LU11a - SQLi Grundlagen

Lernziele

1. Erklären können was eine SQLi ist und welches grundlegende Ziel diese Attacke hat.
2. Ursachen nennen können, warum SQLi auch in der Gegenwart ernstgenommen werden muss.

Einleitung

Die *SQL-Injection* ist eine Technik, die sich auf der OWASP-Top10-Liste seit Jahren grösster Beliebtheit erfreuen darf. Die *Popularität* verdankt das verdankt der Tatsache Attacken auf schlecht gesicherte Seiten einfach zu ermöglichen.

Ein *Injection* bedeutet *Einspritzung*, bzw. *einschleusen*. Eine SQL-Injection startet also Angriffe auf Datenbanken mithilfe von SQL-Anweisungen mit dem Ziel unrechtmässig an den Datenbankinhalt zu gelangen.



Bekannte Attacken

Diese nachfolgenden Beispiele zeigen, dass auch professionell erstellte Webapplikationen anfällig auf SQLI-Attacken sind.

- Im Oktober 2014 wturde in den Medien über einen Datenverlust bei der *PSN* (Playstation-Network) berichtet. Hinter der Schlagzeile «SQL-Injektion: Sicherheitslücke erlaubt Zugriff auf Sony-Kundendaten …“ verbarg sich eine SQLI-Attacke. Die 7 Millionen Benutzerdaten, die seitens unberechtigten abgezogen wurden, führten zu einem massiven Imageverlust.
- Die Business-Socialmedia-Plattform *linkedin* hat auf die gleiche Art und Weise im Jahre 2012 6.5 Millionen Benutzerdaten verloren.
- Im Juni 2016 verlor die „University of Greenich“ 2.7 GB vertraulicher Benutzerdaten ihrer Studenten und Mitarbeitende.

Die eben aufgeführten Beispiele sind nur einige aus der lange Liste der *SQLI Hall-of-Shame*. Man kann also daraus folgern, dass die SQLI-Vulnerability (Verwundbarkeit) auch in der Gegenwart ernstgenommen werden muss.

Ursachensuche für SQLI-Verwundbarkeit

Obwohl mittlerweile fast schon ein *alter Hut*, muss man sich fragen warum unsere heutigen Applikationen nicht per Default gegen diese Attacken geschützt werden bzw. diese verunmöglichen. Nachfolgend finden Sie die wichtigsten Gründe.

Ausbildungsniveau der Webentwickler

Wenn man sich die Entwicklung der IT bzw. den Ausbildungstand der IT-Entwickler genauer ansieht, erscheint es diese Verwundbarkeit nicht mehr ganz so unerklärlich. Speziell in der Webentwicklung gibt es sehr viele Autodidakten, die sich das *Handwerk* selber beigebracht haben. Im Zentrum standen Lösungen, die unter Zeitdruck entwickelt wurden. Es ist verständlich, dass das Thema *Security*leicht übersehen wird, wenn enge Projektvorgaben wenig Spielraum für Sicherheitsvorkehrungen lassen.

Awareness

Der Mensch ist in seinem Wesen *notwendigkeitsorientiert*. D.h. er macht grundsätzlich nichts, was nicht dringend notwendig ist. Beispielsweise werden Kurven auf Strassen nur bei einer entsprechenden Opferzahl gekennzeichnet, sehr bitter aber wahr.

Risiken

Im Gegensatz zum Strassenbau ist aber in der Softwareentwicklung eine Anpassung wegen Sicherheitslücken nur mit sehr grossen Aufwand möglich ist. Bei jeder Anpassung werden zudem Risiken eingegangen neue Fehler einzubauen.

Je nachdem wie vital (Lebenswichtig) die Applikation für das eigene Business ist, will kein Entscheidungsträger das Risiko eingehen, dass die Kernapplikation nach der *Verbesserung* nur noch teileweise, gar nicht mehr funktioniert oder, noch schlimmer, unbemerkte Fehler aufweisen.

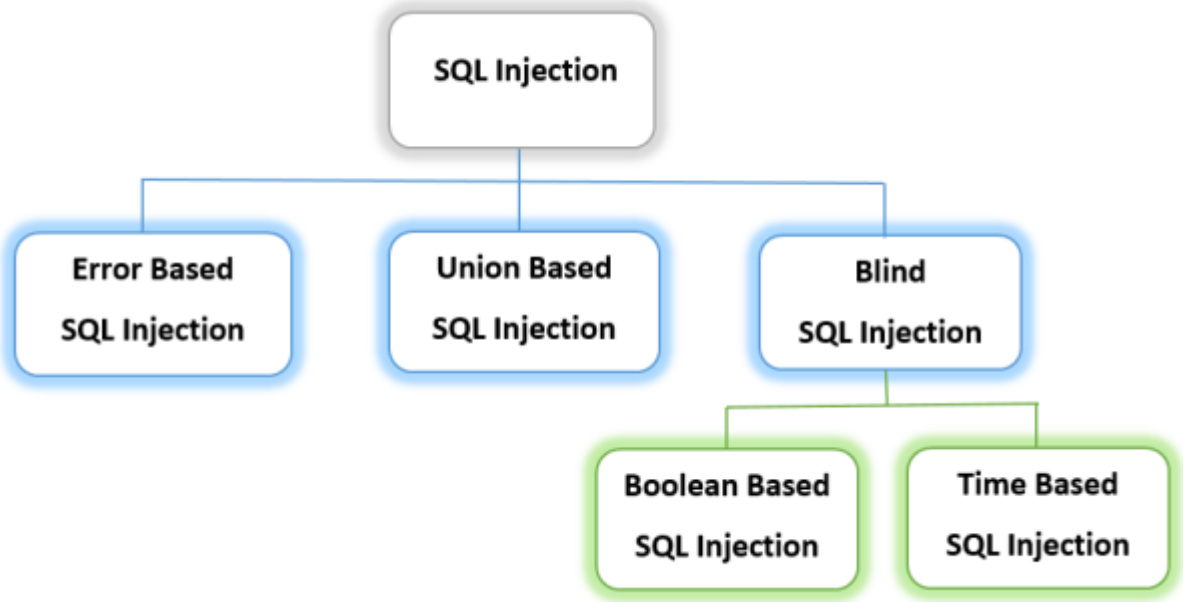
Kosten

Security-Massnahmen verteuern die Entwicklung. Bei knappen Projektbudgets wird teilweise auf Funktionen verzichtet, die der Kunde nicht unmittelbar bemerkt. Eine nachträgliche Anpassung der Applikation kann sehr teuer werden. Wenn nicht wirklich von vitaler Bedeutung, werden solche Investitionen vertagt oder gescheut.

Varianten

Das nachfolgende Schaubild zeigt die verschiedenen Varianten/Hauptarten von SQL-Injection-

Angriffen.



Man unterscheidet dabei mehrere Ansätze:

- **Error Based SQL Injection:** Hierbei nutzen Angreifer bewusst Fehlermeldungen der Datenbank. Aus den detaillierten Fehlermeldungen lassen sich oft interne Strukturen wie Tabellennamen oder Spalten ableiten.
- **Union Based SQL Injection:** Dieser Ansatz verwendet den UNION-Operator, um zusätzliche Abfragen an die Datenbank anzuhängen. Auf diese Weise lassen sich sensible Informationen wie Benutzer- oder Kreditkartendaten direkt im regulären Antwortformat der Anwendung ausgeben.
- **Blind SQL Injection:** Wenn die Anwendung keine sichtbaren Fehlermeldungen liefert, setzen Angreifer auf indirekte Methoden. Dabei gibt es zwei Varianten:
  - **Boolean Based SQL Injection:** Die Abfrage wird so verändert, dass sie unterschiedliche Antworten zurückliefert (z. B. wahr/falsch). Über diese Unterschiede lässt sich die Struktur der Datenbank Stück für Stück erschliessen.
  - **Time Based SQL Injection:** Hier wird die Antwortzeit der Datenbank ausgenutzt. Angreifer fügen Befehle ein, die eine zeitliche Verzögerung auslösen. Anhand der Antwortdauer können sie Rückschlüsse auf die Gültigkeit bestimmter Annahmen ziehen.

Quellennachweis

- PortSwigger-What is SQLi?

Volkan Demir

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m183/learningunits/lu11/01

Last update: **2026/09/16 11:17**