Fakultät Informatik

CYBRAIL

IT-Projekt im Studiengang Bachelor Informatik

vorgelegt von

Mattis Krämer, Robin Rosner, Pascal Blank

Erstellungssemester SS2024 - WS2425

Betreuer: Dr. Martin Geier

Inhaltsverzeichnis

1 Einleitung

CYBRAIL ist ein Tool, das es ermöglichen soll, mögliche Betrugsversuche in Online- oder Präsenzklausuren am Computer zu erkennen. Dabei liefert es eine nachvollziehbare Begründung anhand von verschiedenen Indizien, die während einer Onlineklausur in Logdateien gesammelt und später durch dieses Tool ausgewertet werden können.

1.1 Safe Exam Browser

Der Safe Exam Browser (SEB) ist ein abgesicherter Browser, der speziell für Prüfungen an Schulen entwickelt wurde. Seine Funktionalität ist stark eingeschränkt, um die den Studierenden zur Verfügung stehenden Hilfsmittel erheblich zu beschränken. Der SEB wird auch an unserer Hochschule eingesetzt, weshalb der Fokus dieses Projekts auf diesen Browser ausgerichtet ist. Zu den Funktionen des SEB gehören unter anderem Prüfungen, ob unerlaubte Programme im Hintergrund laufen, ob der Bildschirm geteilt wird, und es wird verhindert, dass Studierende die SEB-Umgebung verlassen, andere Programme öffnen oder Copy-Paste verwenden. Somit bildet der SEB eine solide Grundlage, um faire Prüfungen zu ermöglichen und Betrug zu verhindern.

1.2 Cheating with SEB

Jedoch hat jedes System auch gewisse Schwächen.

1.2.1 Virtuelle Machine

Frühere Versionen des SEB konnten durch den einfachen Austausch einer Programmdaten so verändert werden, dass der Browser in einer Virtuellen Maschine (VM) gestartet werden konnte. Dies wird eigentlich durch mehrere komplexe Prüfungen unterbunden, die den Start in einer

VM verhindern sollten. Diese Manipulation konnte jedoch mit einer etwa fünfminütigen Google-Suche leicht und unkompliziert durchgeführt werden. Dadurch ist es möglich, den SEB in einem Fenster oder auf einem zweiten Bildschirm auszuführen, während auf dem Hostsystem – also dem System, das die VM betreibt – nach Lösungen gegoogelt, Videoanrufe durchgeführt oder Aufzeichnungen gemacht werden können.

Dennoch gab es bereits in diesen Fällen gewisse Indizien in diversen Logdateien, die auf einen Missbrauch hinwiesen. Beispielsweise wird geloggt, wenn sich die Bildschirmauflösung ändert, was passiert, wenn die Fenstergröße der VM angepasst wird oder der Bildschirm in den Vollbildmodus wechselt. Der SEB verfügt zudem über eine Integritätsprüfung, um Modifikationen festzustellen.

Beide dieser Indizien werden jedoch lediglich in Logdateien geschrieben. Ohne eine Auswertung und Sammlung der Logdateien erfährt der Prüfer oder die Aufsichtsperson nichts von solchen Vorfällen.

1.2.2 Automatisches Tippen

Ein weiterer Angriffsvektor ist nahezu unentdeckbar: das automatische Eintippen von Zeichen anstelle des normalen Copy-Pastes. Dies könnte entweder durch auf dem Gerät laufende Software geschehen, die nicht durch den SEB blockiert wird – hier würde nur eine Whitelist helfen, da ein solches Programm beliebig benannt werden kann – oder durch einen Hardware-Dongle, der beispielsweise Text von einem anderen Gerät automatisch eintippt. So könnten Lösungen auf einem zweiten Gerät gesucht oder unter den Studierenden geteilt und anschließend schnell und bequem automatisch eingetippt werden.

1.2.3 SEB Server

Alle Informationen, die in Logdateien gespeichert werden, liegen zunächst auf dem Client des Nutzers und sind somit unzugänglich für die unmittelbare Erkennung von auffälligem Verhalten. SEB bietet jedoch die Möglichkeit, einen optionalen SEB-Server einzusetzen. Dieser kann an der Hochschule aufgesetzt werden und nach korrekter Konfiguration des Servers sowie der Konfigurationsdatei für den Client – welche vor Beginn der Klausur direkt von Moodle geladen

werden kann – Logdateien vom Client empfangen. Dies stellt jedoch die gesamte Funktionalität des Servers dar; es findet keine automatische Auswertung der Logs statt, was bedeutet, dass ein Mensch versucht – vermutlich stichprobenartig – Auffälligkeiten zu finden. Dies ist jedoch sehr schwierig, es sei denn, man ist mit der Funktionsweise des SEB vertraut.

1.3 Lösung

Eine offensichtliche Lösung wäre ein Tool, das diesen Prozess möglichst generisch und erweiterbar automatisiert: CYBRAIL – Cyber Barrier for Reliable Academic Integrity and Log-analysis.

2 Projektziele

Wir als Team haben uns mehrere Ziele für dieses Projekt gesetzt, um ein solches Tool zu entwickeln.

2.1 Log- und Informationssammlung

Unser erstes Ziel war es herauszufinden, inwiefern wir Zugriff auf die gespeicherten Logs erhalten können, wie diese übertragen werden und in welchem Format sie vorliegen. Darüber hinaus haben wir untersucht, wie diese Informationen von einem Computer aufbereitet werden können, um sie anschließend zu analysieren.

2.2 Demonstration der Erkennung von Auffälligkeiten

Ein weiteres wichtiges Ziel war es, herauszufinden, welche Betrugsversuche überhaupt erkannt werden können und welche davon relevant sind. Ebenso war es von Bedeutung zu klären, welche Informationen für die Erkennung benötigt werden. Darüber hinaus mussten wir identifizieren, welche Betrugsversuche vermutlich nicht erkannt werden können und welche zusätzlichen Informationen oder Logs dafür erforderlich wären.

Letztendlich galt es, zu bewerten, auf welche Betrugserkennungsmethoden wir uns im Rahmen dieses Projekts fokussieren wollen, um diese zu demonstrieren.

2.3 Benutzerfreundlichkeit und einfache Bereitstellung

Das letzte Ziel des Projekts war es, eine möglichst einfache Benutzeroberfläche sowie eine unkomplizierte Einrichtung zu ermöglichen. Unser Projekt sollte schnell und einfach in der Praxis einsetzbar sein, sei es auf einem Server oder einem PC, ohne dass ein komplexes Setup erforderlich ist.

3 Projektverlauf und Meilensteine

3.1 Programmentwurf und Architekturentwurf

rewrite

Der Entwurf des Programms sowie die Architektur von CYBRAIL basieren auf einer klaren Strukturierung der Funktionalitäten, um eine effiziente und erweiterbare Lösung zu gewährleisten. Im folgenden Abschnitt wird der Architekturentwurf näher erläutert, gefolgt von einem detaillierten Blick auf die wichtigsten Klassen und deren Interaktionen.

3.1.1 Architekturentwurf

Das Architekturdiagramm (siehe Abbildung **??**) zeigt die modulare Struktur des Systems. CYBRAIL ist in verschiedene Schichten unterteilt, um die Verantwortlichkeiten klar voneinander zu trennen und eine flexible Erweiterbarkeit zu ermöglichen. Die wichtigsten Schichten sind:

- **Benutzeroberfläche (UI):** Diese Schicht stellt die Interaktion mit den Anwendern sicher und bietet eine intuitive Oberfläche, um Prüfungen und Auswertungen zu verwalten. Sie ermöglicht die Anzeige von Logdaten, Konfigurationsmöglichkeiten und die Ausgabe der Analyseergebnisse.
- **Analyse-Engine:** Das Herzstück von CYBRAIL ist die Analyse-Engine, die die gesammelten Logdaten nach Auffälligkeiten untersucht. Diese Schicht enthält alle notwendigen Algorithmen, um potenzielle Betrugsversuche zu erkennen und entsprechende Berichte zu generieren.
- **Datenverwaltung:** Hier werden alle eingehenden Daten verarbeitet und in geeigneten Datenbanken gespeichert. Diese Schicht sorgt für den Zugriff auf Logdateien, Konfigurationsdaten und andere notwendige Informationen.

- **Schnittstellen zur Datenquelle:** Die unterste Schicht stellt die Anbindung an externe Systeme wie den Safe Exam Browser (SEB) sicher. Hier werden Logdaten gesammelt und über definierte Schnittstellen an die Datenverwaltung und Analyse-Engine weitergeleitet.

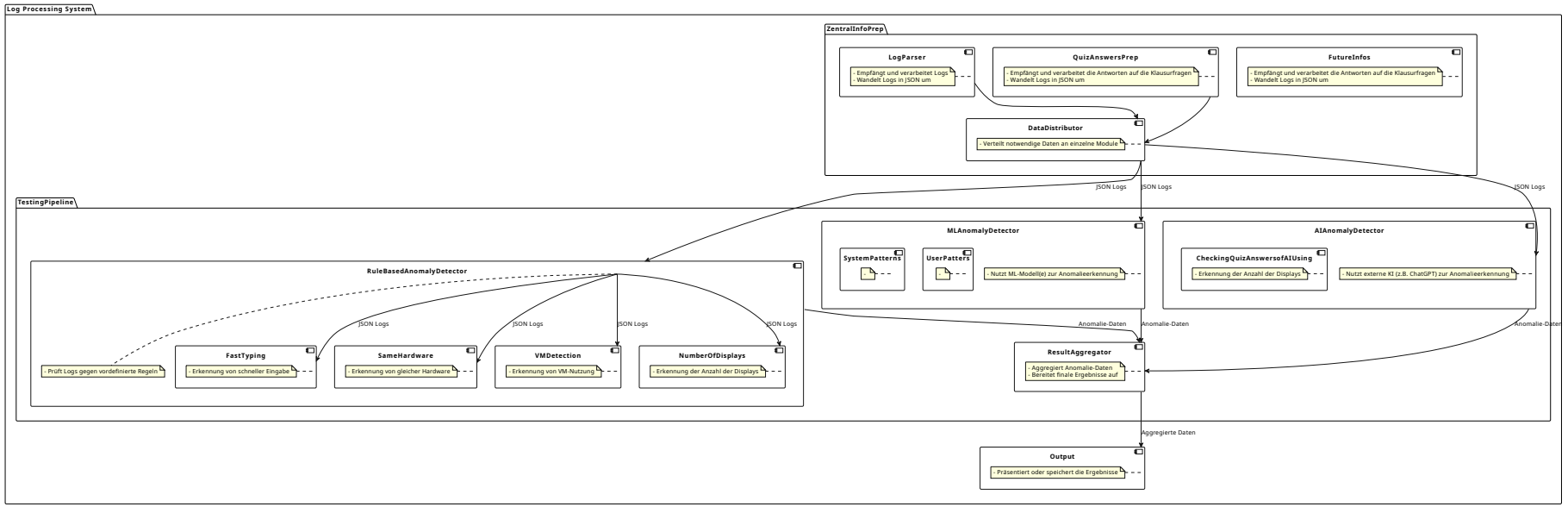

Abbildung 3.1: Klassendiagramm von CYBRAIL

Durch diese modulare Architektur ist es möglich, einzelne Komponenten unabhängig voneinander zu entwickeln, zu testen und bei Bedarf auszutauschen, ohne das Gesamtsystem zu beeinträchtigen.

3.1.2 Programmentwurf

Das Klassendiagramm in Abbildung **??** verdeutlicht die wichtigsten Klassen und deren Beziehungen innerhalb des Systems. Die zentrale Klasse ist `LogAnalyzer`, welche für die Analyse der Logdaten zuständig ist. Die wichtigsten Klassen und deren Funktionen sind:

- `LogAnalyzer`: Diese Klasse enthält die Logik zur Auswertung der Logdateien und identifiziert potenzielles Fehlverhalten basierend auf vorgegebenen Regeln und Mustern.
- `LogFile`: Stellt eine Abstraktion für eine Logdatei dar. Diese Klasse übernimmt das Einlesen und Vorverarbeiten der Logs, die dann vom `LogAnalyzer` ausgewertet werden.
- `User`: Repräsentiert die Informationen zu einem Prüfling oder Nutzer des Systems. Diese Klasse wird insbesondere verwendet, um spezifische Daten während einer Prüfung zu speichern.
- `ReportGenerator`: Diese Klasse erstellt Berichte basierend auf den Ergebnissen der Loganalyse und bietet eine übersichtliche Darstellung der detektierten Auffälligkeiten.

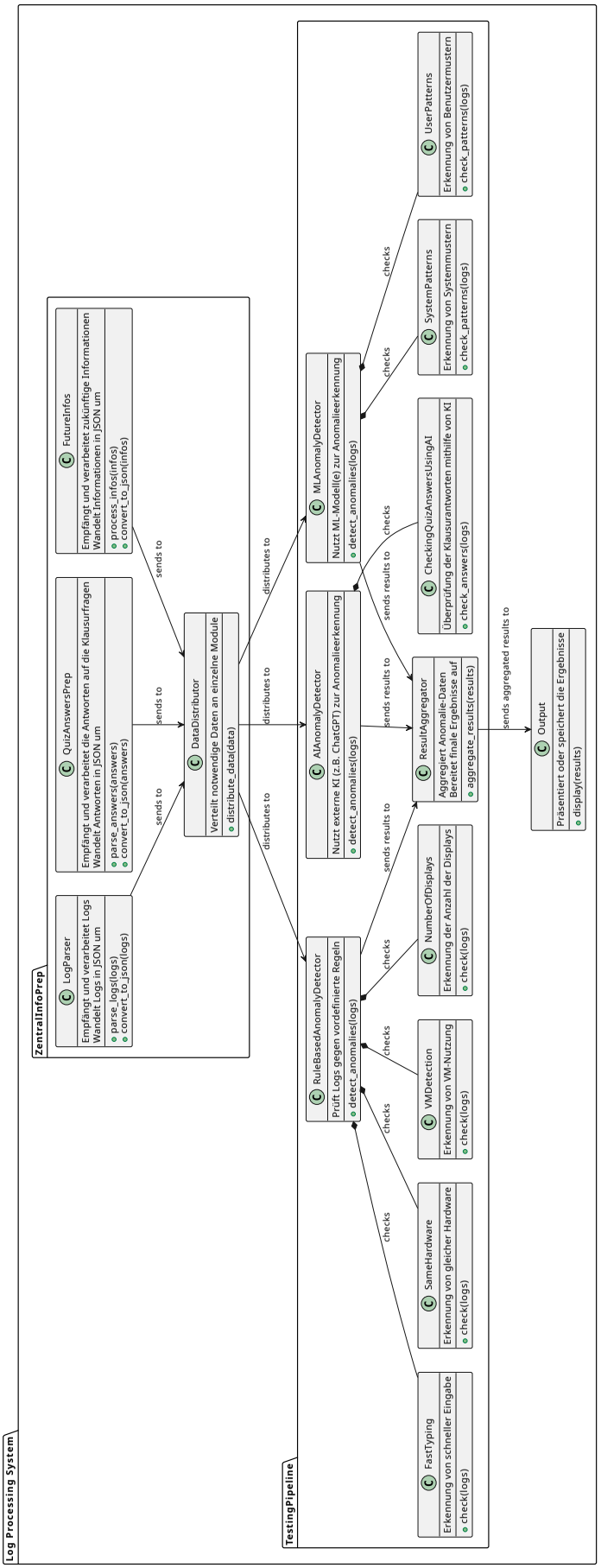

Abbildung 3.2: Architekturdiagramm von CYBRAIL

Das Zusammenspiel dieser Klassen ermöglicht es, Logdaten effizient zu verarbeiten, Auffälligkeiten zu erkennen und verständliche Berichte für Prüfungsaufsichten zu generieren. Die modulare Struktur erlaubt es zudem, neue Regeln und Methoden zur Betrugserkennung hinzuzufügen, ohne bestehende Funktionalitäten zu beeinträchtigen.

3.1.3 Zusammenfassung

Der Architektur- und Programmentwurf von CYBRAIL folgt einem modularen Ansatz, der eine einfache Erweiterbarkeit und Wartbarkeit des Systems ermöglicht. Durch die klare Trennung der Verantwortlichkeiten kann CYBRAIL sowohl für die Verarbeitung großer Datenmengen als auch für eine schnelle Analyse in Echtzeit verwendet werden. Die klare Definition der Klassen und deren Aufgabenbereiche stellt sicher, dass das System flexibel auf neue Anforderungen angepasst werden kann.

4 Ergebnisse und Leistungen

5 Qualitätssicherung

6 Risikoanalyse und Risikomanagement

Bei der Entwicklung von CYBRAIL wurden verschiedene Risiken identifiziert, um mögliche Hindernisse oder Probleme während des Projekts frühzeitig zu erkennen und Gegenmaßnahmen zu definieren. Die Risiken lassen sich in fünf Hauptkategorien unterteilen:

- **Technische Risiken:** Diese umfassen Softwarefehler, Kompatibilitätsprobleme zwischen verschiedenen Versionen des Safe Exam Browsers (SEB) und Sicherheitslücken.
- **Datenschutz- und Compliance-Risiken:** Hier geht es vor allem um Datenschutzverletzungen und die Einhaltung von Vorschriften zum Schutz der Daten der Studierenden.
- **Betriebliche Risiken:** Diese betreffen Ausfälle oder Unterbrechungen des Systems sowie Skalierbarkeitsprobleme.
- **Annahme- und Akzeptanzrisiken:** Hierzu gehören der Widerstand von Lehrenden und Studierenden sowie der Mangel an Schulungen und Unterstützung.
- **Projektspezifische Risiken:** Diese umfassen den fehlenden Zugriff auf Logdaten der Studierenden und die Anbindung an universitäre Dienste wie Moodle nach der Implementierung.

Basierend auf der Wahrscheinlichkeit und dem Schadensausmaß jedes Risikos wurde eine Risikoprioritätszahl (RPZ) berechnet. Diese gibt an, wie dringend das Risiko behandelt werden muss. Für jedes Risiko wurden außerdem Gegenmaßnahmen entwickelt, um dessen Auswirkungen zu minimieren oder ganz zu vermeiden.

Die folgende Tabelle fasst die wichtigsten Risiken zusammen, die im Projekt identifiziert wurden:

Risiko	**Schadenshöhe (1–10)**	**Wahrscheinlichkeit**	**RPZ**	**Gegenmaßnahmen**
Softwarefehler und -probleme: Wie man sie handhabt und was nach dem Projekt passiert	2	0,7	1,4	FAQ, Tests, abhängig von der Geschichte
Kompatibilitätsprobleme mit verschiedenen Versionen des SEB	8	0,1	0,8	Konfigurierbar
Sicherheitslücken	7	0,5	3,5	Sicherheitskonzepte, Penetrationstests
Datenschutzverletzungen	8	0,4	3,2	Sicherheitskonzepte, Penetrationstests, Datenspeicherlebensdauer
Einhaltung der Datenschutzvorschriften der Studierenden	7	0,2	1,4	Datensparsamkeit, Rücksprache mit Petric
Ausfallzeiten und Unterbrechungen	2	0,2	0,4	–
Skalierungsprobleme	2	0,2	0,4	–
Widerstand von Lehrenden und Studierenden	2	0,6	1,2	–
Mangel an Schulungen und Unterstützung	8	0,3	2,4	Dokumentation, Benutzeroberfläche
Kein Zugriff auf Logdaten der Studierenden	10	0,25	2,5	–
Kein Zugriff auf universitäre Dienste wie Moodle nach der Implementierung	10	0,1	1	–

Tabelle 6.1: Risikobewertung und Gegenmaßnahmen

15

6.1 Zusammenfassung der Risikobewertung

Die Risikobewertung zeigt, dass die kritischsten Risiken in den Bereichen Datenschutzverletzungen und Sicherheitslücken liegen, die beide durch entsprechende Sicherheitsmaßnahmen und Penetrationstests gemindert werden können. Gravierend wäre jedoch der fehlende Zugriff auf die Logdaten der Studierenden, da diese essenziell für die Funktion unseres Tools sind. Ohne diese Daten wäre eine Auswertung und Erkennung von Betrugsversuchen nicht möglich. Allerdings schätzen wir die Wahrscheinlichkeit, dass hierfür keine Lösung gefunden werden kann, als sehr gering ein. Mit den entsprechenden technischen und organisatorischen Maßnahmen, wie der Nutzung des SEB-Servers und der korrekten Konfiguration des Systems, sollten wir in der Lage sein, auf die notwendigen Logs zuzugreifen und diese sicher zu analysieren.

7 Kommunikation

8 Ressourcenmanagement

9 Lessons Learned

10 Abschlussbewertung

11 Fazit und Ausblick

Abbildungsverzeichnis

Anhang