BECYB Projekt

Tomasz Lewiński

Aleksander Gajowniczek

Juliusz Kluge

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

26 maja 2025

Spis treści

1. Wprowadzenie

W ramach tematu "Bezpieczeństwo sieci - rozwiązania i narzędzia, nowe koncepcje" nasz zespół skupi się na eksploracji zastosowania modeli językowych AI w dziedzinie cyberbezpieczeństwa. Konkretnym zadaniem, które podejmiemy, będzie "Wykorzystanie ChatGPT do automatyzacji analizy zagrożeń".

2. Rozumienie tematu

Temat projektu zakłada wykorzystanie modelu językowego ChatGPT (lub podobnego) do stworzenia narzędzia wspomagającego analizę zagrożeń bezpieczeństwa w systemach informatycznych. Główne założenia to:
— Automatyzacja analizy logów systemowych
— Wsparcie dla administratorów w identyfikacji potencjalnych incydentów bezpieczeństwa
— Sugerowanie możliwych scenariuszy reakcji na wykryte zagrożenia

3. Zawężenie zagadnień

Projekt skupi się na następujących aspektach:
— Analiza wybranych typów logów systemowych (np. logi serwera WWW, firewalla)
— Identyfikacja podstawowych wzorców ataków (SQL Injection, XSS, Brute Force)
— Tworzenie czytelnych raportów dla administratora
— Implementacja podstawowych scenariuszy reakcji

4. Cele projektu

4.1. Cel główny

Stworzenie prototypu cybersecurity chatbota wykorzystującego API OpenAI do wstępnej analizy logów systemowych i wsparcia administratora w ocenie zagrożeń.

4.2. Cele szczegółowe

— Zbudowanie działającego prototypu w Pythonie z wykorzystaniem OpenAI API
— Opracowanie zbioru testowych logów zawierających różne typy zagrożeń
— Implementacja mechanizmu klasyfikacji zagrożeń
— Stworzenie interfejsu użytkownika (CLI lub webowy)
— Testy funkcjonalne rozwiązania

5. Zakres projektu

Projekt obejmuje:
— Analizę istniejących rozwiązań
— Projekt architektury systemu
— Implementację podstawowej wersji chatbota
— Testy na syntetycznych danych
Projekt **nie obejmuje**:
— Pełnej integracji z systemami produkcyjnymi
— Zaawansowanych mechanizmów detekcji zagrożeń
— Szkolenia własnego modelu AI

6. Przewidywane efekty

— Działający prototyp chatbota analizującego logi
— Dokumentacja techniczna rozwiązania
— Zbiór testowych przypadków użycia
— Raport z testów i oceny skuteczności

7. Literatura i materiały pomocnicze

— Oficjalna dokumentacja OpenAI API: `https://platform.openai.com/docs`
— OWASP Top 10: `https://owasp.org/www-project-top-ten/`
— Common Log Format specification
— Publikacje naukowe dot. zastosowań AI w cyberbezpieczeństwie
— Dokumentacja bibliotek Python do parsowania logów (np. pygtail, loguru)

8. Wnioski

Przedstawiony dokument określa wstępny zakres projektu, który może ulec niewielkim modyfikacjom w trakcie realizacji. Głównym celem jest stworzenie funkcjonalnego prototypu demonstrującego potencjał wykorzystania modeli językowych w automatyzacji analizy zagrożeń bezpieczeństwa.

9. Realizacja Projektu

Poniżej przedstawiono szczegółowy raport realizacji projektu według przesłanego zakresu, obejmujący:

1. Analizę istniejących rozwiązań
2. Projekt architektury systemu
3. Implementację podstawowej wersji chatbota
4. Testy na syntetycznych danych (pominięto w tym dokumencie)

Każda z sekcji zawiera omówienie kluczowych zagadnień, fragmenty kodu oraz wskazówki dotyczące wdrożenia.

9.1. Analiza Istniejących Rozwiązań

Do analizy wybrano następujące narzędzia wspierające analizę logów serwerowych:

— **GoAccess**
  Zalety:
  — Szybka analiza w czasie rzeczywistym
  — Intuicyjny interfejs (tekstowy i webowy)
  — Łatwa instalacja i konfiguracja
  Wady:
  — Ograniczone funkcje bezpieczeństwa
  — Brak automatycznego alertowania
— **AWStats**
  Zalety:
  — Obsługa wielu formatów logów
  — Szczegółowe raporty graficzne
  — Możliwość analizy logów HTTP, FTP, poczty
  Wady:
  — Brak zaawansowanej analizy bezpieczeństwa
  — Może wymagać dodatkowej konfiguracji
— **Elastic Stack (ELK)**
  Zalety:

— Skalowalność i elastyczność
— Zaawansowane wyszukiwanie i analiza logów
— Bogate możliwości wizualizacji
Wady:
— Duże wymagania sprzętowe
— Wysoka krzywa uczenia się
— **Splunk**
Zalety:
— Rozbudowana analityka i monitoring
— Zaawansowane raportowanie i alertowanie
— Wsparcie techniczne
Wady:
— Wysokie koszty licencyjne
— Przerost formy nad treścią w małych projektach
— **ManageEngine EventLog Analyzer**
Zalety:
— Automatyczne wykrywanie zagrożeń
— Raporty zgodności z regulacjami
— Intuicyjny interfejs
Wady:
— Mniejsza elastyczność niż rozwiązania open-source
— Potencjalnie wysoki koszt wdrożenia
— **SolarWinds Loggly**
Zalety:
— Łatwa integracja z chmurą
— Szybkie wyszukiwanie i filtrowanie
— Skalowalność i dostępność
Wady:
— Wymaga stałego połączenia z Internetem
— Koszt rośnie wraz z ilością danych
— **LORG**
Zalety:
— Open-source zorientowany na bezpieczeństwo
— Identyfikacja zagrożeń i anomalii HTTPD
Wady:
— Ograniczone możliwości wizualizacji
— Może wymagać dodatkowych modułów

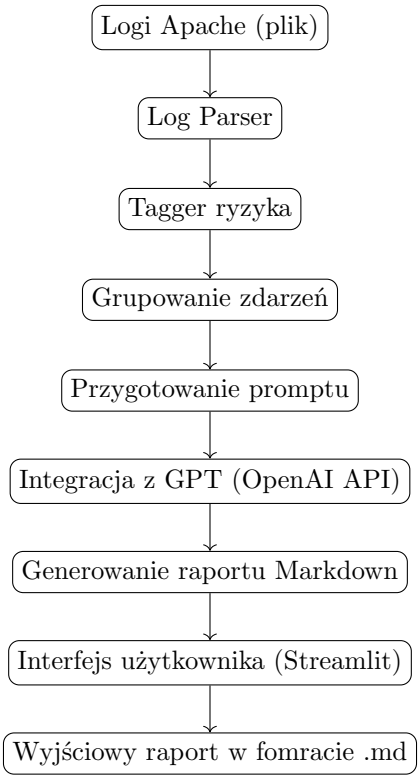
Wniosek: Warto skupić się na integracji heurystycznej analizy logów z generowaniem zaleceń przy użyciu AI. Dodatkowo, chatbotowy interfejs przyspiesza interpretację wyników przez użytkownika końcowego.

9.2. Projekt Architektury Systemu

Moduły systemu:

1. **Log Parser** – parsowanie logów (funkcje `parse_log_line`, `parse_log_file`)
2. **Risk Analyzer** – klasyfikacja zagrożeń (funkcje `classify_risk`, `tag_logs_with_risk`)
3. **Batcher** – grupowanie podobnych zdarzeń (funkcja `batch_logs`)
4. **GPT Integration** – analiza zagrożeń przez OpenAI API (funkcje `format_prompt`, `analyze_logs_with_openai`)
5. **Frontend (Streamlit)** – interfejs użytkownika do przesyłania plików i generowania raportów

Schemat blokowy:



Technologie:
— Język: Python
— Interfejs: Streamlit
— Model: GPT-4o (OpenAI API)
— Konfiguracja: plik `.env`

9.3. Struktura projektu

```
projekt_log_analyzer/
 main.py
 src/
    log_parser.py
    risk_analyzer.py
    batcher.py
 .env
 README.md
```

9.4. Kluczowe funkcje

9.4.1. Interfejs Streamlit:

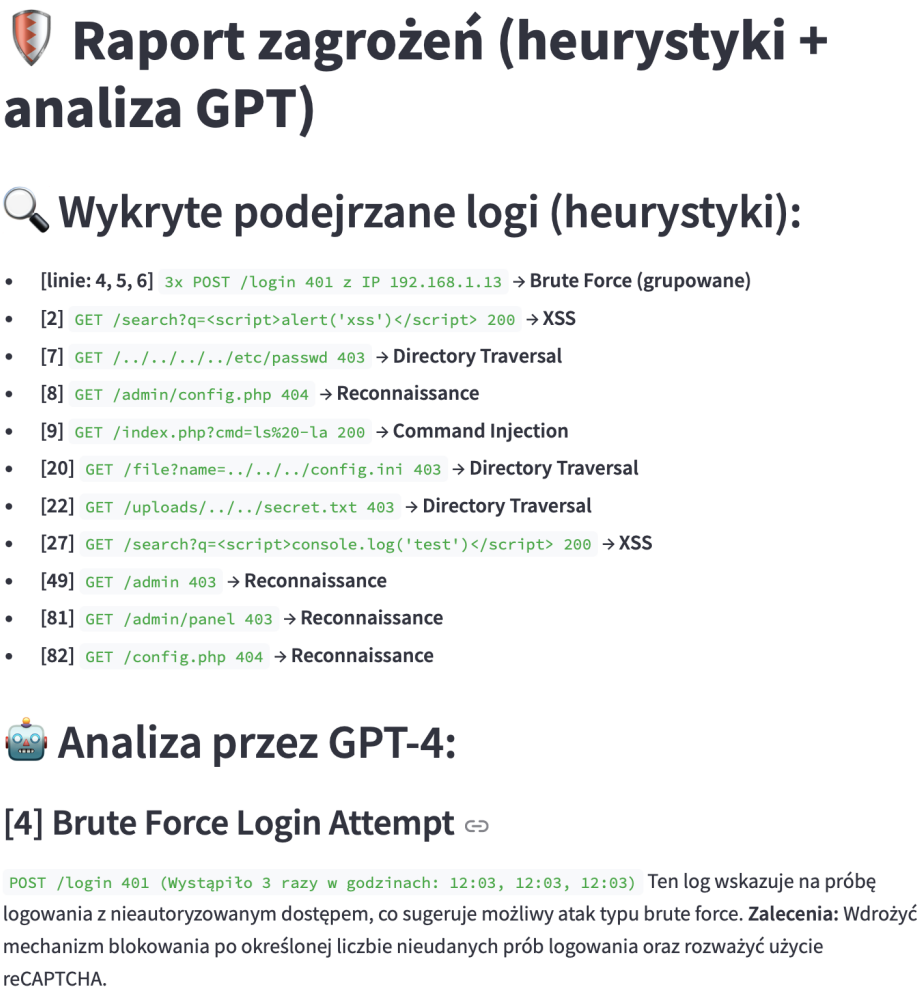
— Umożliwia wgranie pliku logów i uruchomienie analizy
— Wyświetla raport wygenerowany przez GPT w formacie Markdown
— Pozwala pobrać raport na dysk

9.4.2. Chatbot jako interfejs analityka:

— Użytkownik dostarcza logi
— System analizuje dane i klasyfikuje zagrożenia
— GPT generuje wyjaśnienie i rekomendacje
— Raport jest prezentowany i możliwy do pobrania

9.5. Demonstracja działania:

Rys. 1: Frontend aplikacji



Rys. 2: Przykładowa odpowiedź

Przykładowy raport został przesłany w pliku: "threat_report.md"

9.6. Podsumowanie

Projekt łączy analizę logów serwera Apache z heurystyką oceny zagrożeń oraz integracją z GPT w celu wygenerowania szczegółowego raportu.

Cechy rozwiązania:

— **Modularna architektura** – ułatwia rozwój i testowanie
— **Wykorzystanie AI** – zapewnia głębszą analizę i rekomendacje
— **Przyjazny interfejs** – umożliwia szybkie wykorzystanie przez użytkownika końcowego

System ten może stanowić efektywne i nowoczesne narzędzie dla administratorów odpowiedzialnych za bezpieczeństwo infrastruktury serwerowej.