SQL Injection Remediation
IN618 Security

Introduction

In this lab we will take an application with SQL injection vulnerabilities and modify it to remove them. From the `I:` drive, download and extract the file `SQLInjectionRemediation.zip`. It contains a Visual Studio project that we will use for this exercise.

Recall that there are four things we identified last time that reduce vulnerability to SQL injections:

1. sanitise inputs;
2. use parameterised queries;
3. set correct DBMS permissions;
4. control unecessary error message output.

Of these, we will focus on the first two. Our SQLite database doesn't have a strong permissions model and our error messages would be handled when we deploy a production version of our application.

1 Verify the vulnerability

This application has the same SQL injection vulnerability that the example application we saw last time does. We can verify this by insepcting the source code and by trying the exploit. The easiet way to exploit this application is by inputting your attack code in the "Last Name" text box and performing a search.

2 Sanitise your inputs

Initially, the application makes no effort to sanitise its inputs before using them in database queries. Sanitising inputs for SQL is a bit hard, but there are some things we can do:

- Limit the length of input strings to appropriate values. You can use the C# `Substring` method for this.
- Remove inappropriate characters from the inputs. You can use the C# `Remove` or `Replace` methods for this.

3 Parameterised queries

Last time we discussed parameterised queries. Here is an example of a parameterised query prepared for SQLite in C#:

```
SQLiteCommand command = new SQLiteCommand(db_connection);
string sql = "select * from items where item_type=@type";
command.CommandText = sql;
command.Prepare();
command.Parameters.AddWithValue("@type", "B");
SQLiteDataReader reader = command.ExecuteReader();
```

This code will find items whose `type` is “`B`”.

Using this technique, modify your application to use parameterised queries.

4 Submission

Commit your modified code to your SVN repository before the next class meeting. To show that you are using SVN properly, your commit history for this exercise should have at least three commits: one when you begin work before modifying any code, one when you have added input sanitisation, and one when you add parameterised queries.