CloneCademy

Qualitätssicherungsdokument

Gruppe:	Ilhan Simsiki <ilhan.simsiki@stud.tu-darmstadt.de> Leonhard Wiedmann <leonhard.wiedmann@stud.tu-darmstadt.de> Tobias Huber <tobias.huber@stud.tu-darmstadt.de> Claas Völcker <c.voelcker@stud.tu-darmstadt.de>
Teamleiter:	Alexander Nagl <alexander.nagl@t-online.de>
iGEM Team TU Darmstadt:	Thea Lotz <lotz@bio.tu-darmstadt.de> Fachbereich Biologie
Abgabedatum:	xx.xx.xxxx

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelor-Praktikum SoSe 2017
Fachbereich Informatik

Inhaltsverzeichnis

1 Einleitung

CloneCademy ist ein Projekt für das iGEM-Team der TU Darmstadt. Die *international Genetically Engineered Machine competition* (iGEM) ist ein internationaler Wettbewerb für Studierende auf dem Gebiet der Synthetischen Biologie. Dieser wird seit 2003 von der iGEM-Foundation veranstaltet.

Im Rahmen des iGem-Wettbewerbs sollen wir eine Online-Lernplattform für das iGem-Team der TU Darmstadt erstellen. Das Ziel dieser Plattform ist es, durch interaktive Unterrichtseinheiten Prinzipien der Molekularbiologie sowie der syntetischen Biologie zu erlernen, und sowohl eigene Lernfortschritte als auch die anderer Teams begutachten zu können. Darüber hinaus soll es auch anderen Interessierten (z.B. andere iGEM-Teams, Lehrende an Universitäten und Schulen, etc.) möglich sein, eigene Inhalte einzupflegen und zur Verfügung zu stellen.

2 Qualitätsziele

2.1 Datensicherheit (Security)

Im Rahmen des Projekts CloneCademy wird eine Webanwendung entwickelt, auf die über das Internet zugegriffen werden kann. Daher ist die Sicherung gegen unbefugten Zugriff und Änderung der Daten der Webseite in diesem Projekt das wichtigste Qualitätsziel. Da CloneCademy sowohl persönliche Daten der Nutzer*innen als auch Metadaten über die Nutzung der Plattform und die Inhalte der einzelnen Lerneinheiten in einer Datenbank speichert, ist es sehr wichtig, dass Internetnutzer*innen keine Daten verändern oder einsehen können, solange sie dazu nicht die benötigten Rechte besitzen.

Die größte Bedrohung geht von bekannten Webangriffen und Misskonfigurationen im Backend einer Webseite aus. Während dieses Projekts wird darauf geachtet, dass wir die Plattform gegen die wichtigsten Sicherheitslücken in einer Webanwendung sichern.

Diese sind[1]:

1. Möglichkeiten zur Ausführung fremden Codes (Injection & Cross-Site Scripting)
2. Fehler in Authentifizierung und Session-Management
3. Fehlerhafte Zugriffskontrolle
4. Sicherheitsrelevante Fehlkonfiguration
5. Verlust der Vertraulichkeit sensibler Daten
6. Nutzung von Komponenten mit bekannten Schwachstellen
7. Ungenügend geschützte Programmierschnittstelle

Um die Sicherheit der Anwendung zu gewährleisten, folgen wir einem mehrschrittigen Plan.

Ein Entwickler des Teams wird zum Sicherheitsbeauftragten ernannt. Seine Aufgabe ist es, auf die Einhaltung aller Programmierrichtlinien zu achten und Tests zur Validierung aller Schutzziele durchzuführen. Die genaue Vorgehensweise wird im Folgenden detailliert ausgeführt.

Während der ersten Gespräche mit den Auftraggeber*innen wurden Nutzerrollen definiert (Admin, Moderator*in und Nutzer*in). In jeder User Story wird festgehalten, ob und welche Zugriffsbeschränkungen einzelne Module der Webseite oder Daten besitzen. Während der Implementierung der jeweiligen Story achten die zuständigen Entwickler darauf, dass diese Vorgaben

[1] Paraphrasiert nach der verbreiteten Liste des Open Web Application Security Project (OWASP)
Link zum Download: https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf

eingehalten werden. Um dies zu gewährleisten, werden die zur Verfügung gestellten Sicherheitswerkzeuge der genutzten Frameworks (Django, Django REST, Angular2) verwendet.

Während der Implementierung achten alle Entwickler darauf, die Best Practices im Web Development einzuhalten. Als Referenz dafür werden die Handreichungen des Open Web Application Security Project (OWASP)[2] verwendet. In den wöchentlichen internen Code-Reviews wird die Qualität des Codes hinsichtlich dieser Richtlinien überprüft.

Um die Qualität des Codes im Python-basierten Backend automatisiert zu testen, wird das verbreitete Werkzeug *bandit*[3] verwendet. Als statisches Analyse-Werkzeug für das Frontend wird *ng2lint*[4] verwendet. Beide Programme unterstützen die Code-Review, indem die gelieferten Meldungen als Grundlage für eine detaillierte Analyse des Codes genutzt werden. Um eine dauerhaft hohe Qualität des Codes zu gewährleisten werden die Analyse-Tools jede zweite Iteration verwendet. Das Team achtet darauf, alle gefundenen Schwachstellen soweit wie möglich innerhalb der nächsten zwei Iterationen zu beheben und ansonsten eine realistische Einschätzung über die Konsequenzen der gefundenen Fehler an die Auftraggeber*innen zu melden.

Um die tatsächliche Sicherheit des Endproduktes gegen externe Angriffe zu zeigen, wird ein automatisiertes Penetrations-Werkzeug verwendet. Das OWASP empfiehlt hierfür das *Zed Attack Proxy Project*, welches eine Reihe bekannter Angriffe auf die Plattform ausführt und die gefundenen Schwachstellen meldet. Der Sicherheitsbeauftragte führt diese Tests monatlich durch und meldet die Ergebnisse an die jeweiligen Entwickler des betroffenen Moduls, damit diese innerhalb des nächsten Monats behoben werden können. Ist eine Behebung nicht möglich, werden die Konsequenzen dieser Lücke zusammen mit möglichen weiteren Schritten zum Schutz der Anwendung im Betrieb an die Auftraggeber*innen gemeldet.

2.2 Bedienbarkeit (Entwurf)

CloneCademy muss für jede Benutzerrolle (Admin, Moderator*in, Nutzer*in) bedienbar sein. Um dies zu gewährleisten, müssen alle zugängigen Seiten, vor allem das Frontend, intuitiv benutzbar gestaltet werden. Die grundlegende Bedienung der Webseite sollte ohne Einführung erkennbar sein und alle Elemente sollten mit einer entsprechenden Hilfestellung erweitert werden.

2.3 Veränderbarkeit (Entwurf)

Für die Webanwendung CloneCademy ist es wichtig, dass die Anwendung im Nachhinein noch veränderbar ist. Es muss möglich sein, neue Inhalte in die Datenbank einzupflegen und auch den Source Code der Webseite selbst erweitern zu können.

[2] https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf
[3] https://wiki.openstack.org/wiki/Security/Projects/Bandit
[4] https://www.npmjs.com/package/ng2lint

A Anhang

(Am Ende des Projekts nachzureichen)