SQL-Injection
Wie schütze ich meine Webseite vor Angriffen

Inhaltsverzeichnis

Einleitung
Grundlagen von Sql-Injection
Sicherheitslücken in Ihren Anwendungen
Allgemeine Abwehrmöglichkeiten
Konkrete Abwehrmethodiken
Zusammenfassung
Zukunftsausblick

Einleitung

Weltweit sind millionen von Webseiten online und täglich werden es mehr, das Internet ist im Wachstum wie eh und je. Gerade mobile Anwendungen drängen in den Vordergrund und wollen möglichst überall Zugriff auf online Dienste gewähren. Doch sind die heutigen Webseiten wirklich gegen Angriffe von außen gesichert? Wie gut sind Webseiten wirklich gesichert und wo sind die Sicherheitslücken? Diese Fragen sollen in diesem kleinen Artikel geklärt werden und natürlich auch die Gegenmaßnahmen die implementiert werden sollten und Webseiten noch sicherer zu gestalten.

Das Ziel eines Angreifers ist es sich Zugriff auf bestimmte abgesicherte Bereiche zu verschaffen oder einfach nur Schaden an einer Webseite oder dessen Server anzurichten.
Diese Angriffe erfolgen oft über die Datenbankserver, indem alle wichtigen Daten, hinterlegt sind. Bei der sogenannten SQL-Injection versucht der Angreifer Standard T-SQL Abfragen, die im Hintergrund einer nahezu jeder Webseite laufen, zu seinen Gunsten zu verändern, zusätzliche Daten einzuschleusen oder auszulesen. Ein häufig vorkommendes Beispiel dafür wäre ein Userlogin mit Username und Passwort.
Gerade hier ist es für den Angreifer lohnenswert das Login zu knacken in dem er das dahinterliegende Sql-Statement dahingehend manipuliert, dass dieser Zugriff auf die gesicherten Seiten hat, ohne sich jedoch angemeldet zu haben oder gar als ein dritter angemeldet ist.

Voraussetzung für SQL-Injection ist lediglich ein Browser und ein gewisses Grundwissen in der T-SQL Syntax. Ob der Angreifer erfolgreich ist oder nicht liegt alleine in den Händen der Entwickler der Webseite und wie sicher diese Ihre Anwendung gestalten

Hintergrund SQL

Die SQL Sprache (Structured Query Language) ist eine Datenabfragesprache für relationale Datenbanken. Der bekannteste Standard dieser Sprache ist die SQL-92 ANSI Standardisierung. Um diese Abfragen durchzuführen werden diese normalerweise in sogenannten Queries ausgeführt, welches dann bestimmte Daten in Form einer Tabelle zurückgibt. In dem folgenden Bericht wird speziell auf Transact-SQL bezogen welche im Microsoft SQL Server und anderen bekannten Datenbankmanagementsystemen benutzt wird.

Grundlagen von Sql-Injection

Angriffe auf Webseiten können speziell bei Sql-Injection auf mehrere Arten realisiert werden. Im folgenden Text wird die grundlegende Vorgehensweise in einer SQL-Injection erläutert.

Ein typisches Sql-Statement sieht wie folgt aus:

SELECT id, vorname, nachname FROM users

Auf unser oben genanntes Beispiel angelehnt würden also die Abfragen und der Vergleich des Users und seines Passwortes folgendermaßen aussehen:

SELECT id, vorname, nachname FROM users WHERE vorname = 'Sepp' and nachname = 'meier'

Der Angreifer hätte jetzt folgende Möglichkeiten dieses Statement zu seinen Gunsten zu verändern und sein eigenes Sql-Statement durchzuführen:
Meldet sich der User mit folgenden Einträgen an:

Username: Se'pp
Passwort: 123qwe@

kommt es zu einem Fehler in der Datenbank (solange dieser nicht explizit abgefangen wird):

Server: Msg 170, Level 15, State 1, Line 1
Line 1: Incorrect syntax near 'pp'.

Grund dafür ist, dass der User/Angreifer ein Hochkomma eingefügt hat, was in SQL Syntax als Ende des String oder Varchar Wertes gilt. Der SQL Server hat jetzt also versucht 'pp' auszuführen, was natürlich zu einem Fehler führt. Folgendes Statement kommt folglich im SQL Server an:

SELECT id, vorname, nachname FROM users WHERE vorname = 'Se'

Der Angreifer hat jetzt die Macht jedes beliebige Statement auf die Datenbank loszulassen:

Username: Se'; DROP table users --
Passwort: 123qwe@

Folgendes Statement kommt in der Datenbank an:

SELECT id, vorname, nachname FROM users WHERE vorname = 'Se'; DROP table users --
Hinweis: -- sind Kommentar Zeichen in SQL alles was darauf folgt wird ignoriert

Man sollte denken, dass solche Eingaben in eine TextBox leicht abgefangen werden können. Das ist jedoch nicht ganz ohne und außerdem könnte der Angreifer auf ganz anderem Wege Zugriff auf geheime Daten erlangen. Viele Suchfunktionen verwenden folgende Abfrage im Hintergrund:

SELECT Id, Trainingsname, Trainingsdatum FROM Schulungen WHERE id= 1234

Hier wäre es dem Angreifer möglich an das oben stehende Statement seine gewünschte Angabe anzuhängen. Zumeist wird die Id für solche Suchfunktionen über den Querystring übergeben, was ein schwerwiegender Fehler ist, also eine weitere Sicherheitslücke.

Doch das sind nicht die einzigen Probleme. Was passiert, wenn andere SQL Varianten genutzt werden. Jede SQL Variante hat ihre Eigenheiten und damit auch Ihre jeweilige Sicherheitslücken. Betrachtet man sich Access und die zugehörige Microsoft Jet DBMS Engine, so könnten hier Datumswerte mit '#' beeinflusst und als "Escape Sequenz" genutzt werden. Somit also wieder das eigene SQL-Statement "injiziert" und durchgeführt werden.

Sicherheitslücken in Ihren Anwendungen

Viele Sicherheitslücken können schon bei Datenbankzugriffen verhindert werden.
Standardmäßig werden Datenbankzugriffe mit ADO.NET 2.0 in Visual Studio 2005 programmiert.
Im schlechtesten Fall steht in der Anwendung folgender Code:

```
Dim conn as new sqlconnection("Data Source=local;Initial Catalog=Northwind;User
Id=myUsername;Password=myPassword; ")
Dim cmd as new SqlCommand("Select Id, Username, Password  from Users where username= '" +
username + "' and password = '" + password +"'",conn)
Try
conn.open
Dim da as new DataAdapter(cmd)
Dim ds as new Dataset
Da.fill(ds)
GridView1.DataSource = ds.tables(o)
GridView1.DataBind()
Catch ex as exception
Finally
Conn.close
End Try
```

Es handelt sich hierbei um sehr unsicheren Datenbank Zugriffscode, denn das SQL- Statement kann wieder über die „richtige" Eingabe beeinflusst werden.
Der Angreifer schreibt folgendes in die Anmeldung:
Username: admin '--
Username: ' or 1=1--
Username: ' union select 1, 'username', 'password', 1 –
Das Statement wird damit wie folgt abgekürzt:

```
SELECT * from users where Username= admin --
```

...und schon hat jemand Zugriff auf das Administrator Konto...

```
SELECT * from users where Username= '' or 1=1
```

...oder einfach nur Zugriff auf unsere gesicherten Daten

Das ist natürlich nur die Spitze des Eisberges es können auch andere Szenarien eintreten.
Beispielsweise Abfragen die über den Querystring gesendet werden! Schauen wir uns einmal folgende URL an:

http://www.meinewebseite.com/search.aspx?id=10

Die oben angegebene Id wird im Programmcode in die SQL Suchabfrage mit eingebaut, sodass dann alle Einträge mit der Id 10 geladen und dargestellt werden.
Auch hier können jederzeit die oben genannten Tricks angewendet werden. Jedoch sind das nicht nur Select-Statements sondern auch Insert-Statement wie folgendes Beispiel zeigen soll:

```
http://www.meinewebseite.com/search.aspx?id=10; INSERT INTO 'admin_login'
('id,'username','password','beschreibung') VALUES
(666,'meinName','meinPasswort','NA')—
```

Jetzt hat sich der Angreifer ganz einfach einen kompletten Zugriff zum SQL Server ermöglicht!

Die bisher größte Sicherheitslücke, ist die SQL Server interne Kommando Konsole die „xp_cmdshell", wer diese aktiviert erlaubt dem Angreifer diese zu nutzen und somit noch mehr Schaden anzurichten

Um beim Beispiel des Suchquerystrings zu bleiben kann der Angreifer wie folgt auf die Konsole des Sql Servers zugreifen:

http://www.meinewebseite.com/search.aspx?id=10; `exec master..xp_cmdshell 'format C:\'`

Nun hat der Angreifer freie Hand und kann die komplette Datenbank unter seine Gewalt bringen.

Natürlich haben sich einige Leute darüber den Kopf zerbrochen wie der ansteigenden Flut von Angriffen jeglicher Art auf Webseiten getrotzt werden kann und deshalb befasst sich das nächste Kapitel speziell mit den Abwehrmöglichkeiten solcher Angriffe.

Allgemeine Abwehrmöglichkeiten

Man kann sich in vielerlei Hinsicht vor Angriffen speziell im Sql-Injection Bereich schützen. Als erstes sollte man das Prinzip das Microsoft vorlegt "Savety First" beibehalten und die Eigenschaften im Sql Server mit Bedacht setzen (z.B.: Enable cmdshell). Dieses Prinzip besagt, dass alle kritischen Einstellungen die Sicherheitslücken im SQL Server darstellen standardmäßig auf deaktiviert gesetzt werden, um die Nutzer besser schützen zu können.

Weiterhin werden im ASP.NET gewisse Einträge (je nach Einstellung natürlich) aus Textboxen ausgefiltert und abgeblockt wie zum Beispiel das <script> Tag.
Viele Probleme liegen auch in der letztendlichen Implementierung des Programmierers wie im oben stehenden Codebeispiel zusehen ist. Hier wird gezeigt wie man dies umgehen bzw. besser machen kann:

```
Dim conn as new sqlconnection("Data Source=local;Initial Catalog=Northwind;
User Id=myUsername;Password=myPassword; ")
Dim cmd das new sqlCommand("Select Id, Username, Password  from Users where username=
@username and password = @password",conn)
cmd.parameters.addwithvalue("@username", username)
cmd.parameters.addwithvalue("@password", password)
Try
conn.open
Dim da as new DataAdapter(cmd)
Dim ds as new Dataset
Da.fill(ds)
GridView1.DataSource = ds.tables(o)
GridView1.DataBind()
Catch ex as exception
Finally
Conn.close
End Try
```

Mit Hilfe der Parameter lässt sich verhindern, dass bestimmte Sql-Statement verkürzt ober abgeändert werden können, da die Parameter Klasse dies verhindert.
Da die Parameter in speziellen Klassen untergebracht sind können diese nicht durch, die vom Angreifer gewollten, Werte ausgetauscht werden und überprüfen weiterhin ob das Statement unerlaubte Zeichen enthält.

Zu guter letzt wäre da noch der Querystring der zu überprüfen ist, denn auch dort können sich schädliche Einträge einschleichen.
Abschließend sollte man darauf achten, dass Fehlermeldungen, die evtl. durch Falscheingabe erzeugt werden, nicht Aufschluss darüber geben wie die Datenbank oder Tabelle zusammengesetzt ist. Kurzum keine detaillierten Informationen in Fehlermeldungen ausgeben.

Konkrete Abwehrmethodiken

Was gibt es für Möglichkeiten SQL-Injection zu verhindern oder zumindest erschweren? Man könnte die Eingaben via Regular Expression überprüfen. Es handelt sich hierbei um Ausdrücke die Texte nach bestimmten Merkmalen durchsuchen. Folgender Regular Expression Ausdruck beispielsweise

s/[^0-9a-zA-Z]//\

erlaubt lediglich die Eingabe von Ziffern und Buchstaben. Dieser Ausdruck lässt sich ganz einfach über einen RegularExpressionValidator im Visual Studio 2005 einsetzen.

Es lässt sich ebenfalls einige Angriffe ganz einfach abblocken, wenn der SQL Server richtig konfiguriert ist. Standardmäßig setzt Microsoft hier richtig an und setzt alle „gefährlichen" Einstellungen auf den deactivated Status. Um den SQL Server möglichst sicher zu konfigurieren sollte man diesem Beispiel treu bleiben und beispielsweise die cmd_shell deaktiviert lassen.

Ein weiterer Schritt ist, dem User, der auf die Datenbestände aus der Anwendung zugreift, lediglich die Berechtigungen zu geben, die er tatsächlich für die Anwendung benötigt und vor allem die Zugriffe auf Systemdatenbanken zu sperren. Der sql-user, der die benutzeraccounts ausliest und passwörter überpüft benötigt beispielsweise keine Schreibberechtigung. Eine Injection, die ein Statement wie SELECT * FROM Users WHERE Username = ‚abc'; DROP Table Users würde dann fehlschlagen …

Zu guter letzt sollte dann noch die Eingaben des Querystring überprüft werden, die ebenfalls anhand des oben stehenden Regular Expression Ausdruckes überprüft werden können.

// du solltest den leser noch darauf hinweisen, dass sql injection mit parametern nicht mehr möglich ist. Aus meiner sicht ist es sinnvoller, beide „Kapitel" Abwerhmethoden zusammenzufassen.

Zusammenfassung

Beachtet man die oben genannten Tipps und achtet auf die richtigen Security Einstellungen im SQL Server so kann das Risiko eines erfolgreichenAngriffes drastisch minimiert werden. In Zukunft sollte das Thema SQL-Injection der Vergangenheit angehören, da sowohl Entwicklungsumgebungen als auch Datenbankserversysteme immer höhere Anforderungen an Sicherheitsfeatures stellen.

Autoreninfo

Armin Stockner ist Mitarbeiter der ppedv AG und als Trainer für den Bereich .NET tätig. Sein Wissen gibt er neben den Trainings auch als Sprecher der VSone oder der ASP konferenz weiter. Darüber hinaus versorgt er die VB-Magazin Community mit aktuellen Posts und Fachartikeln. Bei Fragen können Sie sich direkt an ArminStockner@hotmail.com wenden.