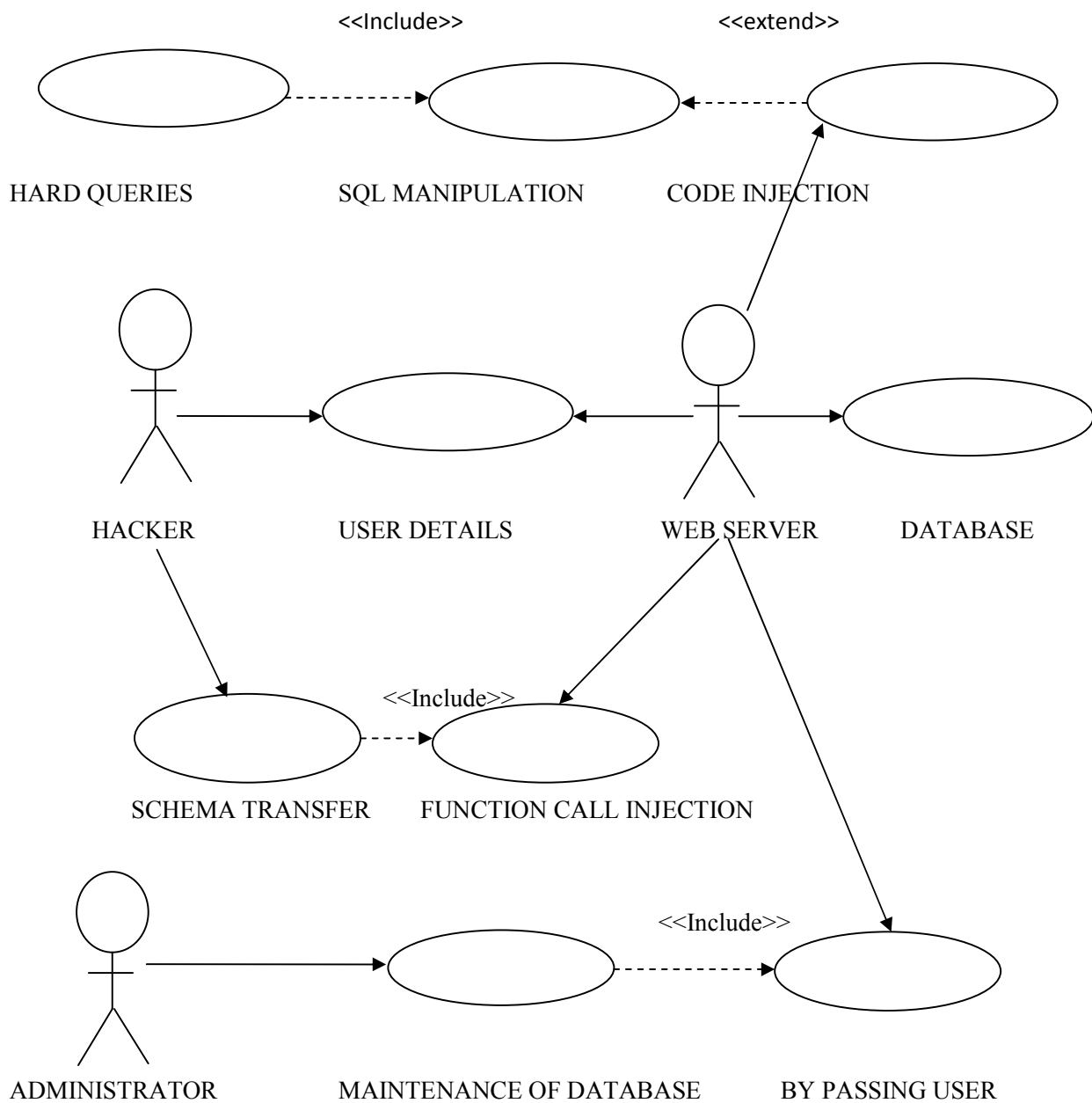


FIG: 4.3.1 Use Case Diagram for SQL Attack Free

4.3.2 Sequence Diagram

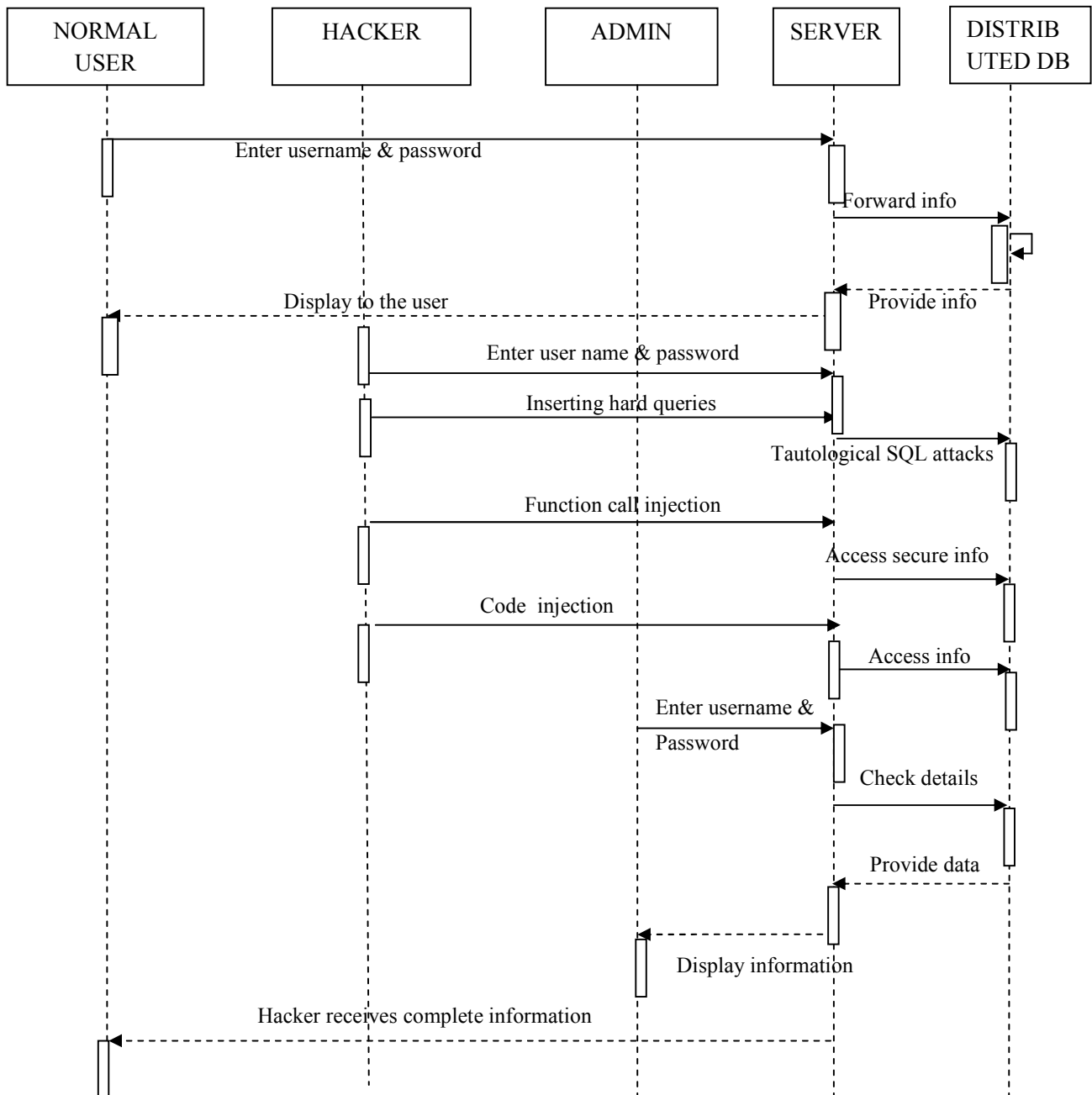


FIG: 4.3.2 Sequence Diagram for SQL Attack Free

4.3.3 Collaboration Diagram

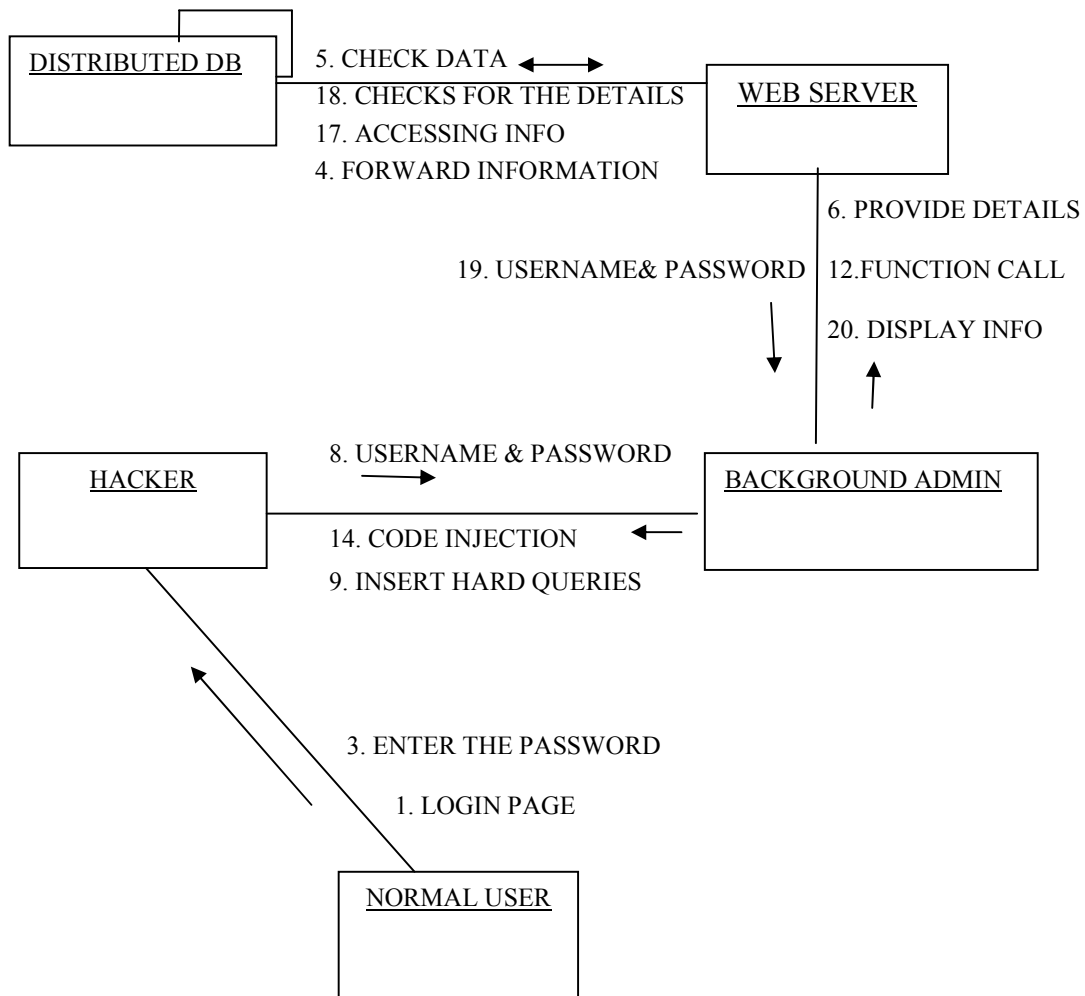


FIG: 4.3.3 Collaboration Diagram for SQL Attack Free

4.3.4 Activity Diagram

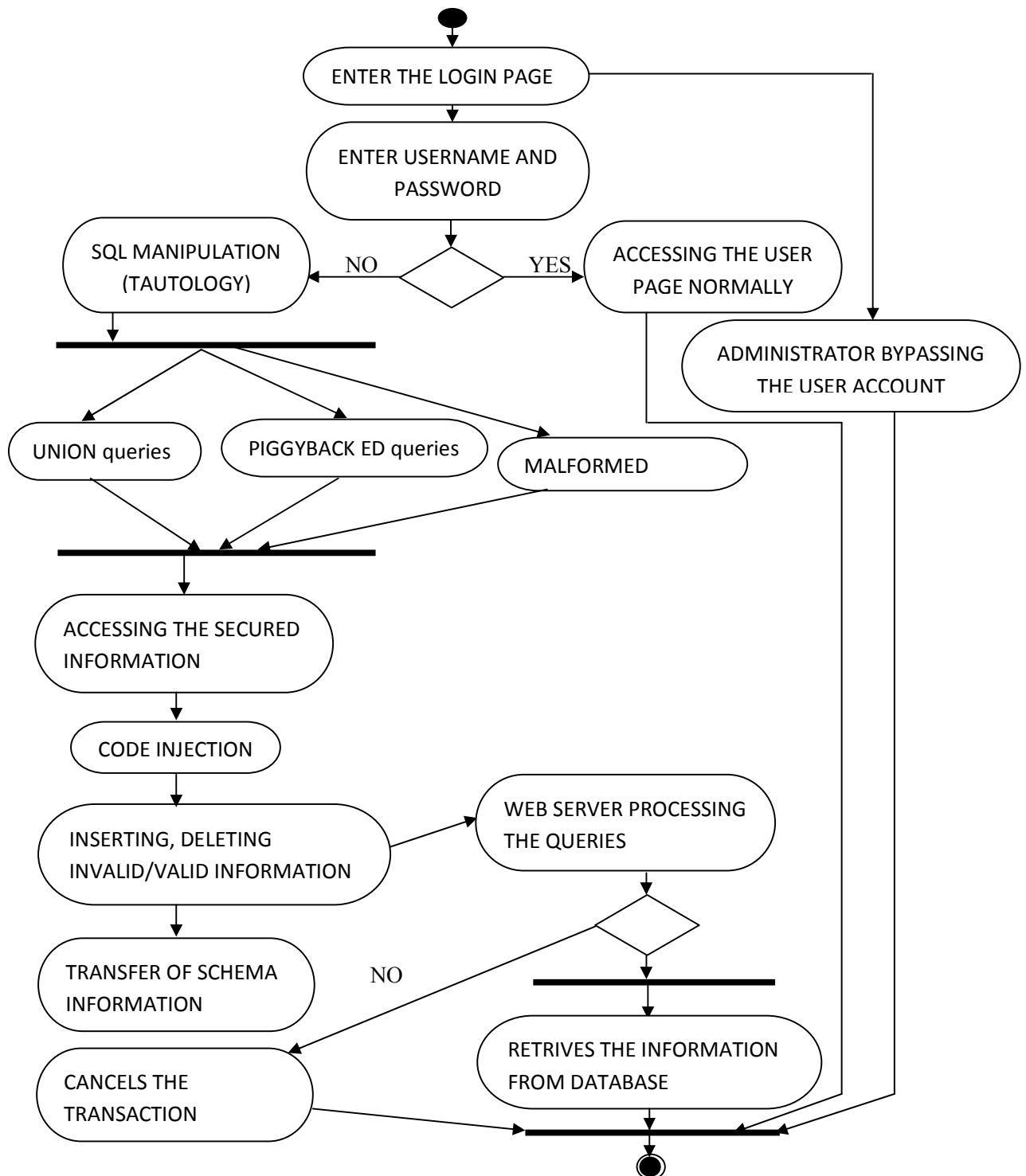


FIG: 4.3.4 Activity Diagram for SQL Attack Free

4.3.5 Class Diagram

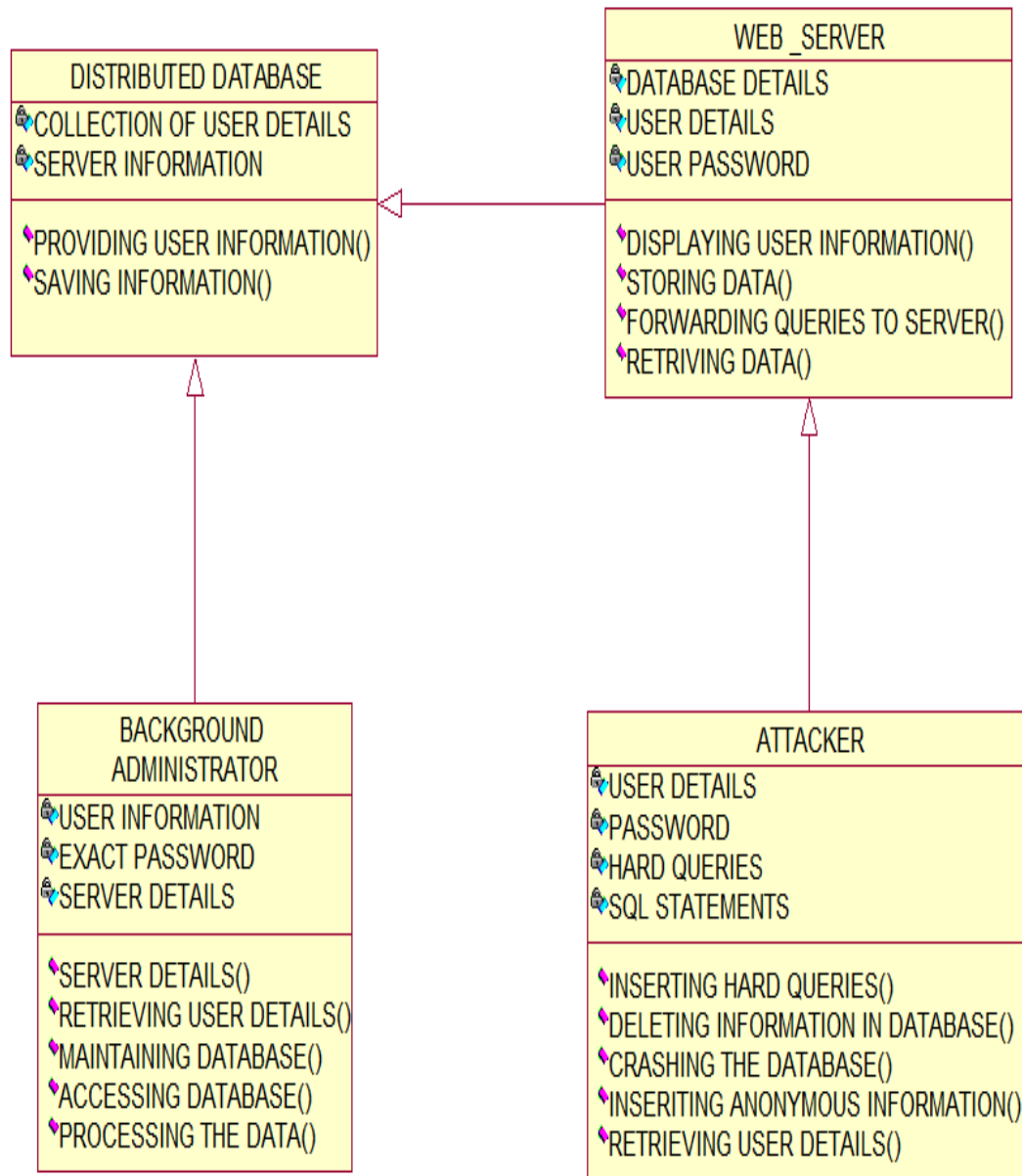


FIG: 4.3.5 Class Diagram for SQL Attack Free

4.4 Database Table Design

Table 1: emp_tab

S.No	USER_ID	USER_NAME
1	318071	Aarthy
2	318072	Vikram
3	318073	Meena
4	318074	Azees
5	318075	Michael

Table 2: product_master_tab

S.No	PRODUCT_NAME	SUBTYPE1	PROD_MODEL
1	Computer	Desktop	D1171
2	Computer	Laptop	L90
3	Mobile	Smart phone	MOB32
4	Monitor	LED	MON098
5	Mouse	Dual Key	MOU567

Table 3: scm_cfp

S.No	CFP_ID	ORDER_ID	SUPPLIER_ID	PRODUCT	QUANTITY	STATUS
1	C150	O79	SUPP1	MONITOR	1	WAITING
2	C151	O80	SUPP3	MOUSE	1	SENDING
3	C152	O81	SUPP6	TV	1	WAITING
4	C153	O82	SUPP3	PRINTER	1	ACCEPT
5	C154	O83	SUPP9	LAPTOP	1	ACCEPT

Table 4: scm_order

O_ID	CUST_ID	PROD	PROD_ TYPE	QUANT	DEST	STATUS	DATE
O71	C01	MONITOR	LED	1	CHENNAI	ACCEPT	3/1/15
O72	C09	MOUSE	DUALKEY	1	TRICHY	SEND	7/2/15
O73	C71	TV	LCD	1	CHENNAI	WAIT	2/3/15
O75	C11	LAPTOP	DELL	1	PONDY	ACCEPT	4/3/15
O76	C14	PRINTER	INKJET	1	MADURA	SEND	29/1/15

Table 5: user_tab

S.No	USER_ID	USER_NAME	USER_PW D	USER_ROLE
1	318071	Aarthy	Aarthy	Customer
2	318072	Vikram	D68005CCF 362B82D084 551B629179 2A3	Logistics
3	318073	Meena	Meena	Manager
4	318074	Azees	Azees	Customer
5	318075	Michael	A984937D1 88C71E30A B5E3ACDA EB6A25	Customer

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 Introduction

The requirements specification is a technical specification of requirements for the software products. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. .

5.2 Requirement Analysis

5.2.1 Hardware Requirements

- Processor : Pentium IV 2.4 GHZ
- Ram : 512 Mb
- Hard disk : 40 GB
- Disk Space : 1 GB

5.2.2 Software Requirements

- Operating System : Windows XP
- Language : Oracle
- Tool Kit : PL/SOL Developer
- Front end : Mod PL/SQL
- Back end : Oracle 10g

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Software Description

In this phase we need to analyze the availability of the resources that are required to design, develop, Implement and Test the project. The resources to be analyzed are Manpower, Time and the system Requirements. Teams of two members are involved in the entire SDLC life cycle except the testing phase. The testing phase is guided by the professional testers before the implementation of the product. Time Analyzed to complete the project is approximately four months with 4 hrs.on daily basis except weekends. System requirements are analyzed and listed below.

6.1.1 Oracle 10g

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads,

because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

6.1.1.1 Storage

The Oracle RDBMS stores data logically in the form of table spaces and physically in the form of data files ("data files"). Table spaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. Data blocks form the basic units of data storage.

A DBA can impose maximum quotas on storage per user within each table space.

6.1.1.2 Partitioning

Newer versions of the database can also include a partitioning feature: this allows the partitioning of tables based on different set of keys. Specific partitions can then be easily added or dropped to help manage large data sets.

6.1.1.3 Monitoring

Oracle database management tracks its computer data storage with the help of information stored in the SYSTEM table space. The SYSTEM table space contains the data dictionary—and often (by default) indexes and clusters. A data dictionary consists of a special collection of tables that contains information about all user-objects in the database. Since version 8i, the Oracle RDBMS also supports

"locally managed" table spaces which can store space management information in bitmaps in their own headers rather than in the SYSTEM table space (as happens with the default "dictionary-managed" table spaces). Version 10g and later introduced the SYSAUX table space which contains some of the tables formerly stored in the SYSTEM table space, along with objects for other tools such as OEM which previously required its own tablespace.

6.1.1.4 Disk files

Disk files primarily represent one of the following structures:

- Data and index files: These files provide the physical storage of data, which can consist of the data-dictionary data (associated to the tablespace SYSTEM), user data, or index data. These files can be managed manually or managed by Oracle itself ("Oracle-managed files"). Note that a datafile has to belong to exactly one tablespace, whereas a tablespace can consist of multiple datafiles.
- Redo log files, consisting of all changes to the database, used to recover from an instance failure. Note that often a database will store these files multiple times, for extra security in case of disk failure. The identical redo log files are said to belong to the same group.
- Undo files: These special datafiles, which can only contain undo information, aid in recovery, rollbacks, and read-consistency.
- Archive log files: These files, copies of the redo log files, are usually stored at different locations. They are necessary (for example) when applying changes to a standby database, or when performing recovery after a media failure. It is possible to archive to multiple locations.

- Temp files: These special data files serve exclusively for temporary storage data (used for example for large sorts or for global temporary tables)
- Control file, necessary for database startup. "A binary file that records the physical structure of a database and contains the names and locations of redo log files, the time stamp of the database creation, the current log sequence number, checkpoint information, and so on."

Data files can occupy pre-allocated space in the file system of a computer server, utilize raw disk directly, or exist within ASM logical volumes.

6.1.2 SQL

Structured Query Language is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standards (ISO) in 1987.

Since then, the standard has been enhanced several times with added features. Despite these standards, code is not completely portable among different database systems, which can lead to vendor lock-in. The different makers do not perfectly adhere to the standard, for instance by adding extensions, and the standard itself is sometimes ambiguous.

The SQL language is subdivided into several language elements, including:

- Clauses, which are constituent components of statements and queries. (In some cases, these are optional.)
- Expressions, which can produce either scalar values, or tables consisting of columns and rows of data.
- Predicates, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- Queries, which retrieve the data based on specific criteria. This is an important element of *SQL*.
- Statements, which may have a persistent effect on schemata and data, or which may control transactions, program flow, connections, sessions, or diagnostics.

Insignificant whitespace is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

6.2 Module Description

The system is sub divided into various modules. Each module has some certain set of operations to be performed. The various modules used are listed as follows:

- Bypassing login
- SQL Manipulation
- Function call Injection
- Accessing secured Information
- Transfer of schema information

The sub processes that are carried out by these modules can be categorized as follows

Bypassing login involves the conversion of username and password into raw data using MD5 algorithm.

SQL Manipulation involves the process of Bind variable concepts and Function call Injection uses the concept of DBMS_Assert. Accessing secured Information is done by using the concept of wrapping the database.

Transfer of schema information is provided by using the two database concept. This provides the security for information.

6.2.1 Bypassing Login

Tautology-based attacks are among the simplest and best known types of SQLIAs. The general goal of a tautology based attack is to inject SQL tokens that cause the queries

Conditional statements which always evaluate to true. Although the results of this type of attack are application specific, the most common uses are bypassing authentication pages and extracting data. In this type of injection, an attacker exploits a vulnerable input field that is used in the queries WHERE conditional. This conditional logic is evaluated as the database scans each row in the table. If the conditional represents a tautology, the database matches and returns all of the rows in the table as opposed to matching only one row, as it would normally do in the absence of injection. An example of a tautology-based SQLIA for the example in Section 2 is the following:

```
SELECT acct FROM users WHERE login='' OR 1=1 -- ' AND pin=
```

Because the WHERE clause is always true, this query will return account information for all of the users in the database.

Bypassing authentication is that attackers enter some special user name and password in the login dialog to log on the system with administrator privileges. Hackers they can login to admin profile or any profile with privileges without knowing username and password, every login process have condition in server side, they can inject the statement to satisfy the condition for both size.

6.2.2 SQL Manipulation

This attack occurred when developers don't filter content of SQL statements in the input dialog box. If attackers gain administrator privileges, attackers has basically controlled the information of the whole site, the harm is quite large for user's privacy and the website.

To avoid this problem we using BINDVARIABLE concept. Bind variables are one of those Oracle concepts that experts frequently cite as being key to application performance. Each time the query is submitted, Oracle first checks in the shared pool to see whether this statement has been submitted before. If it has, the execution plan that this statement previously used is retrieved, and the SQL is executed. If the statement cannot be found in the shared pool, it will not allow the SQL to execute.

The way to get Oracle to reuse the execution plans for these statements is to use bind variables. Bind variables are substitution variables that have the effect of sending exactly the same SQL to Oracle every time the query is executed. In our project, this concept to bind the values from the client size and compare to the database values.

Here the Input given is HARD TAUTOLOGICAL SQL QUERIES where then it takes the process to move on to the next step where it then Equates Each Query To Bind Variables and Then Query Is Checked For Its Condition once after the condition is checked it provides us the Output where The Tautological Based Attacks Are Avoided .Thus the concept of Bind Variable is used efficiently to avoid the SQL manipulation problems.

6.2.3 Function Call Injection

Making key operations of the database refers to the attacker insert a number of additional illegal SQL statements into the normal structure of the dynamic SQL statement, resulting in the server executes normal SQL statements that client sends, together with additional SQL statements which attackers construct. These additional SQL statements are often key operations to the database such as deleting the data table, modifying table fields, adding data, deleting data.

Although tautology-based attacks can be successful, for instance, in bypassing authentication pages, they do not give attackers much flexibility in retrieving specific information from a database. Union queries are a more sophisticated type of SQLIA that can be used by an attacker to achieve this goal, in that they cause otherwise legitimate queries to return additional data. In this type of SQLIA, attackers inject a statement of the form “UNION < injected query >.” By suitably defining < injected query >, attackers can retrieve information from a specified table.

```
SELECT acct FROM users WHERE login='' UNION SELECT cardNo  
from CreditCards where acctNo=7032 -- AND pin=
```

The original query should return the null set, and the injected query returns data from the “Credit Cards” table. In this case, the database returns field “card No” for account “7032.” The database takes the results of these two queries, unites them, and returns them to the application. In many applications, the effect of this attack would be that the value for “card No” is displayed with the account information.

Similar to union queries, this kind of attack appends additional queries to the original query string. If the attack is successful, the database receives and executes a query string that contains multiple distinct queries. The first query is generally the original legitimate query, whereas subsequent queries are the injected malicious queries. This type of attack can be especially harmful because attackers can use it to inject virtually any type of SQL command. In our example, an attacker could inject the text “0; drop table users” into the pin input field and have the application generate the following query:

```
SELECT acct FROM users WHERE login='doe' AND pin=0; drop  
table users
```

The database treats this query string as two queries separated by the query delimiter (“;”) and executes both. The second malicious query causes the database to drop the users table in the database, which would have the catastrophic consequence of deleting all user information. Other types of queries can be executed using this technique, such as the insertion of new users into the database or the execution of stored procedures. Note that many databases do not require a special character to separate distinct queries, so simply scanning for separators is not an effective way to prevent this attack technique.

To overcome this problem we propose the Oracle Package called DBMS_ASSERT using this package we avoid this Function Call Injection problem. To using this package we can ignore the fake actions, our system get the input from user text field and give it this DBMS_ASSERT package its read the input and if it's find any default SQL Keyword its stop the process or else its pass the input to server execution.

6.2.4 Accessing Secure Schema

Executing system commands of the database is to use mixing methods and construct special database objects to attack the database. There often exist extended stored procedures beginning with XP_ for the developers to call in the database. Attackers take advantage of this feature to call the system stored procedure in the SQL statement submitted to the database, in order to executing the database system commands. Stored procedures provide developers with an extra layer of abstraction because they can enforce business wide database rules, independent of the logic of individual Web applications. Unfortunately, it is a common misconception that the mere use of stored procedures protects an application from SQLIAs: Similarly to any other software, the safety of stored procedures depends on the way in which they are coded and on the use of adequate defensive coding practices. Therefore, parametric stored procedures could also be vulnerable to SQLIAs, just like the rest of the code in a Web application.

This type of attacks is giving the database schema information from the database and from this attack hackers get the information about the packages and function and procedures for our application from database.

To secure the database schema and source code from hackers we using the concept Wrapped. Its hide All the Information from hackers using Wrapped in a Same Package and Processed.

6.2.5 Transfer of Schema Information

A SQL Injection vulnerability in a function that executes with the privilege of the caller (defined with `AUTHID CURRENT_USER`) in an anonymous PL/SQL block is not useful for an attacker if it is used directly, but an attacker can use a vulnerability of this kind to:

Get around the need to create a function to inject and use this vulnerable function to inject the SQL statements. To do this the vulnerability must be in an anonymous PL/SQL block of an `AUTHID CURRENT_USER` function (in order to be able to define the transaction as autonomous).

Execute SQL statements in a web application vulnerable to SQL Injection even if the vulnerability is in a `SELECT` and no other statement is allowed to be added.

Basically have a possible to access the process from one schema to other schema, so users also can execute the admin process, and users can insert a new record to product table and they can delete a record from table or else can also drop the table from the database also. To avoid this problem we implement the `AUTHID CURRENT_USER` to give a privilege to all schemes.

CHAPTER 7

SYSTEM TESTING

7.1 Testing Objectives

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation.

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. A test case is one that has high probability of finding a yet undiscovered error. A successful test is one that uncovers as yet undiscovered error.

The basic types of testing are:

- White Box Testing
- Black Box Testing

7.2 Test Case Design

7.2.1 White Box Testing

This is a code testing strategy and it checks for the correctness of the every statement in the program. To follow this testing strategy, there should be cases that result in the execution of every instruction in the program or module. The test cases should make sure that independent paths within a module are executed at least once as and when required.

- Exercise all logical decision on their true or false side.
- Execute all loops at their boundaries and within their operational bounds.

The testing strategy, on the face of it, sounds exhaustive. If every statement in the program is checked for its validity, there doesn't seem to be much scope of error.

7.2.2 Black Box Testing

To perform Black Box Testing, the analyst examines the specification taking the program or module. It checks for the basic functionality and how it should perform on the various conditions when submitted for processing. By examining the result, the analyst can report whether the program performs according to the specified requirements.

7.2.3 Testing Strategy

Software testing consists of series of tests, which are implemented sequentially. These tests are:

- Test Plan
- Unit testing
- Integration testing
- System testing

7.2.3.1 Test Plan

Test plan specifies the objective of testing for completion criteria, system integration plan, and methods to be used on modules and particular test cases to be used. The four types of test that a software product must satisfy are:

- Function Test specifies operating conditions, input values and expected result.
- Performance Test verify response time, execution time and throughout primary and secondary memory utilization links.

- Stress Test is to determine the limitations of the system.
- Structural Test is concerned with examining the internal processing logic of a software system.

7.2.3.2 Unit Testing

Instead of testing the system as a whole, unit testing focuses on the modules that make up the system. Each module is taken up individually and testing for the correctness in the coding and logic. Errors resulting from interaction of modules are initially avoided.

The advantages of unit testing are:

- Size of a module is quite small and errors can easily be located.
- Confusing interaction of multiple errors in widely different parts of software eliminated.
- Module level testing can be exhaustive.

The obvious assumption made when unit test is pursued is that, individual modules can be isolated from the system module interacts with other modules in the system and, to isolate module, the analyst must stimulate these interactions.

7.2.3.3 Integration Testing

It tests for the errors resulting from integration modules. One specific target of integration testing is the interface: whether parameters match on both sides as to type, permissible ranges and meaning. Analyst tries to find areas where modules have been designed with different specifications.

This is the top-level testing. In this all modules tested separately would be put together and tested for producing the ultimate result of the system.

7.2.3.4 System Testing

The main objective of the system testing is to find out the discrepancies between the developed system and its original objective, current specifications and the system documentation.

7.2.3.5 Peak Load Testing

It is done to determine whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand.

7.2.3.6 Stress Testing

This is performed to determine the capacity of the system stored to transactions. Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.

7.2.3.7 Performance Time Testing

It is done to determine the length of time used by the system to process a transaction or user request.

7.2.3.8 Recovery Testing

It is done to determine the ability of the software to recover from failure. The recovery may be automatic or may need human intervention. In either case the mean time to repair is evaluated to determine whether it is within acceptable limits.

7.2.3.9 Procedure Testing

It determines the clarity of the documentation operation and use of system. Asking the user to do exactly what the manual requests to perform.

7.3 Test Case Results

Test	Requirement or Purpose	Action / Input	Expected Output	Actual Result	P / F
1.	Enter username, password and role	To click the register button	Password is encrypted	Same as encrypted	Pass
2.	Encrypted password	Preprocessor	Encrypted password should be converted to raw data	Same as encrypted	Pass
3.	Login with username and password	To click login button	Check with database	Same as encrypted	Pass
4.	Enter sql code in webpage text entry	To convert values to bind variable	Check with keyword	Same as encrypted	Pass
5.	Hard tautological query	Check with DBMS – assert package	DML keyword are not executed	Same as encrypted	Pass
6.	Wrapped database value	To store all records in database with hidden	Data are hidden using wrapping	Same as encrypted	Pass
7.	Database name	Compile all value in one object	Access from another object	Same as encrypted	Pass

CHAPTER 8

RESULT AND DISCUSSION

Thus we have proposed a system where we have enhanced the features of the existing system. We have eliminated the threat of hacking thereby providing high security features. Hackers like Insiders, Intruders and Attackers are prevented from hacking by avoiding them to access to the database. We have proposed the system mainly to avoid the drawbacks and to flaws and errors of the existing system. Our project mainly concentrates on online shopping we it might be done worldwide therefore having high chances of hacking.

This project prevents unauthorized access of hackers where the user cannot get the secure information. Here the user cannot add the SQL statements to the database or call any oracle function. Here the database administrator is also restricted from viewing the code. Thus we have created a trusted database where we provide privacy and data confidentiality at the same time. Thereby, avoiding users to place their data in a third party database where there are high chances of data leaks.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 Conclusion

The principles of SQL attacks and attack processes are analyzed. It introduces the visiting process based on client/server model. On this basis, a database protection system against SQL attacks, mainly including the protection for ordinary users and administrators is achieved. Experiments show that this is a very effective protection system. But there are also some lacks of the protection system, protective measures taken by this system can protect the common SQL attacks, but not prohibit some rare attacks. Therefore, more research in the SQL attacks based on the protection system should be done. Of course, new attack methods are emerged with the network's development; the system protection systems should also be continuously improved and perfected.

9.2 Future Enhancement

Though the protection of database in a small area of network has been achieved using these concepts, it should be achieved in a wide area of network. The SQL protection as well as recovery of the information should be achieved in an easier manner. The disk space required for storing the data should be partly reduced. The information storage capacity of a database system should be enhanced without leaking out necessary information of a user. The information should be wrapped in a secured manner so that no one can access it. Likewise this database protection system must be enhanced in the nearly future.

APPENDIX

Database Design

Table 1: emp_tab;

Name	Type	Nullable	Default	Comments
------	------	----------	---------	----------

USER_ID	NUMBER	Y		
---------	--------	---	--	--

USER_NAME	VARCHAR2(100)	Y		
-----------	---------------	---	--	--

Table 2: product_master_tab;

Name	Type	Nullable	Default	Comments
------	------	----------	---------	----------

PRODUCT_NAME	VARCHAR2(200)			
--------------	---------------	--	--	--

SUBTYPE1	VARCHAR2(200)			
----------	---------------	--	--	--

PROD_MODEL	VARCHAR2(200)			
------------	---------------	--	--	--

Table 3: scm_cfp;

Name	Type	Nullable	Default	Comments
------	------	----------	---------	----------

CFP_ID VARCHAR2(100) Y

ORDER_ID VARCHAR2(100) Y

SUPPLIER_ID VARCHAR2(100) Y

PRODUCT VARCHAR2(100) Y

QUANTITY NUMBER Y

STATUS VARCHAR2(50) Y

Table 4: scm_order;

Name	Type	Nullable	Default	Comments
------	------	----------	---------	----------

ORDER_ID	VARCHAR2(100)	Y		
CUSTOMER_ID	VARCHAR2(100)	Y		
PRODUCT	VARCHAR2(100)	Y		
PROD_TYPE	VARCHAR2(100)	Y		
PNAME	VARCHAR2(100)	Y		
QUNATITY	NUMBER	Y		
DESTINATION	VARCHAR2(100)	Y		
STATUS	VARCHAR2(100)	Y		
MOD_DATE	DATE	Y	sysdate	

Table 5: user_tab;

Name	Type	Nullable	Default	Comments
------	------	----------	---------	----------

USER_ID	NUMBER	Y		
USER_NAME	VARCHAR2(100)	Y		
USER_PWD	VARCHAR2(100)	Y		
USER_ROLE	VARCHAR2(100)	Y		

Source Code

SQLAttackFree_UserDetails

Create or replace package body ELOG_USERDETAILS is

```
procedure signup(  
    usernameuser_tab.USER_NAME%type  
    ,password1user_tab.USER_PWD%type  
    ,role1user_tab.USER_ROLE%type  
)  
    // inserting table values  
(  
    USER_NAME,
```

```

USER_PWD,

USER_ROLE

)

Values(

username,

password1,

role1

);

commit;

//signup

function isValidUser(

username varchar2

,password1 varchar2

)

return boolean

is

u_pwd USER_TAB.USER_PWD%type;

user_name USER_TAB.USER_name%type;

sql1 varchar2(5000);

```

begin

SQL1:='select user_name from USER_TAB where USER_ID=:a and
user_pwd=:b';

//execute immediate will occur

functiongetUserRole(userid varchar2,pasword varchar2) return varchar2

is

u_roleUSER_TAB.USER_ROLE%type;

begin

htp.p('***getUser'); -- @@@@

selectUSER_ROLE,user_name into u_role,gv_u_name from USER_TAB where
USER_ID=userid;

--debug_aarthy_tab(gv_u_name ||'user');

htp.p('***USER ID : '||userid); -- @@@@

//exception detected

procedureinsertOrder(order_idSCM_ORDER.ORDER_ID%type default
to_char(1),

customer_idSCM_ORDER.CUSTOMER_ID%type,

productSCM_ORDER.PRODUCT%type,

prod_typeSCM_ORDER.PROD_TYPE%type,

pnameSCM_ORDER.PNAME%type,

```

quantitySCM_ORDER.QUNATITY%type,

destination SCM_ORDER. DESTINATION%type,

status SCM_ORDER. STATUS%type) is

    //inserting order

procedureVerifyUser(username USER_TAB.USER_ID%type ,password1
USER_TAB.USER_PWD%type,click varchar2 default 'login') is

u_roleUSER_TAB.USER_ROLE%type;

sql1 varchar2(5000);

begin

http.p('Hello');

if(isValidUser(username,password1)) then

http.p('USER ID : '||username||' <br> Password: '||password1); -- @@@@

u_role:=getUserRole(username,password1);

case when u_role = 'manager' then

ELOG_WEBPAGE.manager;

when u_role = 'customer' then

ELOG_WEBPAGE.customer(gv_u_name);

whenu_role = 'supplier' then

ELOG_WEBPAGE.supplier;

```

```

whenu_role = 'logistics' then

ELOG_WEBPAGE.logistics;

else

http.p('<html><body><font color="red">invalid password</font></body></html>
');

ELOG_WEBPAGE.loginPage;

end case;

else

http.p('<html><body><font color="red">invalid password</font></body></html>
');

ELOG_WEBPAGE.loginPage;

end if;

exception when others then

http.p('<html><body><font color="red">exeception</font></body></html> ');

ELOG_WEBPAGE.loginPage;

endVerifyUser;

proceduregetOrderDetails(customer_namescm_order.CUSTOMER_ID%type
default null ) is

begin

if (customer_name is null) then

```

```

select ORDER_ID,

        CUSTOMER_ID ,

        PRODUCT,

        PROD_TYPE,

        PNAME,

        QUNATITY,

        DESTINATION,

STATUS,mod_date

bulk collect into g_orderlist

fromscm_order

wheretrunc(mod_date) >trunc(sysdate) - 30;

else

select ORDER_ID,

        CUSTOMER_ID ,

        PRODUCT,

        PROD_TYPE,

        PNAME,

        QUNATITY,

        DESTINATION,

```


STATUS,mod_date

bulk collect into g_orderlist

fromscm_order

where CUSTOMER_ID=customer_name and trunc(mod_date) >trunc(sysdate) -
30;

end if;

endgetOrderDetails;

proceduregetCfpDetails is

begin

select * bulk collect into g_cfplist from scm_cfp;

endgetCfpDetails;

procedurePlaceOrder(product scm_order.product%TYPE,

 type1 scm_order.prod_type%TYPE

,pnamescm_order.pname%TYPE

,quantityscm_order.qunatity%TYPE

,destinationscm_order.destination%TYPE

,customer_name varchar2)is

order_idvarchar2(200);

cfp_idvarchar2(200);

```

--customer_idVARCHAR2(100);

Supplier_idVARCHAR2(100);

status varchar2(100);

dest varchar2(300);

begin

    --customer_id := 'cust1';

    Supplier_id := 'supp1';

    status := 'waiting';

    select DBMS_ASSERT.SIMPLE_SQL_NAME(destination) into dest  from dual ;

    select
    customer_name||ORDERID_SEQ.NEXTVAL,customer_name||CFPID_SEQ.NEX
    TVAL into  order_id,cfp_id from dual;

execute      immediate      'begin
ELOG_USERDETAILS.insertorder(''||order_id||',''||customer_name||'

                                ,''||Supplier_id||',''||cfp_id||'

                                ,''||product||',''||type1||'

                                ,''||pname||',''||quantity||'

                                ,''||status||',''||destination||''); end;' ;

```

exception when others then

```
http.p('<html><body>
```

```
<font color="maroon"><b/>
```

Destination field contains SQL keywords is validated by dbms_assert package.

```
</font>
```

```
</body></html> ');
```

```
elog_webpage.customer(customer_name);
```

```
endPlaceOrder;
```

```
procedureinsertorder(order_idscm_order.order_id%TYPE
```

```
,customer_namescm_order.customer_id%TYPE
```

```
,Supplier_idscm_cfp.Supplier_id%TYPE
```

```
,cfp_idscm_cfp.cfp_id%TYPE
```

```
,productscm_order.product%TYPE
```

```
,type1scm_order.prod_type%TYPE
```

```
,pnamescm_order.pname%TYPE
```

```
,quantityscm_order.qunatity%TYPE
```

```
,statusscm_order.status%TYPE
```

```
,destinationscm_order.destination%TYPE) is
```

```

begin

insert into scm_order

    (order_id,customer_id,product,prod_type,pname,qunatity,destination,status)

values(order_id,customer_name,product,type1,pname,quantity,destination,status);


insert into scm_cfp(cfp_id,order_id,supplier_id,product,quantity,status)

values(cfp_id,order_id,Supplier_id,product,quantity,status);

commit;

http.p('<html><body>

<font color="white"><b/>

order has been placed successfully.

</font>

</body></html> ');

ELOG_WEBPAGE.customer(customer_name);

endinsertorder;


procedureacceptOrder (index varchar2 default null

,select varchar2 default null

,index1 varchar2 default null

```

```

        ) is

v_orderorderlist;

begin

    if inde is not null then

        gv_order := getorder(inde);

        ELOG_WEBPAGE.supplier;

    end if;

    if selec is not null then

        v_order := getorder(index1);

        changestatus(v_order, selec);

        http.p('status changed');

        ELOG_USERDETAILS.getOrderDetails;

        ELOG_WEBPAGE.supplier;

    end if;

end acceptOrder;

function getorder(inde varchar2) return orderlist is

v_orderorderlist;

begin

```

```
select * bulk collect into v_order from scm_order where order_id=inde;
```

```
returnv_order;
```

```
endgetorder;
```

```
procedurechangestatus(v_orderorderlist,selec varchar2) is
```

```
begin
```

```
    UPDATE scm_order SET status = selec where order_id=v_order(1).order_id;
```

```
    UPDATE scm_cfp SET status = selec where order_id=v_order(1).order_id;
```

```
commit;
```

```
endchangestatus;
```

```
proceduremoveorder (inde varchar2 default null
```

```
,selec varchar2 default null
```

```
,index1 varchar2 default null
```

```
    ) is
```

```
v_orderorderlist;
```

```
begin
```

```
ifinde is not null then
```

```
gv_order := getorder(inde);
```

```
ELOG_WEBPAGE.logistics;
```

```
end if;
```

```
ifselec is not null then
```

```
  v_order :=getorder(index1);
```

```
  changestatus(v_order,selec);
```

```
  http.p('status changed');
```

```
  getOrderDetails;
```

```
ELOG_WEBPAGE.logistics;
```

```
end if;
```

```
endmoveorder;
```

```
proceduresearchorder (search_field varchar2,fieldvalue varchar2) is
```

```
  SQL1 varchar2(5000);
```

```
  c number :=1;
```

```
  TYPE EmpCurTyp IS REF CURSOR;
```

```
  emp_cvEmpCurTyp;
```

```
  fieldvalue1  varchar2(5000) :=upper(fieldvalue);
```

```
begin
```

```
  g_orderlist := orderlist();
```

```

--g_orderlist:=orderlist();

sql1 := 'select *

fromscm_order

where upper('||search_field||') like '''||'%'||fieldvalue1||'%'";

openemp_cv for sql1;

LOOP

FETCH emp_cv INTO ord_det;

exit when emp_cv%notfound;

g_orderlist.extend;

g_orderlist(c) := ord_det;

c:=c+1;

END LOOP;

CLOSE emp_cv;

exception when others then

http.p('<html><body><font    color="red">error    occurred    while    accessing
unauthorized program</font></body></html> ');

--g_orderlist := orderlist(ord_det);

ELOG_WEBPAGE.searchorder;

```


endsearchorder;

procedureinsertproduct(productname varchar2, subtype1 varchar2, prodmodel
varchar2) is

begin

if (insertprodtable(productname,subtype1,prodmodel) = 'Success') then

http.p('Product has been added in Database');

ELOG_WEBPAGE.insertporductpage;

else

http.p('<html><body>Please enter the details
correctly</body></html> ');

ELOG_WEBPAGE.insertporductpage;

end if;

ELOG_WEBPAGE.insertporductpage;

endinsertproduct;

functioninsertprodtable(productname varchar2, subtype1 varchar2, prodmodel
varchar2) return varchar2 is

pragmaautonomous_transaction ;

begin

insert into Product_master_tab values (productname,subtype1,prodmodel);

```
commit;

return 'Success';

exception when others then

return 'error';

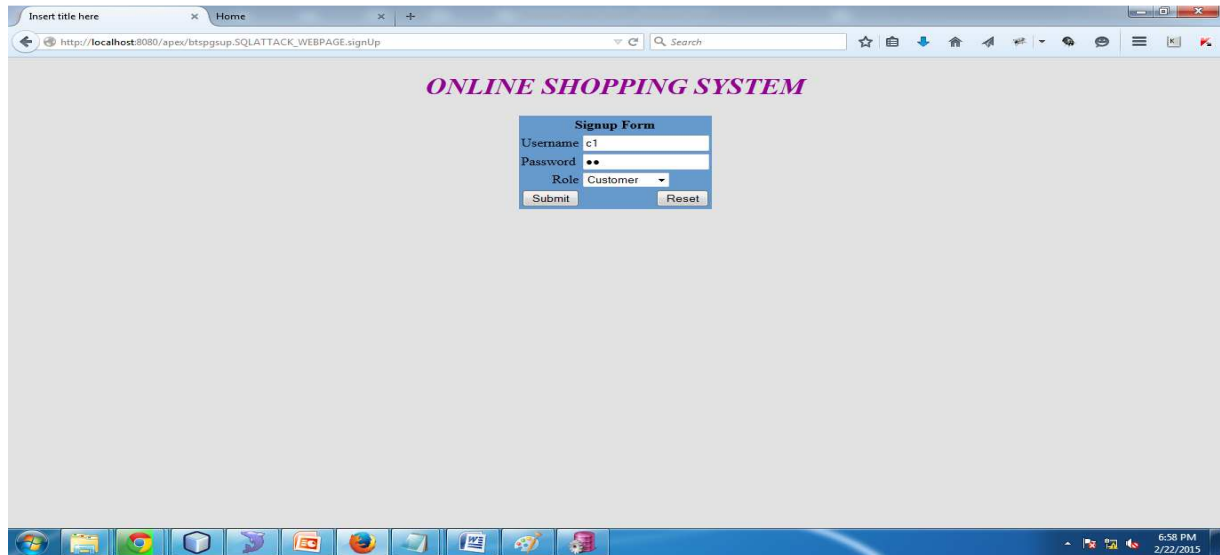
endinsertprodtable;

end ELOG_USERDETAILS;

/
```

Screen Shots

Signup:



The screenshot shows a web browser window with a single tab titled "Home". The address bar displays the URL "http://localhost:8080/apex/btspgsup.SQLATTACK_WEBPAGE.signUp". The page content features the title "ONLINE SHOPPING SYSTEM" in a purple, italicized font. Below the title is a "Signup Form" with the following fields: "Username" (containing "c1"), "Password" (masked with two dots), and "Role" (a dropdown menu set to "Customer"). At the bottom of the form are "Submit" and "Reset" buttons. The browser's taskbar at the bottom shows various application icons and a system clock indicating 6:38 PM on 2/22/2015.

Insert title here x Home x +

http://localhost:8080/apex/btspgsup.SQLATTACK_WEBPAGE.signUp

ONLINE SHOPPING SYSTEM

Signup Form

Username c1

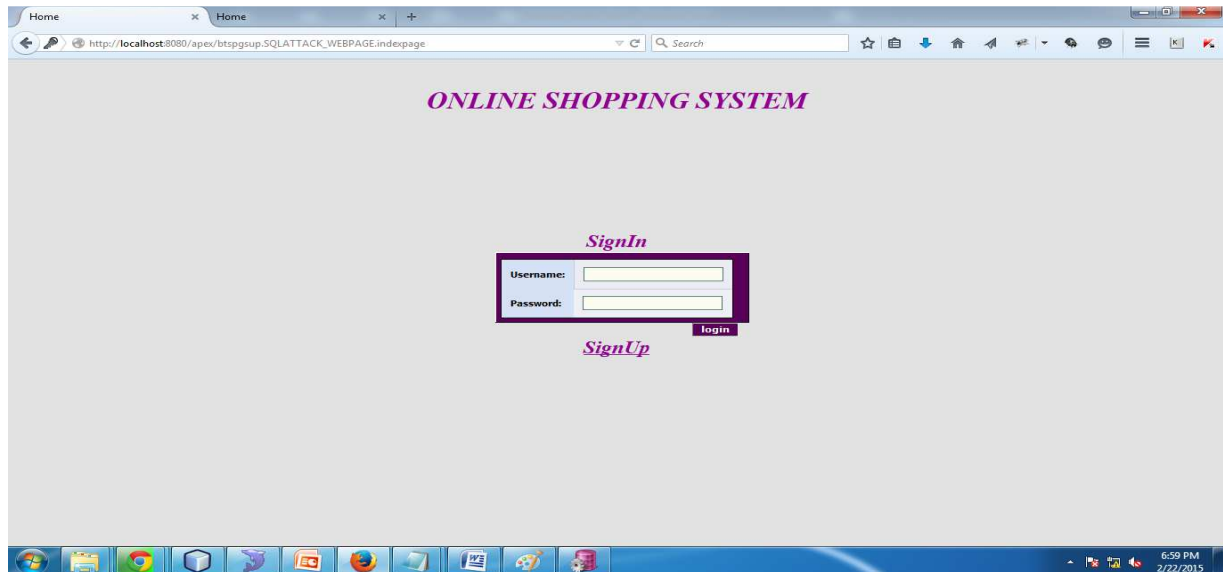
Password ••

Role Customer

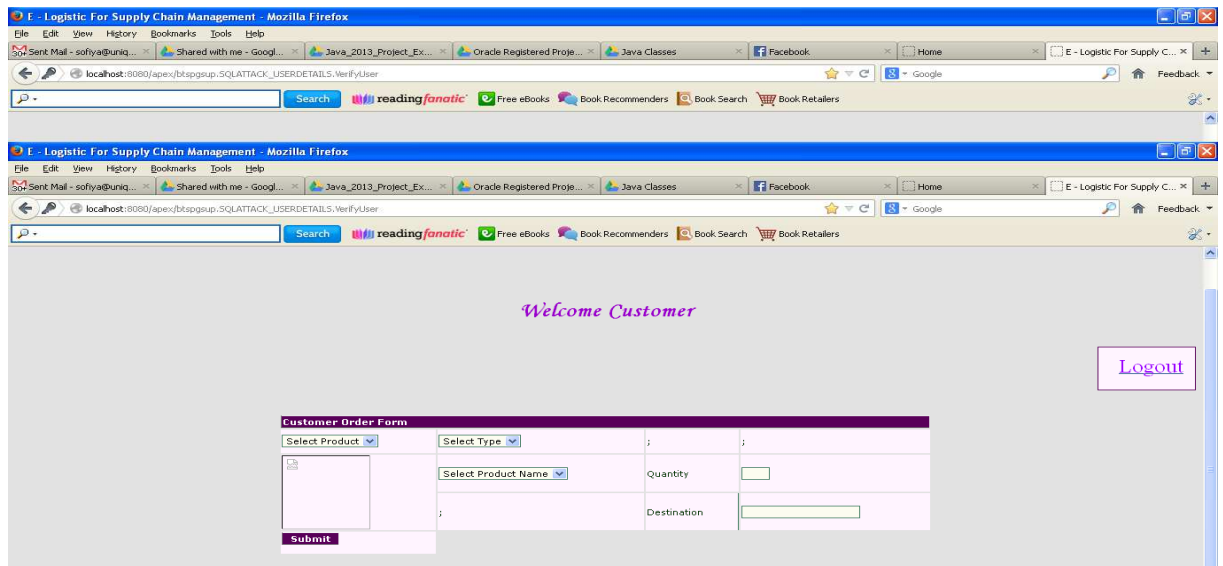
Submit Reset

6:38 PM 2/22/2015

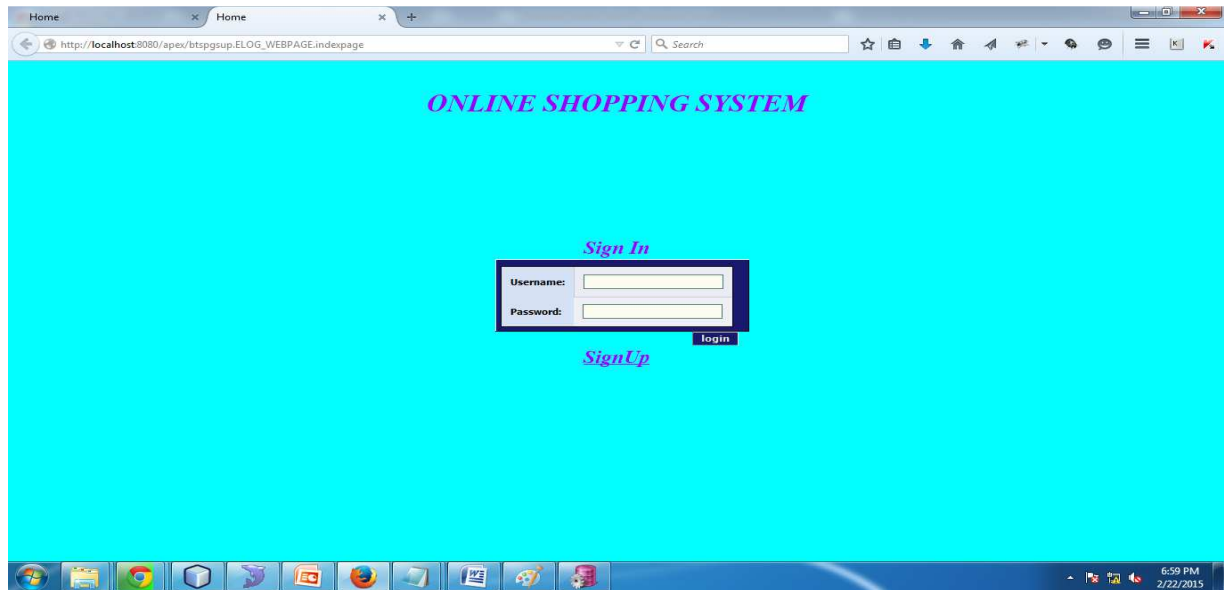
Existing Home Page:



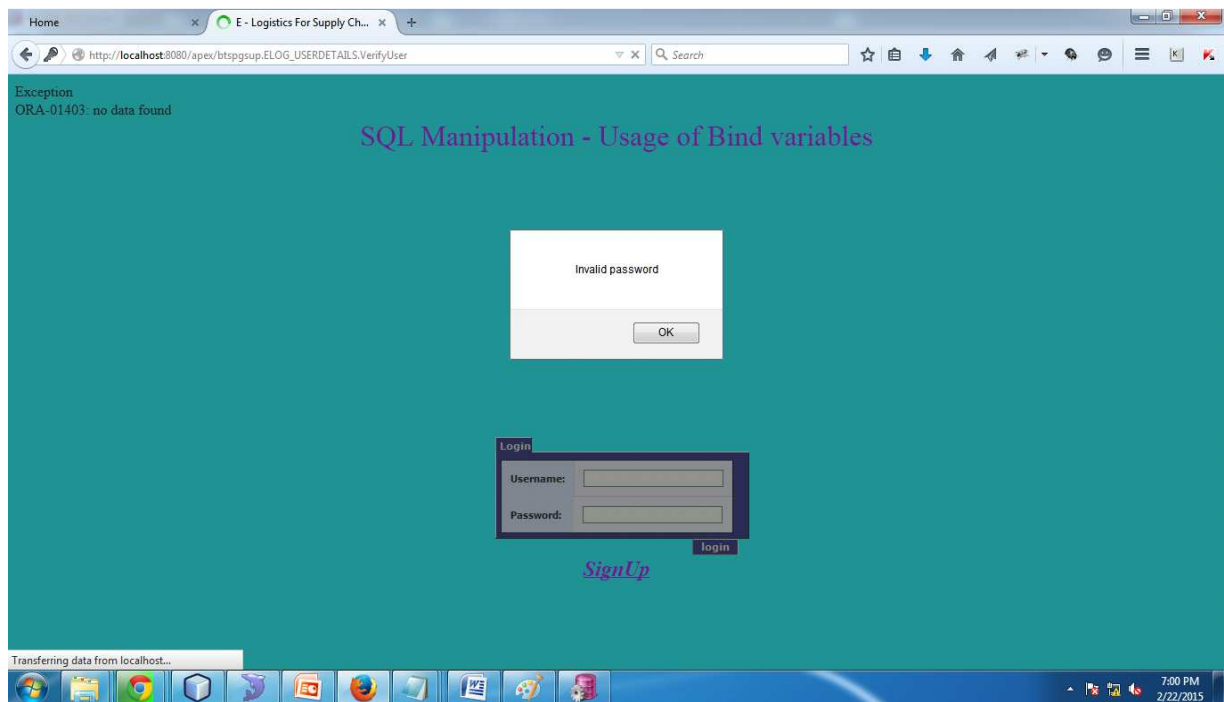
Bypass Login (allow to Next Page in Existing System):



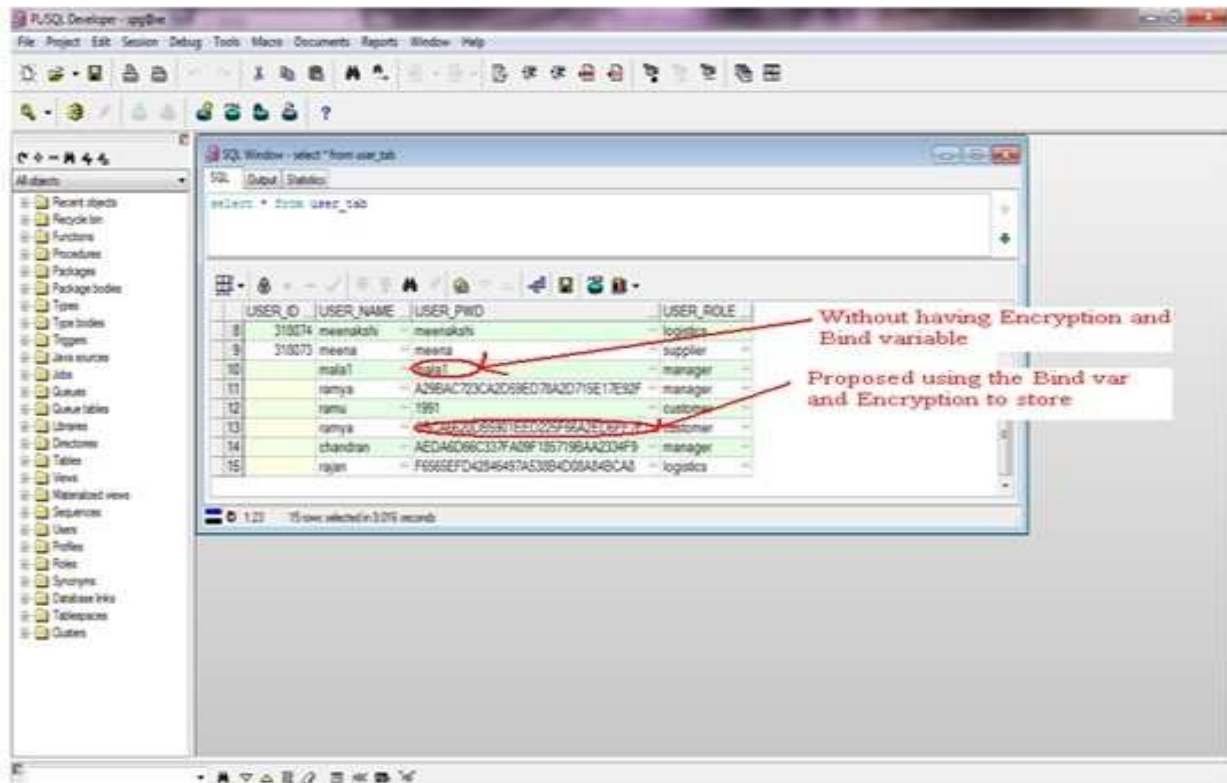
Proposed Home Page:



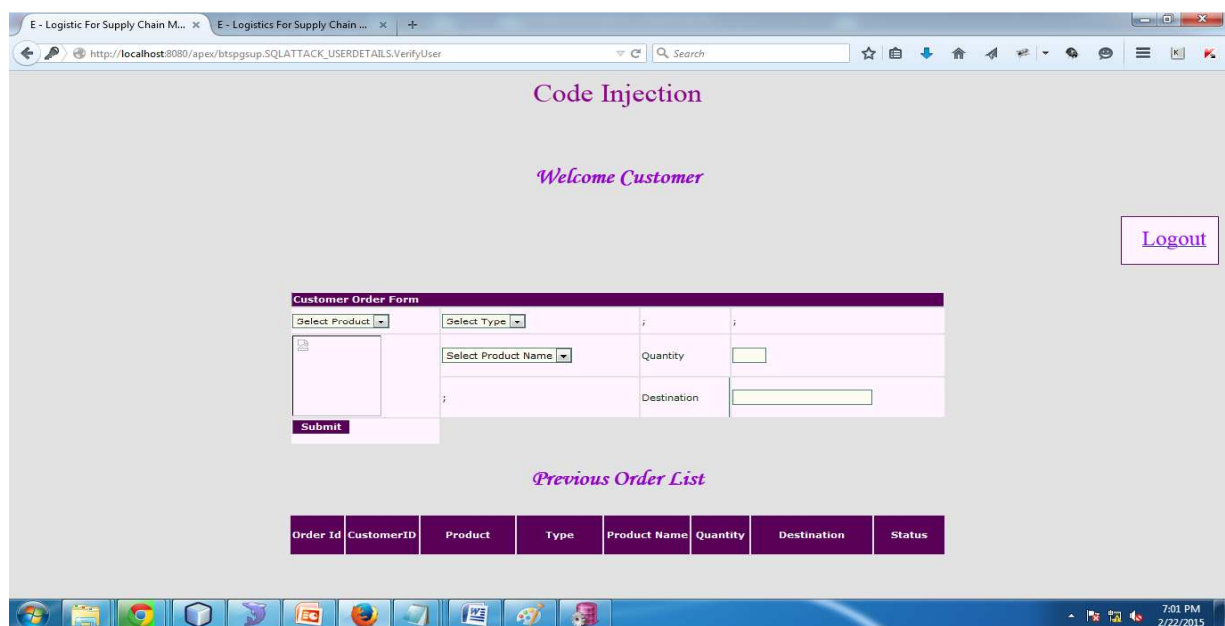
Proposed System (Not allow) because of bind variable and MD5:



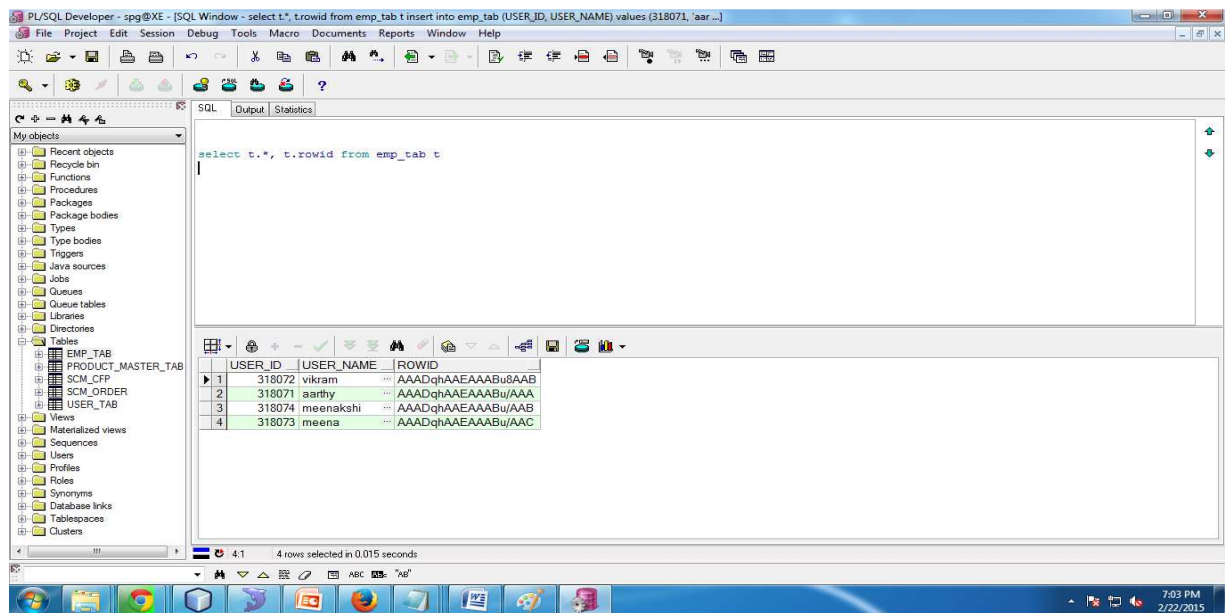
Data Storage structure of Systems:



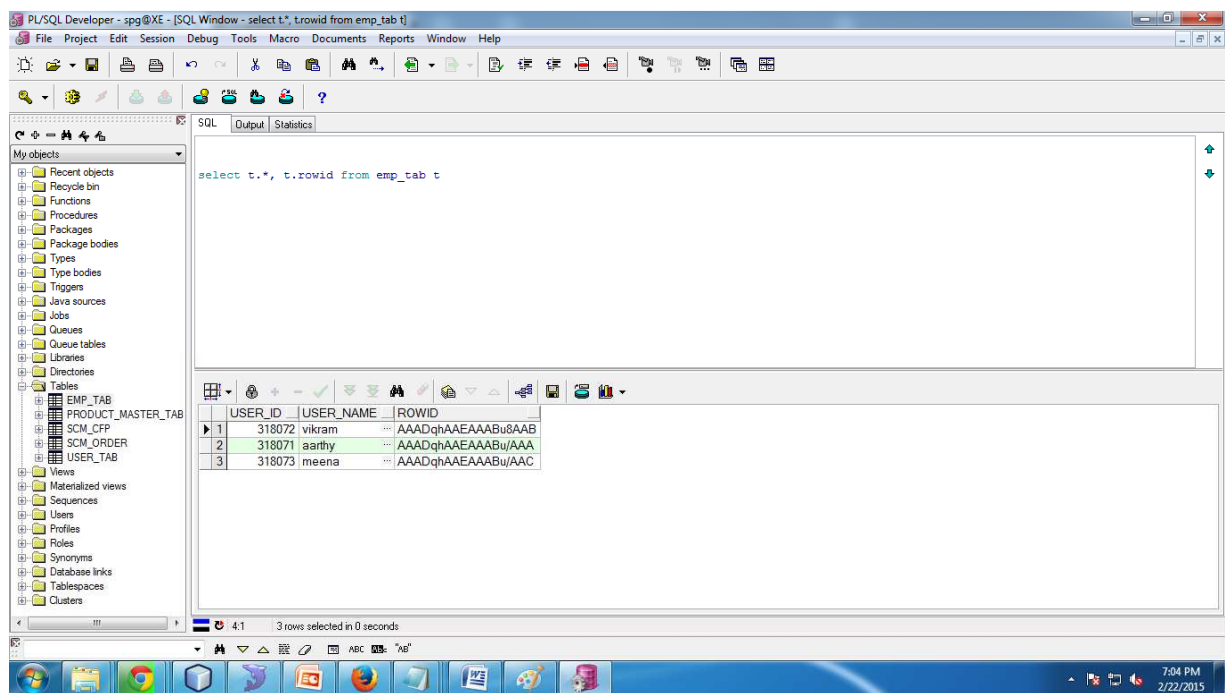
Code Injection:



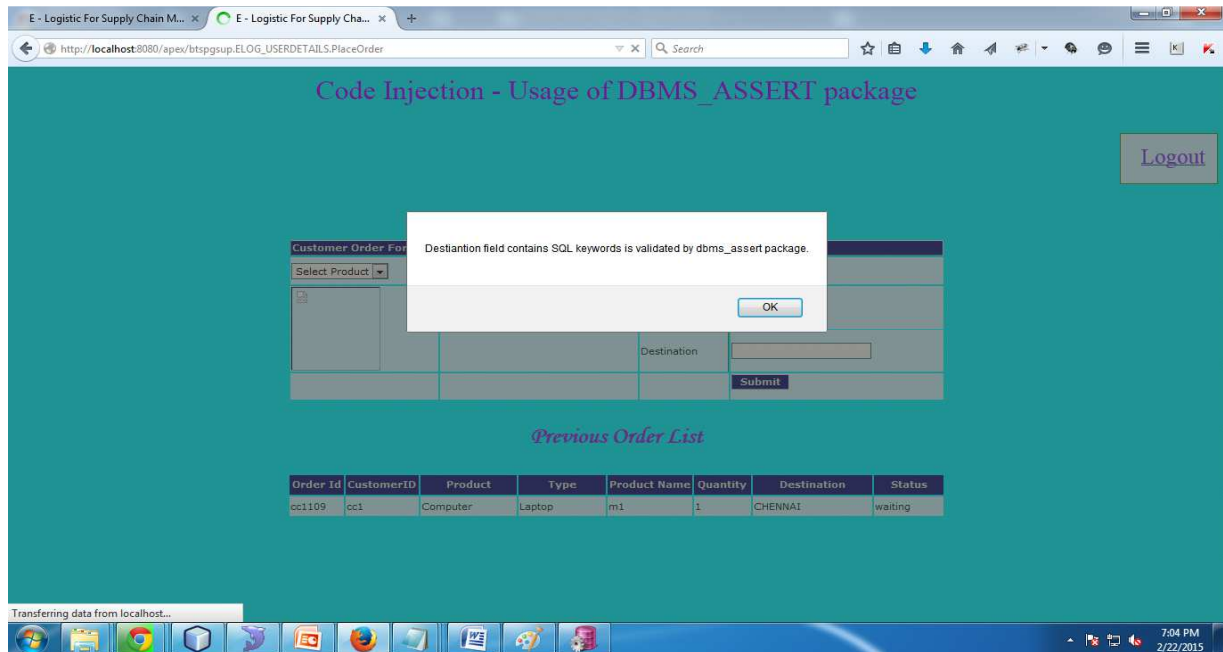
Before Code Injection Database:



After Code Injection Database:

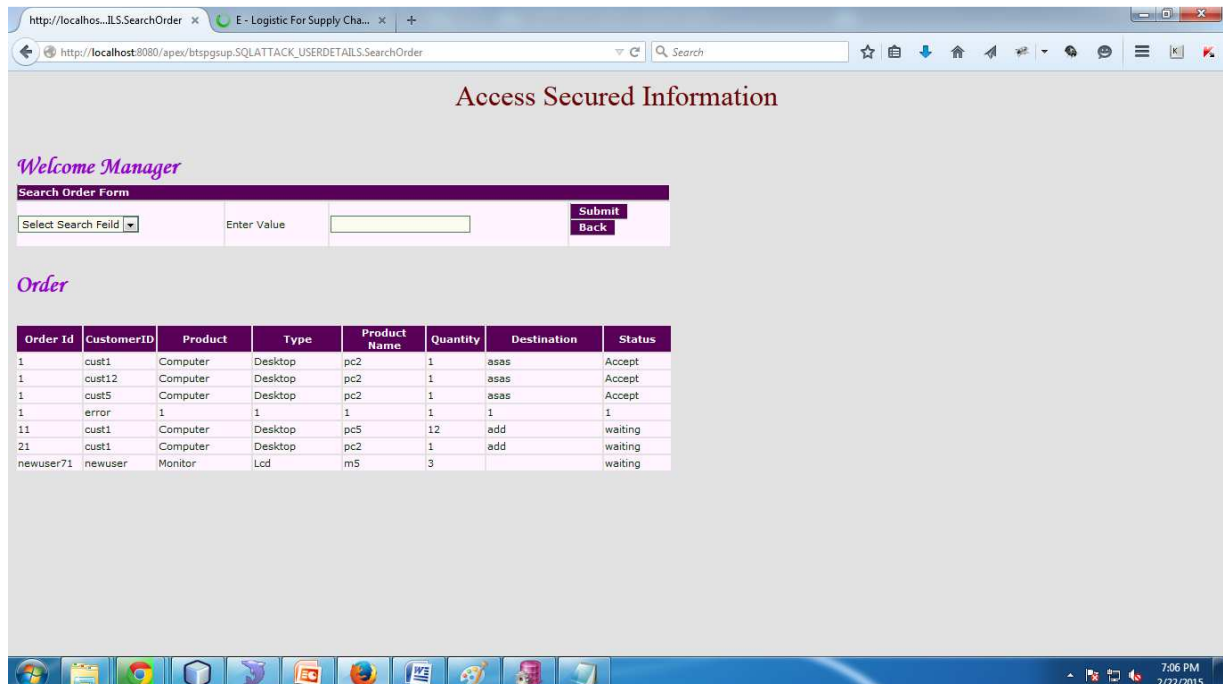


Proposed avoid the Code Injection:

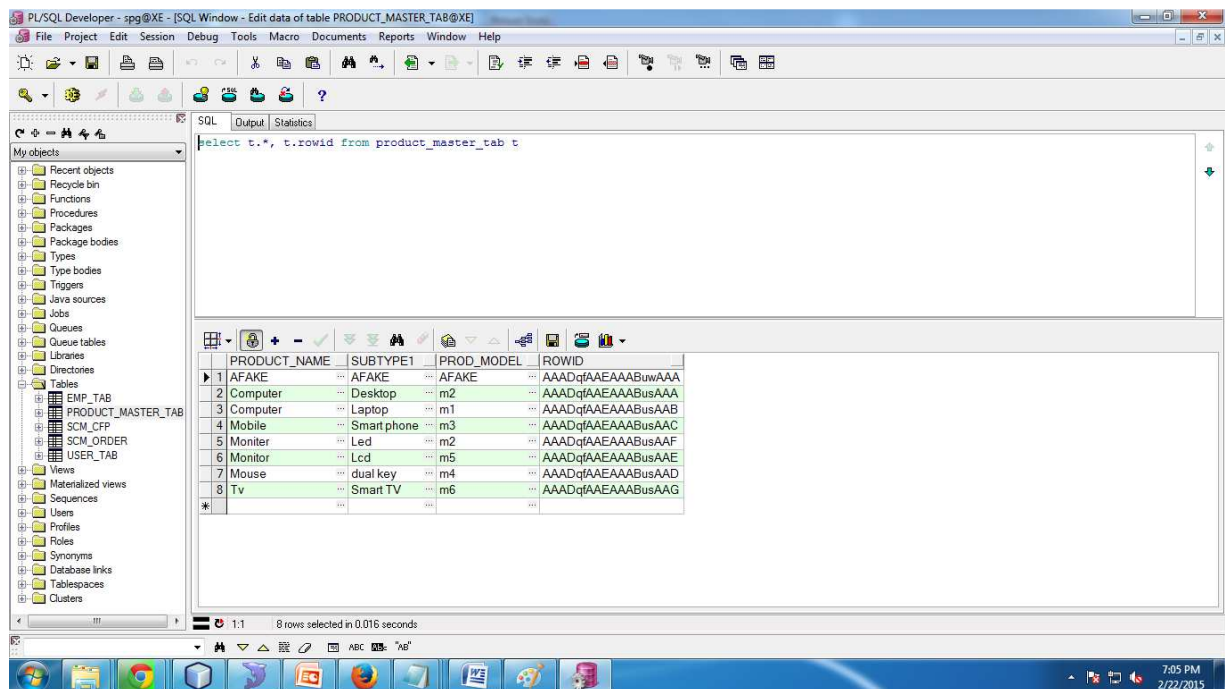


Function Call Injection:

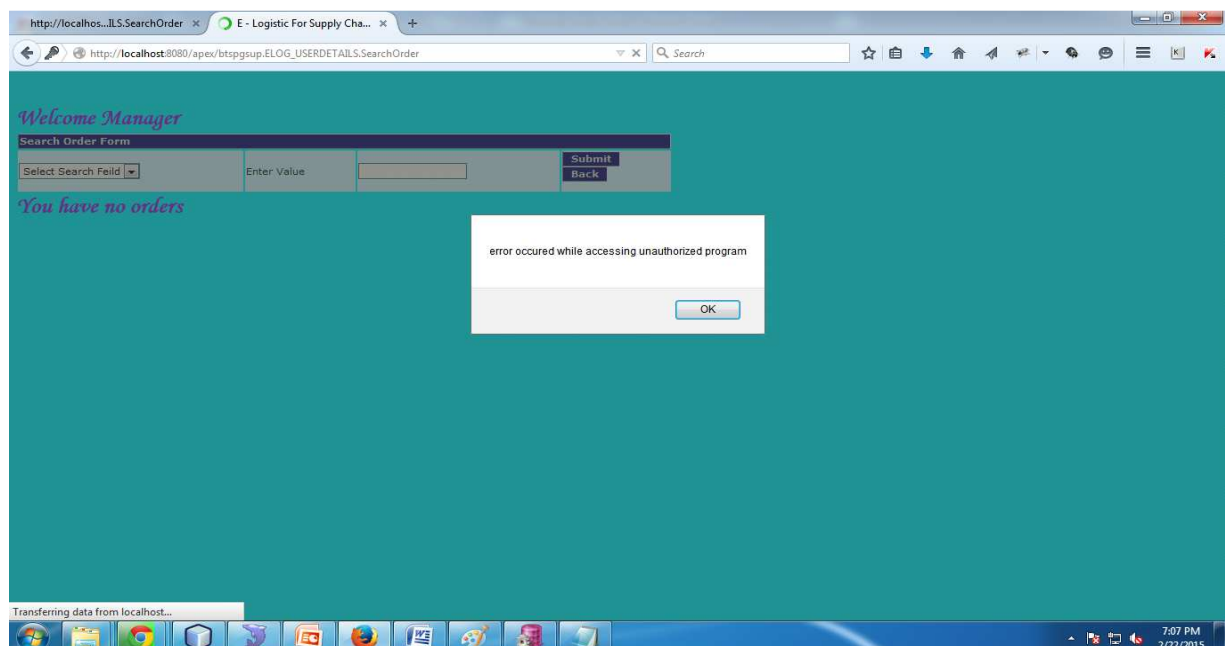
Existing:



Database After Function call Injection:



Proposed Avoid the function Call:



Access secure Information:

The screenshot shows a web browser window with the URL `http://localhost:8080/apex/btspgsql.SQLATTACK_USERDETAILS.SearchOrder`. The page title is "Access Secured Information". Below the title, there is a "Welcome Manager" message and a "Search Order Form" with a dropdown menu labeled "Select Search Field", an input field labeled "Enter Value", and two buttons: "Submit" and "Back".

Below the search form, there is a section titled "Order" containing a table with the following columns: Order Id, CustomerID, Product, Type, Product Name, Quantity, Destination, and Status. The table contains 15 rows of data, all with a quantity of 1 and a status of 1.

Order Id	CustomerID	Product	Type	Product Name	Quantity	Destination	Status
1	Procedure manager;	1	1	1	1	1	1
1	end SECURE_PACKAGE;	1	1	1	1	1	1
1	package SECURE_PACKAGE is	1	1	1	1	1	1
1	procedure csa;	1	1	1	1	1	1
1	procedure customer(customer_name varchar2 default null);	1	1	1	1	1	1
1	procedure indexpage;	1	1	1	1	1	1
1	procedure insertporductpage;	1	1	1	1	1	1
1	procedure loginPage;	1	1	1	1	1	1
1	procedure logistics;	1	1	1	1	1	1
1	procedure searchorder;	1	1	1	1	1	1
1	procedure signUp;	1	1	1	1	1	1
1	procedure signUpSuccess;	1	1	1	1	1	1
1	procedure supplier;	1	1	1	1	1	1

Proposed System:

The screenshot shows a web browser window with the URL `http://localhost:8080/apex/btspgsql.ELOG_USERDETAILS.SearchOrder`. The page title is "You have no orders". Below the title, there is a "Welcome Manager" message and a "Search Order Form" with a dropdown menu labeled "Select Search Field", an input field labeled "Enter Value", and two buttons: "Submit" and "Back".

Below the search form, there is a message "You have no orders". A modal dialog box is displayed in the center of the screen with the text "error occured while accessing unauthorized program" and an "OK" button.

REFERENCES

1. B. Bhattacharjee, N. Abe, K. Goldman, B. Zadrozny, C. Apte, V.R. Chillakuru, and M. del Carpio, "Using Secure Coprocessors for Privacy Preserving Collaborative Data Mining and Analysis," Proc. Second Int'l Workshop Data Management on New Hardware (DaMoN '06), 2006.
2. E. Mykletun and G. Tsudik, "Aggregation Queries in the Database-as-a-Service Model," Proc. 20th IFIP WG 11.3 Working Conf. Data and Applications Security, pp. 89-103, 2006.
3. R.A. Popa, C. Redfield, and N. Zeldovich, "Cryptdb: Protecting Confidentiality with Encrypted Query Processing," Proc. 23rd ACM Symp. Operating Systems Principles (SOSP '11), 2011.
4. V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motwani, "Distributing Data for Secure Database Services," Proc. Fourth Int'l Workshop Privacy and Anonymity in the Information Soc. (PAIS '11), pp. 8:1-8:10, 2011.
5. Y. Chen and R. Sion, "To cloud or Not to Cloud?: Musings on Costs and Viability," Proc. Second ACM Symp. Cloud Computing (SOCC '11), pp. 29:1-29:7, 2011.
6. S. Bajaj and R. Sion, "TrustedDB: A Trusted Hardware Based Outsourced Database Engine," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2011.
7. E. Mykletun and G. Tsudik, "Incorporating a Secure Coprocessor in the Database-as-a-Service Model," Proc. Innovative Architecture on Future Generation High-Performance Processors and Systems (IWIA '05), pp. 38-44, 2005.

8. N. Anciaux, M. Benzine, L. Bouganim, P. Pucheral, and D. Shasha, "GhostDB: Querying Visible and Hidden Data Without Leaks," Proc. 26th Int'l ACM Conf. Management of Data (SIGMOD), 2007.
9. L. Bouganim and P. Pucheral, "Chip-Secured Data Access: Confidential Data on Untrusted Server," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), pp. 131-141, 2002.
10. G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two Can Keep a Secret: A Distributed Architecture for Secure Database Services," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 186-199, 2005.
11. TPC-H Benchmark, <http://www.tpc.org/tpch/>, 2013.
12. MD5 Code Converison technique
<http://nsfsecurity.pr.erau.edu/crypto/md5.html>
13. Bind Variable technology implementation
http://docs.oracle.com/cd/A81042_01/DOC/sqlplus.816/a75664/ch34.htm.
14. FIPS PUB 140-2, Security Requirements for Cryptographic Modules,
<http://csrc.nist.gov/groups/STM/cmvp/standards.html#02>, 2013.
15. Wrapper classes for the primitive types in JAVA www.cs.rit.edu/~rlaz/cs2-20082/slides/WrapperClasses.pdf
16. IBM 4764 PCI-X Cryptographic Coprocessor, <http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml>, 2007.
17. SCPU Validating the high performance
csrc.nist.gov/nissc/1999/proceeding/papers/p16.pdf
18. DBMS Assert conceptual theory Implementation http://oracle-base.com/articles/10g/dbms_assert_10gR2.php

Journal Acceptance

