Politechnika Wrocławska
Wydział informatyki i Telekomunikacji

Kierunek: **Cyberbezpieczeństwo (CBE)**
Specjalność: **Bezpieczeństwo danych (CBD)**

PRACA DYPLOMOWA
INŻYNIERSKA

Zastosowanie dużych modeli językowych do wykrywania i naprawiania błędów bezpieczeństwa i podatności w kodzie aplikacji webowych

Patryk Fidler

Opiekun pracy
Dr. hab. inż. Maciej Piasecki

Słowa kluczowe: modele językowe, Sztuczna Inteligencja, statyczna analiza kodu

WROCŁAW 2024

STRESZCZENIE

Praca inżynierska zatytułowana "Zastosowanie dużych modeli językowych do wykrywania i naprawiania błędów bezpieczeństwa i podatności w kodzie aplikacji webowych" koncentruje się na zastosowaniu zaawansowanych modeli językowych, takich jak GPT-3.5, GPT-4, a w razie możliwości Falcon, do automatycznego wykrywania i naprawiania błędów bezpieczeństwa w kodzie oprogramowania i aplikacji webowych.

Motywacja tej pracy wynika z rosnącej roli dużych modeli językowych (LLM) w różnych dziedzinach, w tym w cyberbezpieczeństwie. W kontekście tych działań, badana jest możliwość wykorzystania tych modeli do wykrywania i naprawiania podatności takich jak XSS, SQL Injection, CSRF, Buffer Overflow i tym podobne.

Punktem wyjścia dla pracy dyplomowej jest artykuł napisany w 2021 roku "Can OpenAI Codex and Other Large Language Models Help Us Fix Security Bugs?" - Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, Brendan Dolan-Gavitt (https://arxiv.org/pdf/2112.02125v1.pdf). Autorzy podkreślają znaczący potencjał tych modeli, a niniejsza praca ma na celu kontynuację tych badań i poszerzenie ich zakresu.

Planowane działania obejmują przygotowanie zbioru danych zawierającego podatne bazy kodu źródłowego, testowanie zdolności detekcji błędów przez modele językowe OpenAI, oraz porównanie tych wyników z istniejącymi rozwiązaniami, oferowanymi przez firmę Snyk.

Szczególny nacisk zostanie położony na wykorzystanie technik soft-prompting i in-context learning, które mogą pomóc w udoskonaleniu detekcji i wyników, nawet przy ograniczonych zasobach. Praca przewiduje również implementację autonomicznego agenta AI zdolnego do analizy kodu, wykonania testów bezpieczeństwa i podejmowania decyzji na podstawie wyników tych testów i kontekstu.

Opcjonalnie, badane będą możliwości detekcji błędów przez otwarte modele językowe, a w dalszej perspektywie, możliwości specjalizacji modeli w zakresie cyberbezpieczeństwa za pomocą fine-tuning'u. Wszystkie te działania mają na celu nie tylko zrozumienie, ale także poprawienie możliwości LLM w kontekście cyberbezpieczeństwa.

Głównym celem pracy jest zwiększenie świadomości na temat potencjału dużych

modeli językowych w cyberbezpieczeństwie oraz proponowanie praktycznych rozwiązań, które mogą pomóc programistom w tworzeniu bardziej bezpiecznych aplikacji.

ABSTRACT

The engineering thesis titled "Application of Large Language Models for Detecting and Fixing Security Bugs and Vulnerabilities in Web Application Code" focuses on the utilization of advanced language models, such as GPT-3.5, GPT-4, and if possible, Falcon, for automated detection and rectification of security bugs in the code of software and web applications.

The motivation for this work stems from the growing role of Large Language Models (LLMs) in various fields, including cybersecurity. In the context of these efforts, the possibility of using these models to detect and fix vulnerabilities such as XSS, SQL Injection, CSRF, Buffer Overflow, and the like is being investigated.

The starting point for this dissertation is the 2021 article "Can OpenAI Codex and Other Large Language Models Help Us Fix Security Bugs?" by Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, Brendan Dolan-Gavitt (https://arxiv.org/pdf/2112.02125v1.pdf). The authors emphasize the significant potential of these models, and this work aims to continue this research and broaden its scope.

Planned activities include preparing a dataset containing vulnerable source code databases, testing the error detection capabilities of OpenAI language models, and comparing these results with existing solutions offered by Snyk.

Particular emphasis will be placed on the use of soft-prompting and in-context learning techniques, which can help improve detection and results, even with limited resources. The work also envisages the implementation of an autonomous AI agent capable of analyzing code, performing security tests, and making decisions based on the results of these tests and context.

Optionally, the possibilities of error detection by open language models will be explored, and in the longer term, the possibilities of specializing models in the field of cybersecurity through fine-tuning. All these actions aim not only to understand but also to improve the capabilities of LLMs in the context of cybersecurity.

The main goal of the work is to increase awareness of the potential of large language models in cybersecurity and to propose practical solutions that can help developers create more secure applications.

SPIS TREŚCI

WPROWADZENIE

Niniejsza praca inżynierska nosi tytuł "Zastosowanie dużych modeli językowych do wykrywania i naprawiania błędów bezpieczeństwa i podatności w kodzie aplikacji webowych". Celem tej pracy jest zbadanie, w jaki sposób zaawansowane modele językowe, takie jak GPT-3.5, GPT-4 oraz modele otwarto-źródłowe - Mistral 7B, Falcon-7B-instruct, mogą być wykorzystane do automatycznego wykrywania i naprawiania błędów bezpieczeństwa w kodzie oprogramowania i aplikacji webowych. W tym celu zostanie opracowane i zaimplementowane narzędzie do statycznej analizy kodu, które będzie wykorzystywać modele językowe do generowania kodu i naprawy błędów. Narzędzie to zostanie przetestowane i porównane z innymi rozwiązaniami, takimi jak Snyk, które oferują podobne funkcjonalności. W pracy zostaną przedstawione wyniki badań, które mają na celu odpowiedzieć na pytanie, czy modele językowe mogą być wykorzystane do tego celu, oraz jak skuteczne są one w porównaniu z innymi rozwiązaniami. W ramach pracy zostaną również zbadane ograniczenia i wyzwania związane z wykorzystaniem tych technologii w kontekście cyberbezpieczeństwa.

PYTANIA BADAWCZE

W ramach pracy stawiam następujące pytania badawcze:

1. Czy duże modele językowe mogą być wykorzystane do wykrywania i naprawiania błędów bezpieczeństwa w kodzie aplikacji webowych?
2. Jak skuteczne są te modele w porównaniu z innymi rozwiązaniami?
3. Czy komercyjne modele językowe znacznie różnią się od otwartych modeli?
4. Jakie są ograniczenia i wyzwania związane z wykorzystaniem tych technologii w kontekście cyberbezpieczeństwa?

HIPOTEZY

Hipotezy pracy to:

1. Duże modele językowe, dzięki swojej zdolności do analizy i generowania kodu, mogą skutecznie identyfikować i naprawiać błędy bezpieczeństwa w kodzie źródłowym.
2. Mimo obiecującego potencjału, modele te mogą napotykać ograniczenia, szczególnie w bardziej złożonych i specyficznych scenariuszach związanych z cyberbezpieczeństwem.

UZASADNIENIE TYTUŁU

Tytuł pracy został dobrany tak, aby odzwierciedlał główny obszar zainteresowania badawczego, jakim jest wykorzystanie nowoczesnych technologii językowych w celu poprawy bezpieczeństwa aplikacji webowych. W kontekście rosnącej zależności od cyfrowych rozwiązań, temat ten zyskuje na znaczeniu, oferując nowe perspektywy i podejścia do zagadnień bezpieczeństwa. Tytuł można skrócić do **"Zastosowanie dużych modeli językowych w statycznej analizie kodu"**, ponieważ tak nazywa się problem odnajdywania i korekcji błędów w kodzie źródłowym. Korpus badawczy pracy został rozszerzony o projekty open-source aplikacji natywnych i desktopowych oraz wycinki błędnego kodu i poprawnego kodu.

OMÓWIENIE LITERATURY NAUKOWEJ I STOPNIA JEJ PRZYDATNOŚCI

Podstawę teoretyczną pracy stanowi literatura naukowa skupiająca się na dużych modelach językowych oraz ich zastosowaniu w cyberbezpieczeństwie. Szczególną uwagę poświęcono artykułowi "Can OpenAI Codex and Other Large Language Models Help Us Fix Security Bugs?", który posłużył jako punkt wyjścia dla badań. Praca ta ma na celu kontynuację i poszerzenie zakresu tych badań, wykorzystując literaturę naukową jako fundament do eksploracji nowych możliwości w zakresie analizy i naprawy błędów w kodzie. Różnica między pracą a literaturą naukową polega na tym, że praca skupia się na praktycznym zastosowaniu modeli językowych w celu wykrywania i naprawiania błędów bezpieczeństwa w kodzie, podczas gdy literatura naukowa skupia się na badaniu możliwości Sztucznej Inteligencji w tym zakresie.

CEL PRACY

Głównym celem pracy jest zbadanie możliwości wykorzystania dużych modeli językowych do wykrywania i naprawiania błędów bezpieczeństwa i podatności w kodzie źródłowym aplikacji webowych. W tym kontekście można wyróżnić następujące cele pośrednie:

— Opracowanie praktycznego rozwiązania do statycznej analizy kodu dla aplikacji webowych oraz lokalnych.
— Badanie skuteczności dużych modeli językowych w wykrywaniu podatności i luk bezpieczeństwa.

ZAKRES PRACY

Zakres pracy obejmuje:

— Analizę istniejącej literatury i badań, w szczególności artykułu ’Can OpenAI Codex and Other Large Language Models Help Us Fix Security Bugs?’.
— Projektowanie i implementacja narzędzia do statycznej analizy kodu opartego na modelach OpenAI.
— Przygotowanie zbioru danych z kodem zawierającym potencjalne podatności.
— Testowanie i porównanie skuteczności z innymi rozwiązaniami, np. oferowanymi przez firmę Snyk.
— Analiza wyników i formułowanie wniosków.

1. ANALIZA ISTNIEJĄCEJ LITERATURY ORAZ DOTYCHCZASOWYCH BADAŃ

1.1. CAN OPENAI CODEX AND OTHER LARGE LANGUAGE MODELS HELP US FIX SECURITY BUGS?

1.1.1. Metodologia

W badaniu "Czy OpenAI Codex i inne duże modele językowe mogą pomóc nam naprawić błędy bezpieczeństwa?", autorzy skupili się na wykorzystaniu dużych modeli językowych (LLM) do naprawy podatności w kodzie w sposób zero-shot. Badanie koncentrowało się na projektowaniu monitów skłaniających LLM do generowania poprawionych wersji niebezpiecznego kodu. Przeprowadzono eksperymenty na szeroką skalę, obejmujące różne komercyjne modele LLM oraz lokalnie wytrenowany model.

1.1.2. Wyniki

Wyniki wykazały, że LLM mogą skutecznie naprawić 100% syntetycznie wygenerowanych scenariuszy oraz 58% podatności w historycznych błędach rzeczywistych projektów open-source. Odkryto, że różne sposoby formułowania informacji kluczowych w monitach wpływają na wyniki generowane przez modele. Zauważono, że wyższe temperatury generowania kodu przynoszą lepsze wyniki dla niektórych typów podatności, ale gorsze dla innych.

1.2. EXAMINING ZERO-SHOT VULNERABILITY REPAIR WITH LARGE LANGUAGE MODELS

W artykule "Examining Zero-Shot Vulnerability Repair with Large Language Models", autorzy kontynuowali badanie możliwości wykorzystania LLM do naprawy podatności w kodzie, koncentrując się na wyzwaniach związanych z generowaniem funkcjonalnie poprawnego kodu w rzeczywistych scenariuszach. Badanie to rozszerzało wcześniejsze prace, biorąc pod uwagę bardziej złożone przypadki użycia LLM.

Podstawowe pytania badawcze były następujące:

1. Czy LLM mogą generować bezpieczny i funkcjonalny kod do naprawy podatności?
2. Czy zmiana kontekstu w komentarzach wpływa na zdolność LLM do sugerowania poprawek?

3. Jakie są wyzwania przy używaniu LLM do naprawy podatności w rzeczywistym świecie?
4. Jak niezawodne są LLM w generowaniu napraw?

Eksperymenty potwierdziły, że choć LLM wykazują potencjał, ich zdolność do generowania funkcjonalnych napraw w rzeczywistych warunkach jest ograniczona. Wyzwania związane z inżynierią promptów i ograniczenia modeli wskazują na potrzebę dalszych badań i rozwoju w tej dziedzinie.

1.3. RÓŻNICE MIĘDZY OBECNĄ PRACĄ A ISTNIEJĄCĄ LITERATURĄ

W przeciwieństwie do dotychczasowych badań skoncentrowanych głównie na teoretycznym potencjale dużych modeli językowych (LLM) w kontekście zero-shot, niniejsza praca dyplomowa podejmuje kroki w kierunku praktycznego zastosowania tych technologii. Główną różnicą jest tutaj zastosowanie metod takich jak Retrieval Augmented Generation (RAG) oraz in-context learning, co przesuwa nasze podejście w stronę kontekstu few-shot.

— **Zastosowanie Metod RAG i In-context Learning:** W odróżnieniu od tradycyjnych podejść zero-shot, które polegają na generowaniu odpowiedzi bez uprzedniego dostosowania modelu do specyficznego zadania, moja praca wykorzystuje RAG i uczenie się w kontekście, aby lepiej dostosować modele do konkretnych scenariuszy związanych z bezpieczeństwem kodu. Te metody pozwalają na bardziej precyzyjną analizę i naprawę błędów w kodzie.
— **Praktyczne Zastosowanie Modeli Językowych:** Podczas gdy większość istniejących badań skupia się na badaniu możliwości SI w teorii, ta praca koncentruje się na praktycznym zastosowaniu modeli językowych do wykrywania i naprawiania błędów bezpieczeństwa w kodzie. Przez to podejście, praca ta dostarcza bezpośrednich, aplikatywnych rozwiązań, które mogą być wykorzystane w rzeczywistych środowiskach programistycznych.

Takie podejście pozwala nie tylko na zrozumienie teoretycznego potencjału LLM, ale także na ocenę ich praktycznej przydatności w realnych scenariuszach związanych z cyberbezpieczeństwem. Znacząco poszerza to zakres badań w dziedzinie wykorzystania sztucznej inteligencji do poprawy bezpieczeństwa aplikacji, dostarczając nowych perspektyw i rozwiązań.

2. METODYKA

W niniejszej pracy dyplomowej zastosowano szereg metod i środków, aby zbadać i ocenić potencjał dużych modeli językowych w kontekście wykrywania i naprawiania błędów bezpieczeństwa w kodzie źródłowym aplikacji.

Metody:

— Algorytmy fragmentowania (własny kod)
— In-context learning (uczenie się w kontekście)
— Retrieval Augmented Generation - RAG (napisany własny kod, dostępny poprzez OpenAI Assistant API)
— Analiza porównawcza
— Programowanie obiektowe oraz funkcyjne

Środki:

— Modele językowe GPT-3.5, GPT-4
— Ollama - środowisko kontenerowe dla modeli językowych, łatwe w użyciu, otwartoźródłowe modele językowe
— OpenAI Assistant API
— Zbiór danych z kodem zawierającym podatności
— Statyczne testy podatności, np. CodeQL
— Istniejące rozwiązania komercyjne, np. Snyk
— Python 3.12
— Komputer osobisty

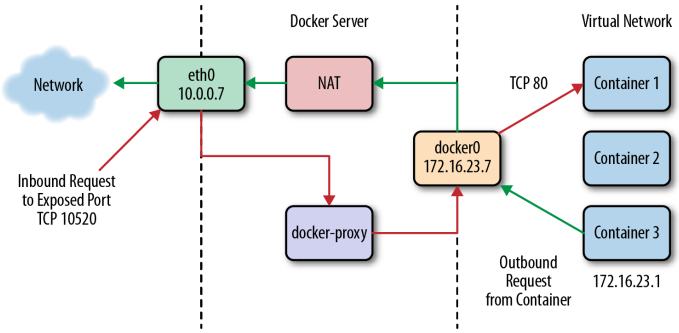
Metody i środki te zostały wybrane, aby zapewnić efektywne i wszechstronne podejście do analizy i naprawy kodu. Generacja wspomagana odzyskiwaniem danych (RAG ang. Retrieval Augmented Generation) oraz uczenie się w kontekście (in-context learning) umożliwiają efektywną analizę i generowanie kodu. Z kolei analiza porównawcza pozwala na ocenę skuteczności różnych modeli i podejść. Wykorzystanie modeli językowych GPT-3.5 i GPT-4, środowiska Ollama, oraz innych narzędzi i zasobów, zapewnia solidną bazę do przeprowadzenia kompleksowych testów i analiz.

2.1. RYSUNKI

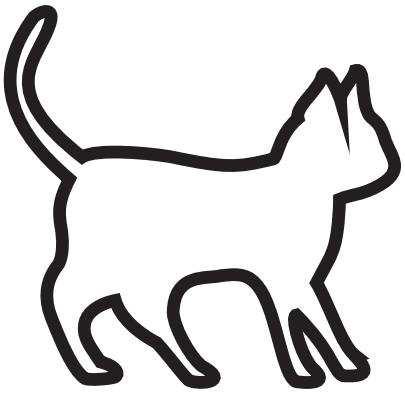
Na rysunku 2.1 …

2.1.1. Dwa rysunki obok siebie

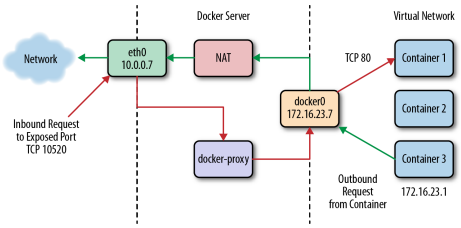
Na rysunkach 2.2 i 2.3 …



Rys. 2.1: Sieć dokera



Rys. 2.2: Lewy rysunek



Rys. 2.3: Prawy rysunek

2.2. TABELE

W tabeli 2.1 …

2.2.1. Równania

$$\sum_{i=1}^{\infty} a_i \tag{2.1}$$

W równaniu 2.1 …

2.3. LISTINGI

Na listingu 2.1 …

Tabela 2.1: Tytuł tabeli (patrz dodatek A)

Pierwszy	Drugi	Trzeci
Pierwszy	Drugi	Trzeci

```c
int main()
{
   int a=2*3;
   printf("**Ala ma kota\n**");
   while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT)); /* EV5 */
   return 0;
}
```

Listing 2.1: Język C

3. PROJEKTOWANIE I IMPLEMENTACJA ROZWIĄZANIA

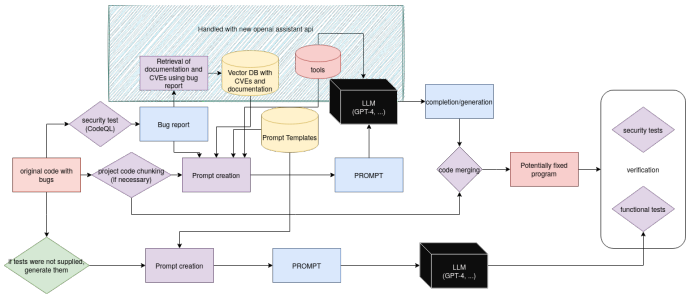
W ramach pracy dyplomowej opracowano aplikację `gptester`, , której celem jest statyczna analiza kodu w poszukiwaniu podatności bezpiec zeństwa oraz proponowanie ich napraw. Aplikacja wykorzystuje modele GPT-4, lub GPT-3.5 dostarczane przez OpenAI i jest zaprojektowana do działania z linii komend, z wynikami zapisywanymi w pliku markdown.

3.1. OPIS ROZWIĄZANIA

Program `gptester` jest narzędziem do statycznej analizy kodu, które korzysta z modelu GPT-4 do generowania raportów dotyczących jakości kodu i proponowania napraw. Głównym celem programu jest bezpieczeństwo kodu, które ma być w przyszłości rozszerzone. Aplikacja została napisana w języku Python, a model GPT-4 jest dostarczany przez OpenAI. Program jest przeznaczony do uruchamiania z linii poleceń, a wyniki są zapisywane w pliku markdown. Program można uruchomić z następującymi argumentami:

```
./main.py -h
usage: main.py [-h] [--model MODEL] [--input INPUT] [--output OUTPUT]
```

Na rysunku 3.1 ...



Rys. 3.1: Schemat blokowy działania aplikacji

PODSUMOWANIE

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

BIBLIOGRAFIA

SPIS RYSUNKÓW

SPIS LISTINGÓW

SPIS TABEL

Dodatki

A. DODATEK 1

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.