



Wykorzystanie dużych modeli językowych (LLM) w zaawansowanej statycznej analizie kodu

Damian Koterba

Promotor: dr inż. Wojciech Książek • Promotor pomocniczy: mgr inż. Maryna Łukaczyk

Politechnika Krakowska im. Tadeusza Kościuszki – Wydział Informatyki i Matematyki

Ograniczenia tradycyjnego SAST

- Narzędzia regułowe dopasowują wzorce, nie rozumieją semantyki i intencji kodu
- Trudności z językami dynamicznie typowanymi i złożonymi przepływami danych
- Wysoki odsetek fałszywych alarmów – reguły przeuczone na znanych wzorcach
- Zależność od języka – brak skalowalności na wielorodne bazy kodu

Dlaczego to ważne dla państwa

- Rosnąca liczba podatności w kodzie administracji i infrastruktury krytycznej
- Wysokie koszty ręcznego audytu bezpieczeństwa przy ograniczonych zasobach
- Presja na cyfrową transformację państwa przy jednoczesnym wzroście zagrożeń
- Potrzeba skalowalnych, językowo-niezależnych narzędzi bezpieczeństwa

Hipoteza: modele LLM mogą już zastąpić tradycyjne narzędzia SAST w wykrywaniu podatności

PYTANIE BADAWCZE

Czy modele LLM mogą być skuteczniejsze od tradycyjnych narzędzi SAST w wykrywaniu podatności w kodzie?

Java

Wybrany ze względu na dostępność narzędzi SAST

Poziom pliku

Bez analizy międzymodułowej

Obecność podatności

Bez klasyfikacji typu CWE

WKŁAD WŁASNY

Implementacja procesu oceny, przygotowanie zbioru CUT oraz analiza porównawcza 34 rozwiązań (30 modeli LLM + 4 narzędzia SAST)

30

Modeli LLM

OpenAI, Google, Anthropic, Mistral, Meta, DeepSeek, xAI (Grok), rodzina Qwen

2740

Plików Java

OWASP Benchmark v1.2, standard SAST publikowany od 2016 r.

242

Plików CUT

Niezależny zbiór walidacyjny, nieopublikowany (Politechnika Krakowska)

F1

Metryki oceny

Precyzja, czułość, F1-score, koszt API (USD), czas inferencji (s/plik)

11 klas podatności objętych badaniem

SQLi • XSS • CMDi • Path Traversal • LDAPi • XPathi • SecureCookie • Crypto • WeakRand • TrustBound • Hash

Zestaw obejmuje klasy podatności zdefiniowane w standardzie OWASP Benchmark, powszechnie stosowanym w ocenie narzędzi SAST

#	Model	Typ	F1
1	Gemini 3 Pro	LLM	0,88
2	GPT-5 Codex	LLM	0,83
3	Grok 4 Fast Reasoning	LLM	0,80
4	Grok Code Fast 1	LLM	0,78
5	O3 Mini Medium	LLM	0,77
6	O3 Mini Low	LLM	0,76
—	Semgrep	SAST	0,66
—	ShiftLeft Scan	SAST	0,60
—	FindSecBugs	SAST	0,56
—	CodeQL	SAST	0,48

Gemini 3 Pro osiąga F1 wyższy o 33% od najlepszego narzędzia SAST (Semgrep)

Top 6 spośród 30 ocenionych modeli LLM – pełne zestawienie w publikacji naukowej

242

pliki Java w zbiorze CUT

Kod studentów, nieopublikowany –
eliminuje efekt zapamiętywania danych

0,75

**F1 najlepszych modeli na
nieznanym kodzie**

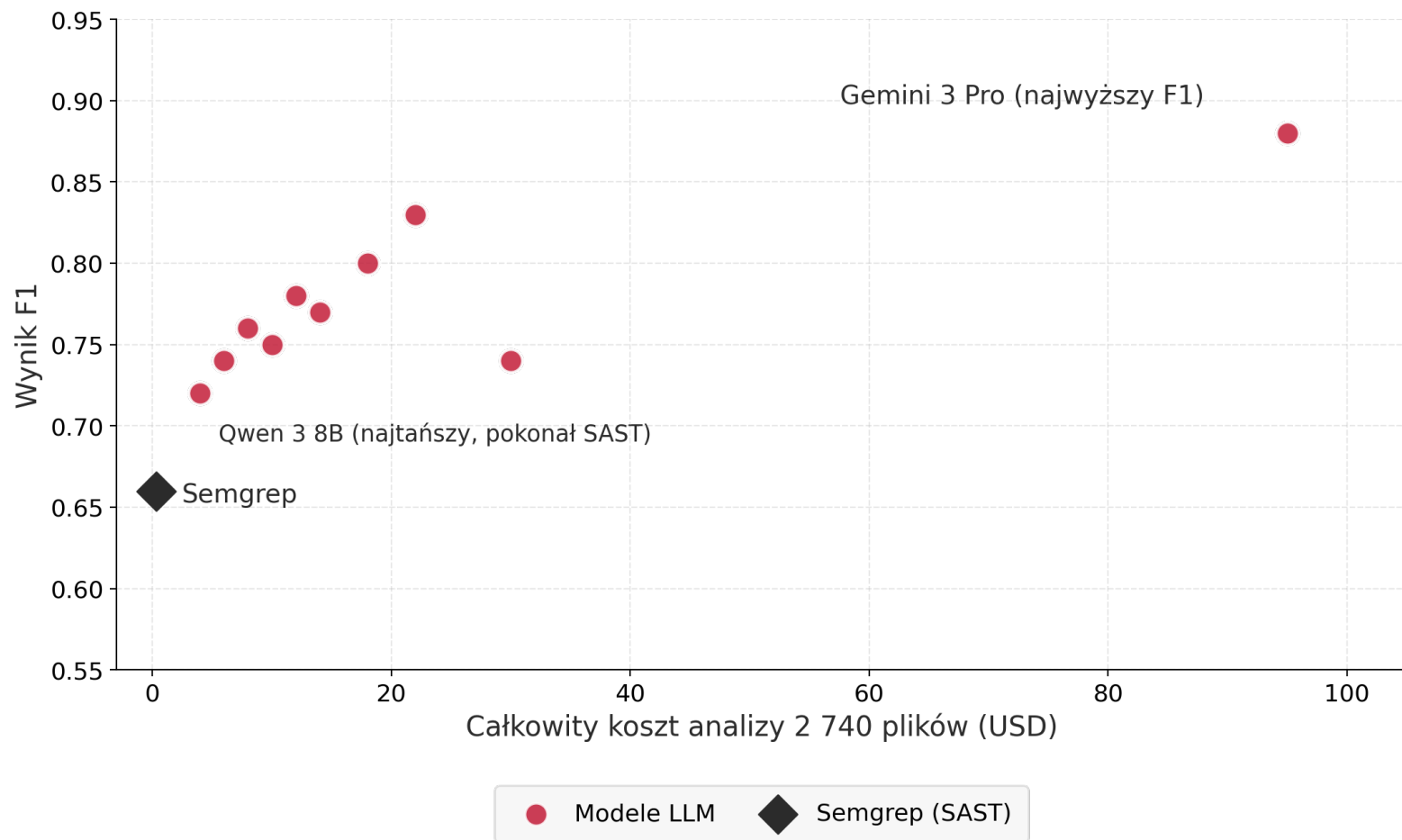
Gemini 3 Pro i Grok Code Fast 1 – wynik
utrzymany na nowych danych

0,66 → 0,42

spadek F1 narzędzia Semgrep

Sztywne reguły SAST nie generalizują na
nowy, nieznaną kod

Kluczowy wniosek: modele LLM wykrywają podatności w kodzie, którego nigdy wcześniej nie widziały



Dane: Tabela 4 publikacji – koszt analizy 2 740 plików Java

Kluczowe obserwacje

- Tańsze modele (O3 Mini, GPT-5 Codex) osiągają wysokie F1 ułamkiem kosztu Gemini 3 Pro
- Nadal przewyższają skutecznością Semgrep przy znacznie niższym koszcie analizy
- Nawet najtańszy Qwen 3 8B pokonał wszystkie 4 narzędzia SAST
- Wybór modelu to kompromis między maksymalnym F1 a budżetem projektu

RECENZOWANA PUBLIKACJA NAUKOWA

Leveraging Large Language Models for Advanced Static Code Analysis

Assessing the Feasibility of AI Superseding Traditional Vulnerability Detection Methods

Information and Software Technology (Elsevier) • tom 198 (2026), artykuł 108213

Autorzy: Damian Koterba, Maryna Łukaczyk, Wojciech Książek – Politechnika Krakowska im. Tadeusza Kościuszki

DOI: 10.1016/j.infsof.2026.108213 • przyjęte do druku: 30 maja 2026

„Wyniki wskazują, że w zastosowaniach praktycznych podejścia oparte na LLM mogą przewyższyć tradycyjne narzędzia, a z dalszym rozwojem – nawet ręczny przegląd kodu w wykrywaniu podatności”

1

Audyt kodu administracji publicznej

Szybsze i tańsze skalowanie przeglądu bezpieczeństwa niż ręczny audyt ekspercki

2

Niezależność językowa

Jeden model zamiast osobnych reguł dla każdego języka w systemach urzędowych

3

Hybrydowy pipeline SAST + LLM

Integracja w CI/CD infrastruktury krytycznej – szybka reguła + głębokie rozumienie kontekstu

4

Nowy wektor ryzyka do zaadresowania

Wstrzykiwanie promptów (prompt injection) w systemach LLM-SAST wymaga dalszych zabezpieczeń

Najważniejsze wnioski

- LLM przewyższają narzędzia SAST: najlepszy F1=0,88 wobec 0,66 dla Semgrep (+33%)
- Na nieznanym kodzie (CUT) Semgrep spada do 0,42, a LLM utrzymują skuteczność
- Niezależność od języka i rozumienie semantyki dają skalowalny SAST
- Nowy wektor ryzyka: wstrzykiwanie promptów w systemach LLM-SAST

Kierunki przyszłych badań

- Testy na rzeczywistych projektach, nie tylko syntetycznych benchmarkach
- Hybrydowa integracja SAST + LLM w pipeline'ach CI/CD
- Rozszerzenie na Python, JavaScript, Go – ocena wielojęzyczna
- Zaawansowane klasy: SSRF, niebezpieczna deserializacja, race conditions

Wyniki wskazują, że w zastosowaniach praktycznych LLM mogą już zastąpić klasyczne narzędzia SAST