Modul 183 Projektarbeit

[Firmenname] | [Firmenadresse]

Ismail kassem, brian rueegg

[Jahr]

Inhalt

[Analyse der Aufgabenstellung 2](#_Toc524702611)

[Sicherheitsprobleme, Schwachstellen 2](#_Toc524702612)

[gefundene Fehler klassifiziert 2](#_Toc524702613)

[Massnahmen zur Verbesserung erläutert 2](#_Toc524702614)

[Verbesserung implementiert 2](#_Toc524702615)

[Testsetting, Testprotokoll 2](#_Toc524702616)

[Erkenntnisse, Empfehlungen, Quellen 2](#_Toc524702617)

# Analyse der Aufgabenstellung

Eine bestehende Applikation soll vom Sicherheitsaspekt verbessert werden. Wir wählten eine ASP.Net-Webapplikation aus, welche wir im Modul 183 erstellt haben. Diese Applikation wird für die Terminverwaltung gebraucht.

# Sicherheitsprobleme, Schwachstellen

Es wäre gut möglich, dass die Applikationen Sicherheitslücken besitzt. Die Applikation hat eine Datenbankanbindung. Wir sahen durch die OWASP Top 10 Liste und erkannten dass Injections(besonders SQL Injections) und Session Managment ein grosses Problem sein könnten. Deshalb schauten wir uns die Artikel von OWASP in diesen zwei Themenbereichen noch genauer an.

# Gefundene Fehler klassifiziert

Wir haben festgestellt, dass es möglich für Benutzter in der Datenbank irgendein Befehl ausführen zu können. So kann z.B. eine ganze Tabelle gelöscht werden. Dies ist fatal fürs Programm. Die Wahrscheinlichkeit, dass jemand dies versucht ist auch sehr hoch, da dieser Exploit der bekannteste ist nach OWASP. Der Exploit ist eine SQL Injection. Es ist möglich den Query, welcher Daten eintragen soll, weitere beliebige Querys einzutragen. Wir konnten kein Fehler bei der Authentifikation finden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Impact | **Hoch** |  |  | SQL Injection bei Eingabefelder |
| **Mittel** |  |  |  |
| **Tief** |  |  |  |
|  |  | **Tief** | **Mittel** | **Hoch** |
|  |  | Likelihood | | |

* Dieses Problem muss unbedingt behoben werden.

# Massnahmen zur Verbesserung erläutert

Es gibt eine Methode nämlich «SqlParameterCollection.AddWithValue(String, Object)» mit dieser ist es möglich einen Query und die Übergabewerte miteinander zu kombinieren, ohne dass eine SQL-Injection möglich ist.

# Verbesserung implementiert

|  |  |
| --- | --- |
| Vorher | MySqlCommand cmd = mySqlConnection.CreateCommand();  cmd.CommandText = "INSERT INTO tbtermin (`terminId`, `terminSubject`, `StartDate`, `EndDate`) VALUES (NULL, " + subject + "," + startdate +"," + endDate +")"; |
| Nachher | MySqlCommand cmd = mySqlConnection.CreateCommand();  cmd.CommandText = "INSERT INTO tbtermin (`terminId`, `terminSubject`, `StartDate`, `EndDate`) VALUES (NULL, @subject, @startDate, @EndDate)";  cmd.Parameters.AddWithValue("@subject", subject);  cmd.Parameters.AddWithValue("@startDate", startdate);  cmd.Parameters.AddWithValue("@EndDate", endDate); |

# Testsetting, Testprotokoll

# Erkenntnisse, Empfehlungen, Quellen

Bei Querys soll nie eine einfache String Addition benutzt werden, sondern falls möglich die «SqlParameterCollection.AddWithValue(String, Object)»-Methode. Da die String Addition schnell zu einer Injection führen kann. Da die oben genannte Methode extra für SQL-Injection abfangen erstellt wurde sollte man immer diese Methode benutzten, falls keine Parameter vorhanden sind kann auf die Methode verzichtet werden, da sie die Performance negativ beeinflussen könnte.