Exposé zur Bachelorarbeit

Arbeitstitel

Sicherheit bei der automatisierten Bewertung studentischer Java-Programme: Entwurf eines statischen Analysewerkzeugs zur Erkennung potenziell schädlichen Codes

---

1. Problemstellung

Im Rahmen von Programmierkursen an Universitäten reichen Studierende regelmäßig Java-Programme zur automatisierten Bewertung über Plattformen wie Moodle ein. Diese Programme werden im Backend ausgeführt und anhand vordefinierter Unit-Tests bewertet. Da der eingesandte Code jedoch in einer echten Java Virtual Machine (JVM) ausgeführt wird, besteht ein grundsätzliches Risiko: Studierende könnten bewusst oder unbewusst sicherheitskritischen oder systemschädlichen Code einreichen.

Mögliche Angriffe beinhalten etwa das absichtliche Belegen großer Mengen Arbeitsspeicher, Dateioperationen (z. B. Löschen wichtiger Daten), Netzwerkkommunikation (z. B. Botnetzverhalten) oder das Umgehen von Einschränkungen durch Reflection oder dynamisches Laden von Klassen. Die bestehende Unit-Test-basierte Bewertungsmethodik erkennt solche Fälle nicht zuverlässig.

---

2. Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines statischen Analysewerkzeugs, das vor der Ausführung eines eingereichten Java-Programms den Bytecode untersucht und auf potenziell gefährliches Verhalten überprüft. Das Werkzeug soll in die bestehende Moodle-Testumgebung integriert werden können und eine zusätzliche Sicherheitsebene zur automatisierten Bewertung darstellen.

---

3. Fragestellungen

Die Arbeit soll unter anderem folgende Fragen beantworten:

- Welche konkreten Gefahren gehen von studentischem Java-Code aus?
- Inwiefern lassen sich diese Gefahren durch statische Analyse des Bytecodes erkennen?
- Welche Methoden der Bytecode-Analyse sind geeignet, und wie lassen sich verbotene Muster zuverlässig identifizieren?
- Wie kann das Tool in ein bestehendes Bewertungssystem (z. B. Moodle) integriert werden?
- Wo liegen die Grenzen statischer Analyse in Bezug auf Sicherheit?

---

4. Methodik

Die Arbeit basiert auf einem forschungs- und praxisorientierten Ansatz. Die zentralen Schritte sind:

1. **Literatur- und Grundlagenrecherche:** JVM-Architektur, Bytecode, Reflection, Java Security Manager (Legacy), Sandboxing-Konzepte
2. **Bedrohungsmodellierung:** Identifikation realistischer Angriffsvektoren anhand typischer Studierenden-Abgaben
3. **Toolentwicklung:** Implementierung eines statischen Analysewerkzeugs mit Java-Bytecode-Frameworks wie ASM oder BCEL
4. **Integration:** Entwicklung eines Prototyps zur Einbindung in ein Moodle-basiertes Bewertungssystem
5. **Evaluation:** Durchführung kontrollierter Tests mit harmlosen und absichtlich schädlichen Java-Programmen

---

5. Erwartete Ergebnisse

- Ein funktionierender Prototyp eines Java-Sicherheitsscanners, der Bytecode auf potenziell gefährliches Verhalten untersucht
- Eine Klassifikation typischer Gefahren und ein Katalog erkannter Muster
- Dokumentierte Integration in eine bestehende Moodle-Testumgebung
- Bewertung der Effektivität und Grenzen der statischen Analyse

---

6. Zeitplan (12 Wochen)

Zeitraum	Aufgabe
Woche 1-2	Literaturrecherche, Bedrohungsmodell
Woche 3-4	Technologierecherche & Prototyp-Design
Woche 4-8	Implementierung des Scanners
Woche 8-10	Integration und Testphase
Woche 10-12	Evaluation, Schreiben & Abgabe
