



**POLITECHNIKA
RZESZOWSKA**
im. IGNACEGO ŁUKASIEWICZA

Aplikacje internetowe

Projekt

Temat: Rezerwacja apartamentu

Wykonał:
Marcin Magryś

1. Cel projektu

Celem projektu było stworzenie aplikacji webowej pozwalającej na rezerwacje apartamentów.

2. Funkcjonalności

- Właściciele i klienci rejestrują się na stronie podając swoje dane: email, login, hasło.
- Właściciele mogą dodawać na stronie swoje pokoje, apartamenty, wprowadzając ich opis, adres, ilość miejsc, lokalizację, wyposażenie.
- Klienci mogą je rezerwować na określony czas.
- Klienci mają dostęp do swoich rezerwacji, gdzie mogą dodawać komentarze na temat pobytu.

Strona została wykonana w technologii ASP.NET w języku C#. Baza danych została wygenerowana z wykorzystaniem Microsoft SQL Server.

3. Przedstawienie problemu SQL Injection

SQL Injection jest jedną z najbardziej znanych i rozpowszechnionych przypadłości wielu aplikacji webowych korzystających z bazy SQL. Polega to na nieautoryzowanym dostępie nieuprawnionego użytkownika poprzez wprowadzenie do formularza na stronie niebezpiecznego fragmentu zapytania SQL. Przyczyną takiego niechcianego dostępu jest zwykle brak odpowiedniej walidacji wprowadzanych treści przez użytkowników. Jeżeli wpisywane na stronie dane przekazywane są bezpośrednio do zapytań może to skutkować poważnymi konsekwencjami:

- nieautoryzowany dostęp w trybie odczytu lub zapisu do całej bazy danych,
- możliwość ominięcia mechanizmu uwierzytelniania,
- możliwość odczytania wybranych plików,
- możliwość tworzenia plików w systemie na którym pracuje baza,
- możliwość wykonania kodu w systemie operacyjnym.

Należy w tym celu dokładnie sprawdzać jakiego rodzaju znaki są wprowadzane w formularzach. Bardzo prostym sposobem na włamanie się do naszej bazy jest użycie apostrofów lub myślników, które pozwalają na przekazywanie całych gotowych zapytań do naszej bazy. Jest to sytuacja

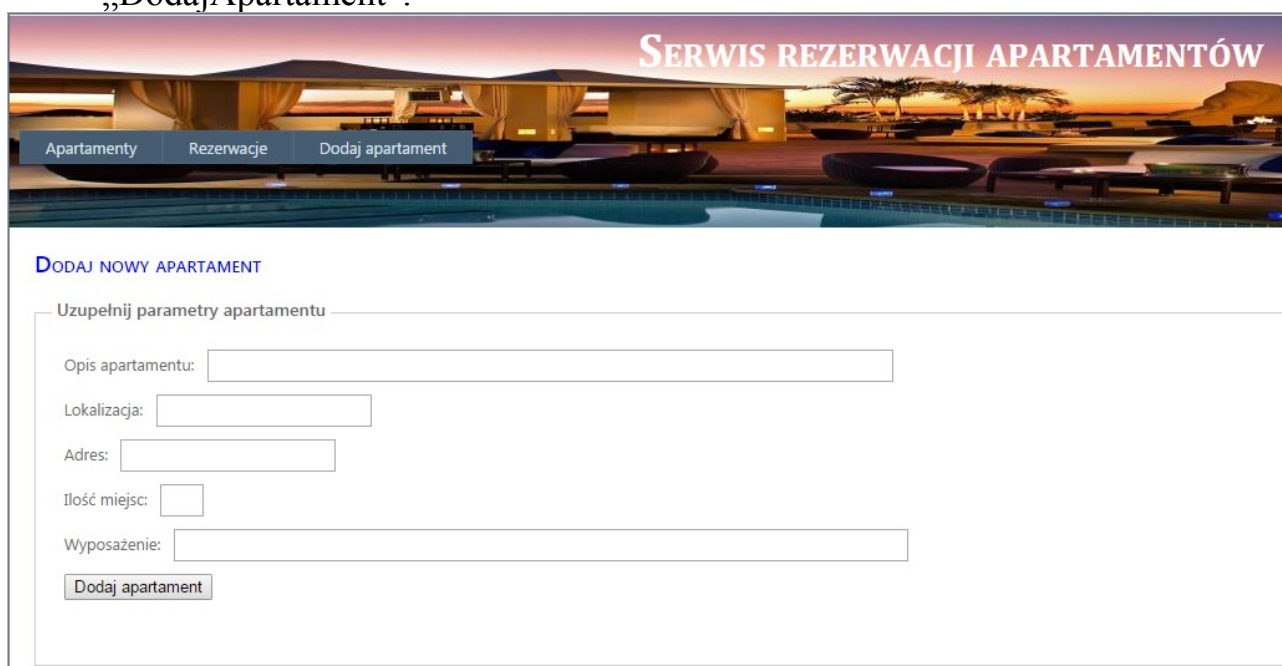
skrajnie niebezpieczna ponieważ atakujący ma otwartą drogę do przechowywanych danych. Dobrze zabezpieczona aplikacja to taka, która nie pozwala na używanie w formularzach dowolnych znaków bądź też znaki niebezpieczne odpowiednio konwertuje, tak by nie mogły one wyrządzić szkód w naszej aplikacji.

Przykłady metod zabezpieczenia przed SQL Injection:

- walidacja poprawności typu danych wejściowych,
- użycie funkcji filtrującej wybrane znaki pól tekstowych,
- użycie zapytań parametryzowanych,
- ograniczenie uprawnień użytkowników DB danej aplikacji webowej,
- użycie procedur składowanych

4. Ochrona formularzy przed SQL Injection

W projekcie, który realizuje został utworzony formularz o nazwie „DodajApartament”.



The screenshot displays a web application for apartment reservations. The header features the title "SERWIS REZERWACJI APARTAMENTÓW" and three navigation buttons: "Apartamenty", "Rezerwacje", and "Dodaj apartament". The main content area is titled "DODAJ NOWY APARTAMENT" and contains a form with the following fields:

- Opis apartamentu: [text input]
- Lokalizacja: [text input]
- Adres: [text input]
- Ilość miejsc: [text input]
- Wypożyczenie: [text input]

A "Dodaj apartament" button is located at the bottom of the form.

Zawarte w nim pola pozwalają na dodawanie parametrów apartamentu takich jak opis, lokalizacja, adres, ilość miejsc, wyposażenie. Po naciśnięciu przycisku „Dodaj apartament” dane te przesyłane są do bazy danych. Za przesyłanie tych danych odpowiadają linie kodu zawarte poniżej:

```

protected void Button1_Click(object sender, EventArgs e)
{
    Session["ID"] = User.Identity.Name;
    String query = "insert into Apartament(Opis,Lokalizacja,Adres,Ilość_miejsc,Login,Wyposazenie) values(@Opis,@Lokalizacja,@Adres,@Ilość_miejsc,@Login,@Wyposazenie)";
    SqlCommand cmd = new SqlCommand(query, bazaAI);
    cmd.Parameters.AddWithValue("@Opis", Opis.Text );
    cmd.Parameters.AddWithValue("@Lokalizacja", Lokalizacja.Text);
    cmd.Parameters.AddWithValue("@Adres", Adres.Text);
    cmd.Parameters.AddWithValue("@Ilość_miejsc", IloscMiejsc.Text );
    cmd.Parameters.AddWithValue("@Login", Session["ID"] );
    cmd.Parameters.AddWithValue("@Wyposazenie", Wyposazenie.Text );
    bazaAI.Open();
    cmd.ExecuteNonQuery();
    bazaAI.Close();
    Page.Response.Redirect(HttpContext.Current.Request.Url.ToString(), true);
}

```

W obsłudze przycisku tworzone jest zapytanie sql pozwalające przesłać wprowadzone przez nas dane do bazy. Jak wspomniano wcześniej jest to dość kluczowy moment ze względów bezpieczeństwa naszej aplikacji. W celu ochrony danych zawartych w bazie wykorzystano parametryzowane zapytania. Dane nie są wprowadzone bezpośrednio w zapytaniu sql lecz przekazywane za pomocą nazwanych parametrów poprzedzonych znakiem „@”. Taki zapis pozwala uchronić przed niechcianą modyfikacją naszego zapytania sql. Kolejnym sposobem na częściowe powstrzymanie ataków jest ograniczenie uprawnień użytkowników do minimum pozwalającego wykorzystywać stworzone dla nich funkcjonalności. W przypadku tej aplikacji dane do formularza mogą wprowadzać jedynie użytkownicy zalogowani jako „Wlasciciele”. Treści wprowadzane do kontrolki powinny być także wstępnie walidowane. W tym celu dla jednego z textbox-ów oznaczonego jako „Opis” dodano kontrolkę RegularExpressionValidator. Pozwala ona filtrować wprowadzane znaki, określać ich ilość, kolejność wystąpień, format, itd.

Dzięki użyciu parametru **ValidationExpression="[0-9a-zA-Ząęłńóśż\.,;\s]*"** w polu opis apartamentu można wpisywać tylko liczby, litery – łącznie z polskimi znakami, kropki, przecinki, myślniki, dwukropki, oraz spacje. Potencjalny atakujący nie będzie miał już możliwości wprowadzać znaków specjalnych, które mogłyby pomóc mu w przekazaniu do naszej bazy własnego zapytania.