Projektübersichtsblatt: Web Vulnerability Scanner (WVS)

Projektziel: Entwicklung eines automatisierten Werkzeugs zur Erkennung von Sicherheitslücken in Webanwendungen, basierend auf den OWASP Top 10, um die Websicherheit für kleine und mittelständische Unternehmen (KMU) zu verbessern.

Gruppenmitglieder und detaillierte Aufgabenverteilung

Mitglied	GitHub-Alias	Rolle im Projekt	Übernommene Aufgaben
Maurice	Aircrack	**Lead Developer / Architekt**	- **Architektur & Kernentwicklung:** Entwurf und Implementierung der modularen Gesamtarchitektur des Scanners. - **Kommandozeilenschnittstelle (CLI):** Entwicklung des Haupteinstiegspunkts (wvs.py) mit typer zur Steuerung des gesamten Programms. - **Scanner-Engine:** Programmierung der zentralen Logik (wvs/scanner/engine.py), die für das dynamische Laden und Ausführen der einzelnen Testmodule verantwortlich ist. - **Konfigurationsmanagement:** Implementierung des Systems zum Einlesen der wvs.toml-Konfigurationsdatei. - **Modulentwicklung:** Erst-Implementierung und Konzeption der Basis-Scan-Module.
Marius	Mariusxy	**Entwickler / Bugfixing & Modul-Spezialist**	- **Modulentwicklung:** Implementierung und Verfeinerung spezifischer Scanner-Module, insbesondere für komplexe Tests wie die Erkennung von Injection-Schwachstellen (A03_Injection). - **Bugfixing:** Systematische Identifizierung und Behebung von Fehlern, die während der Entwicklung und den Tests auftraten. - **Reporting-System:** Maßgebliche Mitarbeit an der Entwicklung der verschiedenen Ausgabeformate, insbesondere des PDF-Reporters (wvs/reporting/pdf_reporter.py). - **Tooling & Tests:** Unterstützung bei der Erstellung und Wartung der Test-Suite.
Paul	Learning Birdi1212	**Koordinator / Qualitätssicherung &**	- **Projektkoordination:** Organisation der Team-Meetings, Überwachung des

		Dokumentation	Projektfortschritts und Sicherstellung der Einhaltung von Deadlines. - **Qualitätssicherung (QA):** Hauptverantwortlich für die Durchführung von Code-Reviews. Überprüfung aller Pull-Requests auf Code-Qualität, Lesbarkeit und Funktionalität. - **Git-Workflow-Management:** Verwaltung des Git-Repositorys, Durchsetzung des Branching-Modells und Zusammenführen (Merging) von Pull-Requests. - **Dokumentation:** Erstellung und Pflege der Projektdokumentation, einschließlich des Lösungskonzepts, der Problemstellung und der finalen Projektdokumentation.

Organisation der Zusammenarbeit

Die Zusammenarbeit im Team war klar strukturiert, um eine hohe Effizienz und Code-Qualität sicherzustellen. Die folgenden Methoden und Werkzeuge kamen zum Einsatz:

- **Kommunikation:** Regelmäßige (wöchentliche) Meetings auf Discord zur Synchronisierung des Fortschritts, zur Diskussion von Problemen und zur Planung der nächsten Schritte.
- **Aufgabenverteilung:** Die Aufgaben wurden nach dem Kick-off-Meeting basierend auf den Stärken und Interessen der einzelnen Mitglieder klar verteilt. Dies ermöglichte eine parallele und effektive Bearbeitung der Arbeitspakete.
- **Versionierung und Workflow:**
    - **Tool:** Git & GitHub
    - **Workflow:** Das Team nutzte einen professionellen Git-Workflow. Die Entwicklung fand in separaten Feature-Branches statt. Neue Features oder Bugfixes wurden über **Pull-Requests** in den main-Branch integriert.
    - **Qualitätssicherung:** Jeder Pull-Request musste von mindestens einem anderen Teammitglied überprüft und genehmigt werden (**Code-Review**). Dieser Prozess stellte sicher, dass der Code den Qualitätsstandards entsprach und Fehler frühzeitig erkannt wurden.
- **Dokumentation:** Die Projektdokumentation wurde kollaborativ in Markdown-Dateien im Docs-Verzeichnis des Repositorys gepflegt, was eine zentrale und versionierte Wissensbasis schuf.