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 Syntehtischen 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. CloneCademy orientiert sich hierbei an CodeCademy, einer interaktive Online-Lernplattform zum Erlernen von Programmierfähigkeiten.

Das Ziel der Plattform ist es, im Rahmen von interaktiven Unterrichtseinheiten Prinzipien der Molekularbiologie sowie der syntetischen Biologie erlernen zu können, und seine eigenen Fortschritte begutachten zu können. Darüber hinaus soll es auch anderen Interessierte (z.B. andere iGEM-Teams, Lehrende an Universitäten und Schulen, etc.) 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, als auch Metadaten über die Nutzung der Plattform und die Inhalte der einzelnen Lerneinheiten in die Datenbank speichert, ist es sehr wichtig, dass Internetnutzer keine Daten verändern oder einsehe können, solange sie dazu nicht die ausreichenden Rechte besitzen.

Die größte Bedrohung geht von bekannten Webangriffen und Misskonfigurationen im Backend einer Webseite aus. Im Rahmen dieses Projektes wird darauf geachtet, dass wir die Plattform gegen die 10 wichtigsten Sicherheitslücken in einer Webanwendung sichern.

Diese sind[1]:

1. Injection
2. Fehler in Authentifizierung und Session-Management
3. Cross-Site Scripting (XSS)
4. Fehlerhafte Zugriffskontrolle
5. Sicherheitsrelevante Fehlkonfiguration
6. Verlust der Vertraulichkeit sensibler Daten
7. Ungenügender Schutz vor Angriffen
8. Cross-Site Request Forgery (CSRF)
9. Nutzung von Komponenten mit bekannten Schwachstellen
10. 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 ist im folgenden detailliert ausgeführt.

[1] 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

Während der Implementierung achten alle Entwickler darauf, die Best Practices im Web Development zu beachten. Als Referenz dafür werden die Handreichungen des Open Web Application Security Project (OWASP)[2] verwendet. Zusätzlich werden die zur Verfügung gestellten Sicherheitswerkzeuge der genutzten Frameworks(Python Django, Django REST, Angular2) eingebunden. 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 und informieren die Code Review. Das Team achtet darauf, alle gefundenen Schwachstellen soweit wie möglich 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 durch und meldet die Ergebnisse an die jeweiligen Entwickler des betroffenen Moduls, damit diese 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 Veränderbarkeit

Für die Webanwendung CloneCademy ist es wichtig, dass die Anwendung im nachhinein noch veränderbar ist. Veränderbar in so fern: Es können neue Inhalte eingepflegt, bearbeitet und gelöscht werden. Da es im CloneCadamy eine Online Lernplattform handelt wie CodeCademy, müssen neue Kurse und Fragen eingepflegt werden.

2.3 Bedienbarkeit

CloneCademy müss für jede Benutzerrolle (Admin, Moderator, Nutzer) leicht zu Bedienen sein. Auch Personen die weniger IT-Affinität besitzen sollten sich auf der Lernplattform gut zurrecht finden können. Als Beispiel für die Moderatoren: Kurse anlegen, bearbeiten, löschen, neue Fragen einpflegen, alte bearbeiten und löschen usw. Diese Aufgaben sollen auch ohne eine Einführung einfach zu machen sein.

[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)