M151 Dokumentation

Inhalt

[Ziel der Website 2](#_Toc106010903)

[Meilensteine 2](#_Toc106010904)

[Technologien 2](#_Toc106010905)

[Thema Sicherheit 2](#_Toc106010906)

[Man in the middle (Man-in-the-Browser) 2](#_Toc106010907)

[Denial of Service, Distributed Denial of Service 2](#_Toc106010908)

[Cross-Site-Request-Forgery 2](#_Toc106010909)

[SQL-Injection 2](#_Toc106010910)

[Cross Site Scripting 2](#_Toc106010911)

[Session Hijacking 3](#_Toc106010912)

[Arbeitsjournal 3](#_Toc106010913)

[23.5.2022 3](#_Toc106010914)

[30.5.2022 3](#_Toc106010915)

[13.6.2022 3](#_Toc106010916)

# Ziel der Website

Das Ziel unsere Website ist es Bilder in eine Datenbank hochzuladen und dort zu speichern. Zudem sollte jeder User sein Persönliches Login haben.

# Meilensteine

* Registration
* Login
* Bilder können gespeichert werden
* Bilder abgerufen

# Technologien

Wir wollen für unsere Website HTML&CSS verwenden. Für die Logik unserer Seite nutzen wir PHP, da wir damit schon andere Webseiten gemacht haben und es die einzige Webtechnologie ist die wir in der Schule vertieft angeschaut haben. Als DBMS haben wir PhpMyAdmin mit MySQL.

# Thema Sicherheit

### Man in the middle (Man-in-the-Browser)

Um die Benutzer unserer Website gegen Man in the Middle zu schützen, hat jeder Benutzer sein eigenes Login. Um ganz sicher zu gehen kann man eine Zwei-Faktor-Authentifizierung verwenden. Noch sicherer wird es wenn der Benutzer all 3 Monate das Passwort ändern muss. Da unsere Website keine sehr privaten Daten enthalten verzichten wir auf die letzten 2 Punkte. Abgesehen vom Passwort verwenden wir https und nicht http damit alles verschlüsselt ist.

### Denial of Service, Distributed Denial of Service

Damit unser System nicht durch ein (Distributed) Denial of Service Angriff lahmgelegt wird, müssen wir unsere Website bei jemandem Hosten der eine Firewall zur Verfügung stellt und genug Leistung besitz damit kleinere Angriff keinen schaden anrichten.

### Cross-Site-Request-Forgery

Um sich gegen Cross-Site-Request-Forgery zu schützen benutzen wir ein Session-Token. Eine Zufalls Zahl mit mindestens 4-Stellen. Diese Zahl wird bei jedem http-Request aus dem Frontend zB. über ein verstecktes Feld mit gegeben und kann dann überprüft werden ob es immer noch die gleiche ist.

### SQL-Injection

Um einen solchen Angriff zu verhindern, sollen Sonderzeichen rausgefiltert werden. Zudem sollte man nur Anfragen erlauben die auch den gewünschten Datentyp liefern und nicht irgendeinen. Zusätzlich sollte man auf sogenannte Prepared-Statments (vereinfachts SQL für häufig gebrauchts SQL) nicht benutze da sie eine Sicherheitslücke sind. Auf Seiten des Datenbankmanagementsystem muss man schauen das es möglichst wenig Benutzerkonten hat und mit nur soviel rechte wie nötig. Das System muss man immer schön Updaten und nicht benutzte Dienste ausschalten.

### Cross Site Scripting

Wir sollten sicher darauf achten das nur Bilder in der Datenbank gespeicherte werden können und keine manipulierte Dateien die unsere Datenbank kaputt machen. Das gleiche gilt für die Registration, hier müssen wir schauen das der Benutzername oder Password nicht schädlicher JavaScript Code oder SQL enthält. Um solche Angriffe zu verhindern müssen wir Sonderzeichen bei Inputfelder verbieten oder wegschneiden zum Beispiel Semikolons zum Schluss oder ‘</>’ weg schneiden. Zusätzlich können wir Libaries verwenden, welche den Text noch genauer auf schädlichen Code überprüft. Das Bild sollten wir auf jeden Fall auf den Dateityp überprüfen und nur zulassen wenn das Dateiformat der Bilder JPEG oder PNG ist, damit kein schädliches Skript abgespeichert werden kann.

### Session Hijacking

Wir wollen auch die öffentlichen Daten, in unserem Fall die Bilder von den privaten Logindaten trennen. Damit im Fall eines Angriffs die privaten Daten nicht erreichbar sind.

# Arbeitsjournal

### 23.5.2022

### 30.5.2022

### 13.6.2022