MULTI MECHANISM APPROACH FOR PREVENTING SQL INJECTION ATTACKS IN STORED PROCEDURES

HASANEIN D. RJEIB1, HAIDER ALSHARQI2 BASHEER AL-SADAWI3
1Faculty of Engineering, University of Kufa
2Information Technology Research and Development Center, University of Kufa,
3IT-RDC center, University of Kufa, Al-Najaf, Iraq.

Abstract- Recently, Web applications have been increased significantly in online services for most of the activities in our animation that are based on Internet-enabled applications. However, ignoring privacy and security aspects of the applications by web developers makes them attractive targets for security issues. In this article, a proposed protective approach that includes three preventive mechanisms that are: parameterized stored procedures, customized error messages, and encryption stored procedures in SQL server to prevent the danger of Structured Query Language Injection Attacks (SQLIA) in stored procedures. In addition, an analysis evaluation of the proposed protective approach with respect to the effectiveness approach is conducted. The proposed protective approach was effectiveness due to it capable of preventing the attacks of stored procedures SQLIAs efficiently.

Keywords: Preventing approach; SQLI; Stored procedures; Injection attack; SQLIA; SQL server.

I. INTRODUCTION

Nowadays, the Internet becomes a widely significant adoption gate for information dissemination and various other online transactions through inventing the wheel for the revolution of informatics in the recent years [1], [2]. We are using the Internet or web applications for most of the activities in our animation [3], [4]. Thereby, the Internet is becoming widespread information infrastructures. Since the emergence of web programming, web applications have become an adequate way to offer access to online services via the Internet [5]. It led applications gain a huge popularity in the world due to; they have achieved enterprise integration through; they allowed a numerous Internet-enabled applications [6], [7]. Web applications typically interact with backend underlying database, whereby, the data underlying web applications often have sensitive information and confidential [8], [9]. Web applications are frequently vulnerable to attacks due to lack in design, configuration faults, or weakness written code of the web applications [10], [11]. Structured Query Language Injection (SQLI) attack takes the benefit of trust existing between the users and the server as well as the feature of an absented input/output validation on the server to reject malicious codes [12]. However, it is important to provide the protection of the web applications from the targeted SQLIAs [13]. According to Halford et al. 2008, proposed an approach to prevent the SQL injection attacks by developed an automated tool which is called WASP tool. The proposed tool stopped all types of attacks successfully and efficiently without generating any false positives [14]. Likewise, in 2010 Bau et al., introduced a black-box scanners detection tool to scan stored SQLI and XSS vulnerabilities. Confirming script injection to storage was the limitation of the presented tool [15]. As well as, Yan et al. 2011, conducted a database system to analyze and protect SQLIAs. The protection system deployed between the application and database. The system supported diverse assessments to ensure database security effectively respect to protective for administrators and ordinary users [16]. As well, Gadgil, 2013 conducted an online banking application to prevent different kinds of SQLIAs. The proposed application was based on WASP tool and it limited to deploy the system...
in real web settings [17]. Furthermore, RT-WASP tool presented by Ali and Shibghatullah, 2016 in a web application based. The suggested tool detect stored procedures SQLIA in real-time environment efficiently and effectively [18]. In addition to, SQLIA intrusion detection framework conducted by Yassin et al., 2017 to provide a high level of portability in the application of SaaS provider's with a service-based manner to improve the security [19]. The current research focuses on the vitality and importance of the problem SQL injection attacks in stored procedures to detect and prevent such attack in the web application environment via proposing a protective approach to prevent stored procedures SQLIAs. The proposed approach includes three mechanisms to encounter web applications or websites from SQLIAs in stored procedures. The steps of proposing a protective approach will be presented and listed in the following Sections.

II. PROPOSING A PROTECTIVE APPROACH (MATERIALS AND MECHANISMS)

User Interface (UI) is required for the interaction between the user and the system. Internet Information Service (IIS) server customization is used to create a new account or log in to an existing account to access websites. The approach includes hardware and software requirements to implement and evaluate the conducted approach. The hardware components involve P-IV– 0.5 GHz to 3.0 GHz processor, 1GB RAM, 20 GB Hard Disk, and SVGA monitor. Whereas, the software requirements are Windows 7/XP operating system, ASP.NET (with C# programming language) is the development end, HTML, ASP.NET, and CSS of the web technology, SQL server 2008 R2 for database server, and the IDE is Microsoft ASP.NET. The first aim of the proposed approach is to mitigate, prevent web applications or websites from storing procedures SQL injection that is ready to stop, prevent stored procedures SQLIA attacks before queries reach the database and cause any damage for sensitive information. These attacks include: bypass attacks with dynamic queries, error messages via a call stored procedures, and no encryption when creating stored procedures in SQL server. We prevent these attacks by adding some mechanisms on the programming level that may ensure the security of the websites or web applications against the stored procedure SQLIA. These mechanisms encompassed encryption, customize error messages and parameterized queries with input validation. Figure 1 present and explain the mechanisms of the preventive approach.

![Figure 1. Mechanisms of the proposed Protective Approach](image-url)
Figure 1 provides a general intuitive overview of the proposed approach that conducted to protect web applications or websites from the danger of SQLIAs in stored procedures. The protective approach includes three methods or mechanisms to perform the preventive from the attack. These mechanisms are conducted to protect a web application from three types of the attacks that are: bypass attacks with dynamic queries, error messages via a call stored procedures, and no encryption when creating stored procedures in SQL server. These types of attacks can be implemented via sending malicious SQL statement (error keywords query) by an attacker to get illegal access to the backend database.

III. IMPLEMENTATION OF THE PROPOSED APPROACH

This Section describes techniques, methods, and protective mechanisms of the proposed approach that is used to prevent the SQLIA in stored procedures. The implementation of the proposed approach will be explained in details and the testing results will be presented by designing two websites which are injected and protect from the SQLIA in stored procedures respectively. The goal of our protective approach is that we propose a way to prevent the web applications or websites from the danger of the SQLIA in stored procedures. These attacks include: bypass attacks with dynamic queries, error messages via a call stored procedures and no encryption when creating stored procedures in SQL server. We prevent these attacks by adding some mechanisms on the programming level that may ensure the security of the websites or web applications against the stored procedure SQLIA. These mechanisms encompassed encryption, customize error messages and parameterized queries with input validation. As we see in Figure 2.

![Implementation Steps of the proposed Approach](image-url)
Our experiment is written in C# programming language and is implemented and tested our protective approach by ASP.NET infrastructure. In our protective approach, we use three mechanisms or methods to prevent the SQLIA in stored procedures such as: parameterized stored procedures, customized error messages, and encryption stored procedures in SQL server. In order to prevent the attack types respectively: bypass attacks with dynamic queries (concatenation keywords), call stored procedures (error messages), and create stored procedures (no encryption) in the SQL server. These types of attacks can be implemented via sending malicious SQL statement (error keywords query) by an attacker in order to get illegal access to the backend database. The next Section, illustrates the procedure of the prevention techniques or methods in each mechanism that is mentioned in Figure 2.

To evaluate our proposed approach, we conduct one experiment with two scenarios by design two websites that are presented a student information system (SIS), and they provided many services such as searching by all, by the first name, adding a new student, etc.. The first scenario (website injected) was vulnerable to different types of SQLIAs. The vulnerable site included the loopholes in both dynamic queries, called stored procedures, and created stored procedures in the SQL server. Whilst, the second scenario (website protected), designed to protect these kinds of attacks, in particular, in stored procedures methods or type. As we see in Figure 3. The testing of the web application will be presented and discussed in the next Section.

IV. MECHANISMS FOR PREVENTING STORED PROCEDURE SQLI ATTACKS

1. Parameterized Stored Procedures

This Section first illustrates the target attack that is a bypass attack (concatenation) with dynamic queries, and explains in details, all steps to prevent this type of attack by using the protective mechanism or method that is parameterized stored procedures to address SQLIA. As we see in Figure 4.
Figure 4. Steps for Parameterized Stored Procedures Mechanism

- **Step One (AN Attacker)**

When an attacker sends malicious code (bypass injection attacks) to implement the injection attack of the website, as we show in Figure 5, the attacker can insert any name and inject it by one of the keyword injection that is always denoted to True value as a syntax result in the SQL server, for instance, when the attacker puts Ahmed’ or 1=1--, then he can get all the students name.
• **Step Two (Malicious SQL Statement or Query)**
  After an attacker sent a malicious code, a web application will build malicious SQL statement or error keywords query that will be checked in the next step. As we have shown in Figure 5, search by the first name by using dynamic query can be implemented by this query:
  \[
  \text{Select * From SISstable Where FirstName = } \text{"} + \text{TextBox1.Text} + \text{""};
  \]

• **Step Three (Verify Query)**
  The query will be checked in order to determine whether it has any bypass injection attack (malicious code). In the worst case, when an attacker can get access to the database, after that, the attacker can get all sensitive information of the student names in our experiment. As we showed in Figure 5.

• **Step Four (Parameterized Stored Procedures)**
  The protection method or mechanism that will be used to prevent the attacks of bypass injection is the parameterized stored procedures at the programming level that may ensure the security of the web applications or websites against the SQLIA. As we see in Figure 6 that explain the stored procedures code with the input parameter.

```sql
CREATE PROCEDURE spSearchByFirstName
  @Name nvarchar(50)
AS
BEGIN
  SELECT FirstName, LastName, Gender, Department, CourseType, MatricNo, StdCountry
  FROM SISstable
  WHERE FirstName = @Name
END
```

**Figure 5. Bypass Injection Attack**

**Figure 6. Parameterized Stored Procedure Code**
We have protected our website from the injection attack that caused by using dynamic queries, by using stored procedure to avoid this attack. That is illustrated by using the parameterized stored procedure with an input parameter to avoid a particular attack.

2. Customized Error Messages

This Section lists in details the attack of error messages by calling the stored procedures that is retrieved from SQL server which is considered one of the SQLIA in stored procedures. This Section also presents all steps that are required for protecting the application or website from this attack by using a mechanism to customize the error messages. As we see in Figure 7.

![Figure 7. Steps for Customized Error Messages Mechanism](image-url)

1. Step One (Attacker Intent)

An error message is one kind of SQLIAs. The stored procedure should be affected from the feedback error message that retrieved from the server side (SQL Server) if there is no customization of the error message. As a result from this attack, the attacker can try to get useful information regarding stored procedures such as name, parameter’s name, etc. as a trick to achieve the attacker target as shown in Figure 8.
Figure 8. Call Procedures Injection (Error Message)

2. **Step Two (SQL Statement or Query)**

   After an attacker sent a malicious code that aims to inject the application or website by using call stored procedure method, web application builds a malicious SQL statement that is checked in the next step.

3. **Step Three (Verify Query)**

   SQL server will check the SQL statement whether it has any call procedures injection then, it will retrieve the error message that would display sensitive information about the stored procedures. In the worst case, when the website is vulnerable to this attack, the database server (SQL server) will retrieve the error message to the user. The error message injection can be caused if there is any error definition in the parameters or the parameter calling such as:

   ```
   cmd.Parameters.Add (new SqlParameter ("+@Name+", TextBox2.Text));
   ```

4. **Step Four (Preventive Mechanism)**

   As we mentioned, an error message or feedback error message is retrieved from the SQL server as a response to any error query that sent to SQL server. The attacker can get a useful information about the target and achieve his goal such as the table’s name, stored procedure’s name, etc., in order to avoid or prevent this kind of the attack by using the mechanism of the customized error messages at the programming level such as try and catch inside the stored procedures in the web applications or websites to enforce the security of the applications against the attack of call stored procedures (error messages). As we see in Figure 9.
3. Encryption Stored Procedures

This Section explains the creation of stored procedures in the SQL server by injecting one of the systems defined commands which affects the stored procedures programming codes, and illustrates the steps to prevent this attack by using encryption stored procedures mechanism. As we see in Figure 10.

![Figure 10. Steps for Encryption Stored Procedure Mechanism](image-url)
1. **Step One (Attacker Intent)**
   An attacker can gain access to the database sources and get sensitive information regarding stored procedures and tables due to poor insecure to the commands that they defined respect to users and systems in the programming level. As we show in Figure 11.

2. **Step Two (SQL System Commands)**
   Some of the system commands inside the data source (SQL server) can be used by the attacker to present the contents of the stored procedure code such as (sp_helptext, sp_help, and sp_depends).

3. **Step Three (Create Stored Procedure in SQL Server)**
   As a result of the attack, the attacker can know all the sensitive information that related to the name, data, tables, etc. And then; an attacker can use one of the attack methods after he got the useful information that obtained by using the command system in the server side regarding the stored procedures codes, and can alter, delete, update, etc. to the database or stored procedure. As we show in Figure 11.

4. **Step Four (Preventive Mechanism)**
   We protected the stored procedures from the attacker who want to display the codes or the contents of the stored procedures or also know sensitive information about the stored procedures regarding tables, parameters, etc. This attack is done by the use of one of the system commands such as sp_helptext, etc. Hence, we can avoid this attack by using the encryption method when creating the stored procedures at the programming level to make sure the attacker cannot know or get any sensitive information of the stored procedures. As we show in Figure 12.

---

**Figure 11. Injection Stored Procedure in SQL Server**

<table>
<thead>
<tr>
<th>Execute of Stored Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>All information of Stored Procedure contents</td>
</tr>
</tbody>
</table>

---

@IJRTER-2018, All Rights Reserved
To evaluate the efficiency and the effectiveness of the proposed approach to prevent the attacks of SQLI in stored procedures. Our approach has many protective mechanisms such as parameterized stored procedures, customized error messages, and encryption stored procedures in the SQL server as we mentioned in Section Three. These protective mechanisms that use to address the attacks that carry types are: bypass attacks with dynamic queries call stored procedures (error message), and no encryption stored procedures in the SQL server. For the purpose of testing, we first test our approach with one experiment (Student Information System websites) that includes two scenarios (websites) to evaluate our approach effectiveness. The first scenario is injected, and the second is protected to similar attacks. We divide each website into two groups: expected result and the actual result. The results from the testing between web applications (websites) can conclude the effectiveness of the proposed approach by preventing and blocking the attacker to get illegal access to the database. Tables 1 and 2 summarize the comparison between the two scenarios of our experiment to perform testing and the evaluation of the protective approach.

1. **First Scenario (Website Injected)**

Figure 13 shows the user interface design of the first scenario (student information system website) of our experiment to test the proposed approach to prevent SQLIA in stored procedures as we mentioned. The website is vulnerable to different types of SQLIAs. These attacks types are bypass attack with...
dynamic queries, called stored procedures (error messages), and no encryption when creating stored procedures in SQL server. Student information system (SIS) has one page in the UI and provide several services such as, search all the students via using an ad-hoc query, search by the first name via using dynamic query and input parameters with stored procedures.

2. Second Scenario (Website Protected)

In this Section, we explain the user interface designed for the second scenario of our experiment which is website protected from SQLIAs in particular, stored procedure attack type. By adding some mechanisms at the programming level that may ensure the security of the website or web application against the SQLIAs. These mechanisms include encryption methods, customize error messages, a parameterized query with input validation. As we have shown in Figure 14. The website (SIS) has two pages in the UI and provide several services such as, search all the students via simple stored procedures, search by the first name via using stored procedures with input parameters, and insert new student with output parameters and return value. We conducted a comparison analysis of the proposed protective approach based on the evaluations respect to the effectiveness of the proposed protective approach. In our evaluations, we evaluated our protective approach based on the effectiveness metrics regarding the capability to prevent the target attacks that are: bypass attacks with dynamic queries, called stored procedures SQLIA (error messages), and no encryption create stored procedures in the SQL server.

Table 1. First Scenario of the Testing Experiment (Injected)

<table>
<thead>
<tr>
<th>Test Case No.</th>
<th>Test Case Name</th>
<th>Test Case Description</th>
<th>Step to be Executed</th>
<th>Expected or Predicated Result</th>
<th>Real or Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>Bypass attack with dynamic queries SQLI</td>
<td>Visiting the home page within the SIS site</td>
<td>Typing down the error keywords or 1=1’ on the textbox. Text and click in search by first name button</td>
<td>Getting all the information students records that stored in the database server</td>
<td>Getting the students information with details</td>
</tr>
<tr>
<td>TC2</td>
<td>Call stored procedures SQLIA</td>
<td>Visiting the home page within the SIS site</td>
<td>Typing down the error keywords or 1=1’ on the textbox. Text and click in search by first name button</td>
<td>Getting of some of sensitive information regarding to stored procedure and tables in the database by the error messages that</td>
<td>Getting of some of sensitive information regarding to stored procedure and tables in the database server</td>
</tr>
<tr>
<td>Test Case No.</td>
<td>Test Case Name</td>
<td>Test Case Description</td>
<td>Step to be Executed</td>
<td>Expected or Predicated Result</td>
<td>Real or Actual Result</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------------</td>
<td>-------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>TC1</td>
<td>Bypass attack with dynamic queries (SQLI).</td>
<td>Visiting the home page within the SIS site.</td>
<td>Typing down the error keywords or 1=1` on the textbox. Text and click in search by first name button.</td>
<td>Blocking the attacker from viewing the records or details of the student information by prevent him to get illegal access to the database.</td>
<td>Blocking the attacker from viewing the records or details of the student information.</td>
</tr>
<tr>
<td>TC2</td>
<td>Call stored procedures SQLIA (error messages).</td>
<td>Visiting the home page within the SIS site.</td>
<td>Typing down the error keywords or 1=1` on the textbox. Text and click in search by first name button.</td>
<td>Prevent the attacker from get some of the sensitive information of the stored procedures by error messages that retrieve to the attacker from SQL server.</td>
<td>Prevent the attacker from get some of the sensitive information of the stored procedures.</td>
</tr>
<tr>
<td>TC3</td>
<td>Create no encryption stored procedures.</td>
<td>Get access on SQL server and use one of the system define commands.</td>
<td>Typing down one of the system commands in the SQL server side to view the context of the procedure codes.</td>
<td>Blocking the attacker from display the sensitive information of the procedures and tables in the SQL server by secure the data inside the procedures to prevent viewing by any attacker.</td>
<td>Blocking the attacker from display the sensitive information of the procedures and tables in the SQL server.</td>
</tr>
</tbody>
</table>

**Table 2. Second Scenario of The testing Experiment (Protected)**

VI. RESULTS AND DISCUSSIONS

We evaluated our proposed approach to prevent the attacks of stored procedures in SQL to assess whether our preventive approach is effective to prevent the threats of the SQLIA based on the capabilities to prevent many kinds of the SQLIA in stored procedures as we summarized in Tables 1 and 2. Based on the evaluation results of the proposed approach, there are many protective mechanisms to prevent the attacks of the SQLIA in stored procedures. The result shows that our mechanisms are effective to protect the web applications against SQLIA in stored procedures efficiently. These protective mechanisms encompassed respectively: parameterized stored procedure mechanism to prevent bypass attack with dynamic queries, customized error message mechanism to prevent the attack...
of call stored procedures (error messages), and encryption stored procedures in order to, prevent the attack of injection in the SQL server-side when create store procedures by using one of the SQL server systems define commands.

VII. CONCLUSIONS AND FUTURE DIRECTIONS

Presented in details in this article, techniques, methods, and mechanisms in to implement, test, and evaluate the protective approach that proposed to prevent the SQLIA in stored procedures. As well as, we discussed the results of the proposed approach regarding, effectiveness of the proposed approach. On the other hand, we discussed the result of the detection and prevention approaches to address SQLIA. From the implementation and testing standpoints, we designed our experiment that has two scenarios (injected websites) that are presented a student information system (SIS). As well as, explained in detailed all steps to implement our approach that has three protective mechanisms to prevent the injection attacks. As we showed in Figure 1. From the evaluation standpoint, we evaluated the effectiveness of our approach by using different criteria. As well as, from the results, discussion standpoint, we presented and listed all the results values of the testing and evaluation of the protective approach that are explained and summarized in Tables 1 and 2 in Section 5 regarding the effectiveness of the proposed approach. The effectiveness evaluation of the proposed approach is based on, the capability to prevent both of bypass attack; call stored procedures (error messages) SQLIA, as well. It is capable of preventing the injection attack of the SQL server system commands via creating stored procedures. Our approach is effective to prevent the SQLIA in stored procedures that used many preventive mechanisms or methods to implement the protective of the attacks such as parameterized stored procedures, customized error messages, and encryption stored procedures in SQL server. We found several significant trends in our evaluation results. Our protective approach can prevent all the target attacks that are affected stored procedures SQLIA effectively. As a part of our future work, we will focus on two goals for the future scope. The first goal is to further improve the performance of the proposed approach. To this end, we plan to extend it to encompass both of SQLI and XSS attacks in the web applications. The second goal is to ensure the effectiveness of the conducted mechanisms. To this end, we will develop it to perform the detection and prevention of SQLI and XSS stored procedures.

REFERENCES


XVII. Gadhgil, S. SQL Injection Prevention in Banking.
