

Projektdokumentation

Maximilian, Lenny und Tyrone

Sicherheitsvorkehrungen

SQL Injection

Funktion: SQL-Injection-Schutz verhindert, dass Angreifer schädliche SQL-Befehle durch Benutzereingaben in eine Anwendung einschleusen können, die dann vom Datenbankserver ausgeführt werden. Dies wird in der Regel durch die Verwendung von Prepared Statements und ORM-Tools erreicht, die sicherstellen, dass Benutzereingaben ordnungsgemäß behandelt und nicht als Teil des SQL-Codes interpretiert werden.

Wie es in unserem Projekt eingebaut ist:

```
1 Verweis | tabnine: test | explain | document | ask
private static bool VerifyPassword(string password, string storedPasswordHash)
{
    using (var sha256 = SHA256.Create())
    {
        var hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
        var hashedPassword = BitConverter.ToString(hashedBytes).Replace("-", "").ToLowerInvariant();
        return hashedPassword == storedPasswordHash;
    }
}
```

Hier haben wir es so gelöst das es mit Logik in der Konvertierung zu SQL nicht gehen kann das eine injection möglich ist.

Wir haben dies getestet und ohne Password kann ein Benutzer nicht in sein Konto rein.

Input Validerung

Funktion: Die Funktion der Input Validierung besteht darin, sicherzustellen, dass alle eingehenden Daten (aus Formularen, URL-Parametern usw.) den erwarteten Typ, das Format und den Wertebereich aufweisen, bevor sie von der Anwendung weiterverarbeitet werden. Dies schützt die Anwendung vor verschiedenen Arten von Eingabe-basierten Angriffen, indem nur korrekte und sichere Daten akzeptiert werden.

Wie es in unserem Projekt eingebaut ist:

```
document.addEventListener('DOMContentLoaded', function () {  
  // Validierungsfunktion für Benutzernamen und Passwort  
  function validateInput(event) {  
    const input = event.target;  
    const validValue = input.value.replace(/[^a-zA-Z0-9]/g, '');  
    if (input.value !== validValue) {  
      input.value = validValue;  
      alert('Nur Buchstaben und Zahlen sind erlaubt.');    }  
  }  
  
  // Event-Listener für die Eingabefelder  
  const usernameInput = document.getElementById('username');  
  const passwordInput = document.getElementById('password');  
  
  usernameInput.addEventListener('input', validateInput);  
  passwordInput.addEventListener('input', validateInput);  
});
```

Wir haben das in unserem ersten Projekt so gelöst das wir nur kleine und grosse Buchstaben von a bis z eingeben können und Zahlen von 0 bis 9.

Wenn jedoch ein anderes Zeichen eingeben wird kommt dieser Fehler auf der Webseite

Dies kann man noch beliebig erweitern um zum Beispiel maximale Zeichen zu erlauben.

ild-Hir

Auf 127.0.0.1:5500 wird Folgendes angezeigt:

Nur Buchstaben und Zahlen sind erlaubt.

Ok

Benutzername

Passwort

Login

Sessions management

Funktion: Sessions Management ermöglicht es einer Webanwendung, den Zustand eines Benutzers über mehrere HTTP-Anfragen hinweg zu verfolgen, indem eine eindeutige Session-ID verwendet wird. Sicheres Sessions Management beinhaltet die Erzeugung von zufälligen, schwer vorhersehbaren Session-IDs und deren sichere Handhabung, um unbefugten Zugriff auf Benutzersessions zu verhindern.

Wie es in unserem Projekt eingebaut ist:

```
namespace Apptest
{
    1 Verweis
    public class SessionManager
    {
        3 Verweise
        private DateTime sessionStart;
        1 Verweis
        private const int idleTimeout = 10; // Idle Timeout in Sekunden für das Be

        0 Verweise | tabnine: test | explain | document | ask
        public SessionManager()
        {
            sessionStart = DateTime.Now;
        }

        0 Verweise | tabnine: test | explain | document | ask
        public void RefreshSession()
        {
            sessionStart = DateTime.Now;
        }

        1 Verweis | tabnine: test | explain | document | ask
        public bool IsSessionActive()
        {
            return (DateTime.Now - sessionStart).TotalSeconds < idleTimeout;
        }

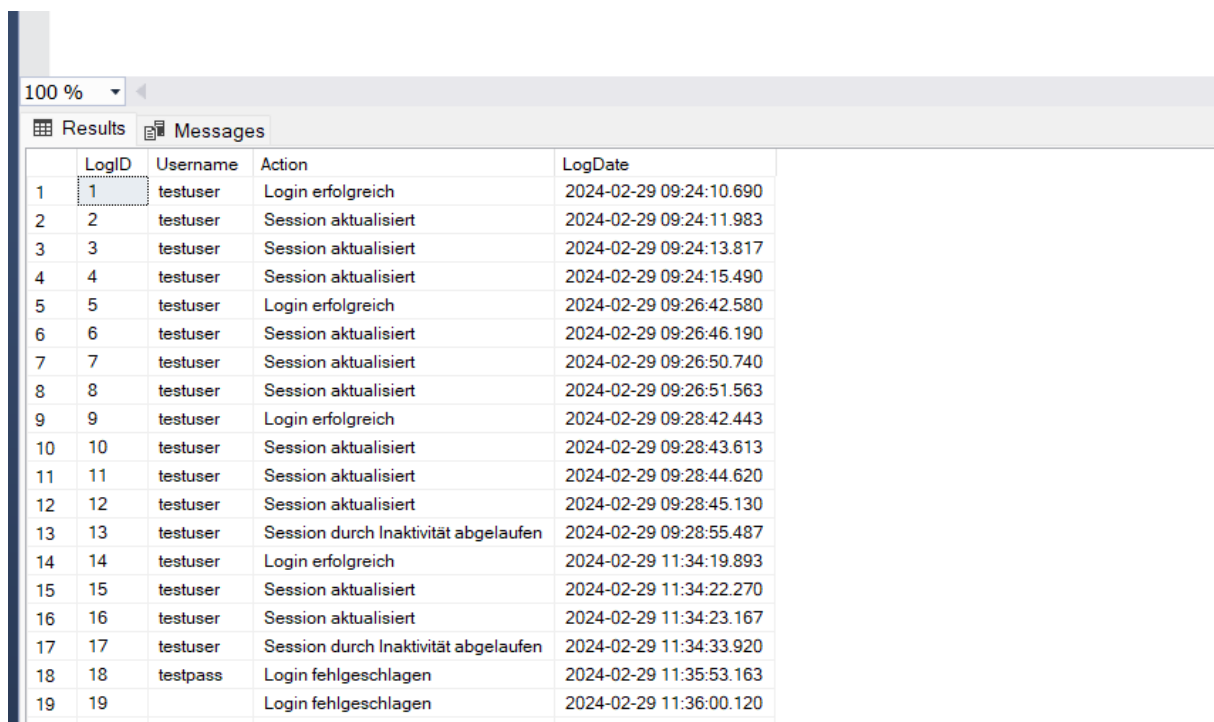
        0 Verweise | tabnine: test | explain | document | ask
        public void CheckIdleTimeout()
        {
            if (!IsSessionActive())
            {
                Console.WriteLine("Session durch Inaktivität abgelaufen.");
                Environment.Exit(0); // Beendet die Anwendung
            }
        }
    }
}
```

Wir haben es so implementiert das wir nach einer bestimmten Zeit die Session abbrechen und der Benutzer sich neu einloggen muss.

Man kann das auch im Applikationslog sehen das es eine Ausgabe gibt so bald ein Benutzer ausgeloggt wird

Dies ist eine einfacher Implementierung um zu zeigen wie ein Session Manager funktioniert .

Erweitern könnte man das auch je nach Bedürfnisse von den APP.



The screenshot shows a database application window with a zoom level of 100%. It contains two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following columns: LogID, Username, Action, and LogDate. The table contains 19 rows of data, showing a sequence of login and session updates for 'testuser' and 'testpass'.

	LogID	Username	Action	LogDate
1	1	testuser	Login erfolgreich	2024-02-29 09:24:10.690
2	2	testuser	Session aktualisiert	2024-02-29 09:24:11.983
3	3	testuser	Session aktualisiert	2024-02-29 09:24:13.817
4	4	testuser	Session aktualisiert	2024-02-29 09:24:15.490
5	5	testuser	Login erfolgreich	2024-02-29 09:26:42.580
6	6	testuser	Session aktualisiert	2024-02-29 09:26:46.190
7	7	testuser	Session aktualisiert	2024-02-29 09:26:50.740
8	8	testuser	Session aktualisiert	2024-02-29 09:26:51.563
9	9	testuser	Login erfolgreich	2024-02-29 09:28:42.443
10	10	testuser	Session aktualisiert	2024-02-29 09:28:43.613
11	11	testuser	Session aktualisiert	2024-02-29 09:28:44.620
12	12	testuser	Session aktualisiert	2024-02-29 09:28:45.130
13	13	testuser	Session durch Inaktivität abgelaufen	2024-02-29 09:28:55.487
14	14	testuser	Login erfolgreich	2024-02-29 11:34:19.893
15	15	testuser	Session aktualisiert	2024-02-29 11:34:22.270
16	16	testuser	Session aktualisiert	2024-02-29 11:34:23.167
17	17	testuser	Session durch Inaktivität abgelaufen	2024-02-29 11:34:33.920
18	18	testpass	Login fehlgeschlagen	2024-02-29 11:35:53.163
19	19		Login fehlgeschlagen	2024-02-29 11:36:00.120

Hier sehen wie wir es in unsere Datenbank gespeichert mit allen Informationen

Applikationslog

Funktion: Das Applikationslog zeichnet detaillierte Informationen über das Verhalten der Anwendung, Benutzeraktionen, Systemfehler und Sicherheitsereignisse auf. Es dient dazu, die Nachvollziehbarkeit von Aktionen innerhalb der Anwendung zu gewährleisten und unterstützt die Analyse und Behebung von Problemen sowie die Erkennung von Sicherheitsverletzungen.

Wie es in unserem Projekt eingebaut ist:

```
namespace Apptest
{
    6 Verweise
    public static class Logger
    {
        1 Verweis
        private static string connectionString = "Server=localhost; Database=BaselCoinDB; Trusted_Connection=True;";

        6 Verweise | tabnine: test | explain | document | ask
        public static void Log(string username, string action)
        {
            using (var connection = new SqlConnection(connectionString))
            {
                var query = "INSERT INTO UserLogs (Username, Action) VALUES (@Username, @Action)";
                using (var command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Username", username);
                    command.Parameters.AddWithValue("@Action", action);

                    connection.Open();
                    command.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Wir haben es so gelöst das so bald ein Benutzer sind einloggt es in der Datenbank mit Uhrzeit und Datum plus was passiert es gespeichert wird.

Das hat den Vorteil wie oben in den Funktionen erklärt, wir können herausfinden falls es ein Fehler gibt das passiert ist und wie wir es lösen können.