

МЕТОД КОНТЕКСТНОГО ВИЯВЛЕННЯ СЕКРЕТІВ У ПРОГРАМНОМУ КОДІ НА ОСНОВІ МОВНОЇ МОДЕЛІ

Вінницький національний технічний університет

Анотація

Запропоновано метод виявлення конфіденційних даних у програмному коді, який поєднує попередній відбір потенційно небезпечних значень за показником ентропії та подальший аналіз контексту за допомогою спеціалізованої мовної моделі. Проведено тестування нейромережі, адаптованої до завдання класифікації фрагментів коду, для демонстрації ефективності методу.

Ключові слова: конфіденційні дані, мовна модель, ентропія, тренування нейромережі, кероване донавчання.

Abstract

A method for detecting confidential data in program code is proposed, which combines preliminary selection of potentially dangerous values based on entropy and subsequent context analysis using a specialized language model. A neural network adapted to the task of classifying code fragments was tested to demonstrate the effectiveness of the method.

Keywords: confidential data, language model, entropy, neural network training, supervised training.

Вступ

У сучасному процесі розробки програмного забезпечення (ПЗ) повсюдно використовується секретна інформація, зокрема облікові дані, токени доступу, криптографічні ключі. Неправильне управління такими даними є однією з поширених причин компрометації інформаційних систем. Однією з найбільш розповсюджених проблем є включення секретів безпосередньо у вихідний код або конфігураційні файли. Оскільки зловмисники здатні отримати доступ до оригінального коду з допомогою різних методів зворотної розробки (декомпіляція, дизасемблювання), виникає загроза витоку секретних даних [1].

Для виявлення конфіденційних даних, які зберігаються у відкритому вигляді, застосовуються інструменти статичного аналізу коду, які переважно використовують сигнатурний підхід на основі регулярних виразів [2]. Подібні засоби ефективно знаходять секрети, що відповідають відомим шаблонам, однак ігнорують дані нестандартного формату або схожі на звичайні слова, які недоцільно позначати як потенційно небезпечні через високу ймовірність помилкових спрацювань.

Одним із перспективних напрямів підвищення ефективності такого аналізу є використання методів машинного навчання. Мовні моделі здатні враховувати семантичні особливості програмного коду та взаємозв'язки між його елементами, що дозволяє виявляти потенційні секрети навіть у випадках, коли вони не відповідають типовим сигнатурам [3].

Результати дослідження

Запропонований метод контекстного виявлення секретів у програмному коді складається з декількох послідовних етапів. Основною ідеєю є попередній відбір потенційно підозрілих фрагментів коду за значенням ентропії з подальшим аналізом їх контексту за допомогою мовної моделі. Такий підхід дозволяє зменшити обсяг програмного коду, що підлягає детальному дослідженню, та зосередити обчислювальні ресурси на найбільш ймовірних кандидатах.

На першому кроці вихідний код розбивається на окремі слова, які доцільно відфальтувати за довжиною понад 7 символів. Для кожного слова з отриманого списку обчислюється значення

ентропії Шеннона, яке дозволяє оцінити ступінь випадковості послідовності символів. Ентропія визначається за формулою

$$H(X) = - \sum_{i=1}^n p(x_i) * \log_2 p(x_i)$$

(1)

де X – послідовність символів; x_i – окремий символ послідовності; p – ймовірність появи x_i в послідовності; n – кількість унікальних символів у послідовності.

Високі значення ентропії характерні для випадкових або псевдовипадкових послідовностей символів, які часто використовуються у токенах доступу, криптографічних ключах та інших секретних даних.

Якщо значення ентропії перевищує заданий поріг, відповідний рядок вважається підозрілим та зберігається для подальшого дослідження. Для такого слова формується контекстний блок, який містить декілька рядків до та після аналізованого елемента. Використання контексту дозволяє врахувати структуру програми, назви змінних та логіку використання.

Створені фрагменти передаються до спеціалізованої мовної моделі, донавченої для задачі класифікації програмного коду. Модель аналізує отриманий блок та визначає ймовірність того, що відповідний рядок містить секретні дані. На основі отриманої оцінки приймається рішення щодо ризику потенційного витоку секретної інформації.

Для перевірки ефективності запропонованого методу було донавчено і протестовано мовну модель. Під час вибору базової нейронної мережі було враховано розмір контекстного вікна, а також віддано перевагу моделям з відкритим кодом над комерційними.

Перелік доступних моделей приведено в табл. 1.

Таблиця 1. Порівняння мовних моделей

Модель	Тип доступу	Розмір контекстного вікна
Llama 3.2 3B	відкрита	128 тис. токенів
Llama 4 109B	відкрита	10 міль. токенів
Mistral Large 3 41B	відкрита	256 тис. токенів
GPT-5	комерційна	400 тис. токенів
Claude Haiku 4.5	комерційна	64 тис. токенів

Для розв’язання поставленої задачі достатньо моделі з відносно невеликим контекстним вікном, оскільки аналіз не потребує складних багаторівневих логічних висновків. Натомість, вдається заощадити обчислювальні ресурси під час тренування і використання моделі. З огляду на це, у дослідженні використано мовну модель Llama 3.2 3B.

Навчальний набір даних було сформовано на основі відкритої бази даних конфіденційної інформації, опублікованої компанією Samsung [4]. Перелік містить широкий набір API-ключів до різних сервісів, облікових даних, геш-значень та іншого.

Було створено список зі 100 фрагментів. Окремі блоки містять послідовності символів, які фактично не використовуються як конфіденційні дані, і не повинні позначатися моделлю як небезпечні. Для кожного прикладу було створено запис, що складається з системної інструкції, вхідних даних і еталонної відповіді. Частина навчального набору зображено на рис. 1.

```
{
  "instruction": "Analyze the code snippet and determine if it contains hard-coded credentials",
  "input": "import requests\nBASE_URL = \"https://api.example.com\"\n\nUSER = \"admin\"\nPASSWORD = \"P@ssw0rd123!\"\n\ndef login():\n    payload\n  \"output\": \"hasCredential: true; confidenceScore: 0.99\"
},
{
  \"instruction\": \"Analyze the code snippet and determine if it contains hard-coded credentials\",
  \"input\": \"const axios = require('axios');\nconst API_URL = 'https://service.example.com';\nconst API_KEY = 'sk_live_51N8ExampleKeyValue1'\n  \"output\": \"hasCredential: true; confidenceScore: 0.98\"
},
{
  \"instruction\": \"Analyze the code snippet and determine if it contains hard-coded credentials\",
  \"input\": \"public class Config {\n    private static final String DB_URL = \"jdbc:mysql://localhost:3306/app\";\n    private static final St\n  \"output\": \"hasCredential: true; confidenceScore: 0.95\"
},
{
  \"instruction\": \"Analyze the code snippet and determine if it contains hard-coded credentials\",
  \"input\": \"import os\nimport psycopg2\nDB_HOST = 'localhost'\nDB_USER = os.getenv('DB_USER')\nDB_PASSWORD = os.getenv('DB_PASSWORD')\n  \"output\": \"hasCredential: false; confidenceScore: 0.97\"
},
{
  \"instruction\": \"Analyze the code snippet and determine if it contains hard-coded credentials\",
  \"input\": \"package main\nimport \"fmt\"\nfunc main() {\n    username := \"test_user\"\n    password := \"CHANGE_ME\"\n    fmt.Println(\"Star\n  \"output\": \"hasCredential: false; confidenceScore: 0.90\"
},
}
```

Рис. 1. Фрагмент тренувального набору даних

Навчання проведено із застосуванням методу квантованої низькорангової адаптації (КНА) [5]. Він дозволяє тренувати великі мовні моделі без зміни більшості їх початкових параметрів. Згідно з методом, створюються нові низькорангові матриці. Під час навчання оригінальні ваги нейронної мережі заморожуються, і для коригування поведінки моделі змінюються тільки додані значення. Таким чином вдається значно заощадити обчислювальні ресурси, необхідні для навчання.

Процедура тренування реалізована з допомогою python-пакета SFTTrainer. Інструмент використовується для проведення керованого донавчання [6]. Для прозорості процесу SFTTrainer обчислює значення функції втрат. Зі зменшенням отриманого числа зростає передбачуваність поведінки мережі.

Тренування тривало 60 ітерацій, протягом яких індекс втрат зменшився на 89%. Графік зміни значення функції втрат зображено на рис. 2.

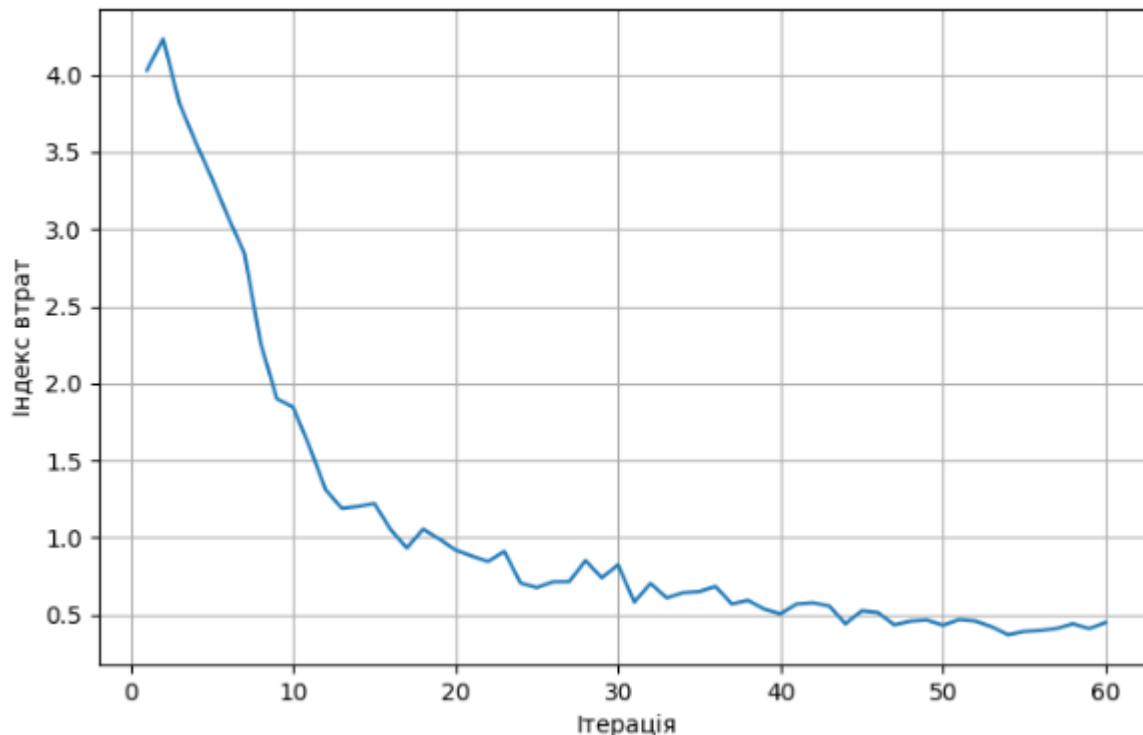


Рис. 2. Графік зміни функції втрат

Після завершення процедури донавчання було проведено тестування моделі на окремому наборі фрагментів програмного коду. Модель формує відповідь у наперед визначеному структурованому форматі незалежно від формулювання запиту. Результат аналізу подається у вигляді рядка: “hasCredential: true; confidenceScore: 0.89”, де hasCredential позначає наявність секретних даних у фрагменті і набуває значень “true” або “false”, а confidenceScore відображає оцінку впевненості моделі і набуває значень від 0 до 1. Додаткове числове значення дозволяє зробити точніші висновки щодо ризиків витоку конфіденційної інформації.

Приклади роботи моделі приведені у табл. 2.

Таблиця 2. Приклади результатів аналізу фрагментів коду

Фрагмент коду	Результат аналізу
API KEY = "sk-85a9e94d"	HasCredential: true; confidenceScore: 0.92
PASS = "admin12345678"	HasCredential: true; confidenceScore: 0.75
PUBLIC = "983514132"	HasCredential: false; confidenceScore: 0.89
random = "idm3213nen12"	HasCredential: false; confidenceScore: 0.92

Було проведено тестування моделі на новоствореному наборі зі 100 блоків програмного коду. Узагальнені результати оцінки блоків мережею наведено в табл. 3.

Таблиця 3. Результати тестування моделі

Фактичний клас	Передбачено: секрет	Передбачено: не секрет
Секрет	54	9
Не секрет	6	29

Для кількісної оцінки ефективності моделі на основі отриманих результатів обчислено основні метрики якості класифікації. Точність класифікації характеризує частку правильних рішень від загальної кількості тестів і визначається за формулою

$$A = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

де TP - правильне позитивна оцінка; TN - правильне негативна оцінка; FP - хибна позитивна оцінка; FN - хибна негативна оцінка.

Прецизійність характеризує частку правильних позитивних оцінок серед усіх випадків, у яких модель повідомила про наявність секрету. Величина визначається за формулою

$$P = \frac{TP}{TP + FP} \quad (3)$$

Згідно з результатами тестування, значення точності становить 0.83 (83%), а значення прецизійності – 0.83 (83%). Такі показники свідчать про здатність натренованої моделі коректно визначати фрагменти програмного коду, що містять конфіденційні дані і відрізняти їх від звичайних символьних послідовностей.

Висновки

Встановлено, що застосування спеціалізованої мовної моделі для аналізу фрагментів програмного коду дозволяє ефективно виявляти секретні дані, які зберігаються у відкритому вигляді. Попередній відбір рядків за показником ентропії значно знижує кількість фрагментів, що потребують детальної перевірки та дозволяє зосередити обчислювальні ресурси на найбільш ймовірних кандидатах.

Результати тестування показали, що донавчена мовна модель стабільно генерує відповіді у заданому форматі та демонструє здатність розрізняти реальні секретні значення і звичайні символьні послідовності з подібними статистичними характеристиками.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. McGowan V., Ventura M., Bell P. Engineering Encounters: Reverse Engineering. Science and Children. 2017. Т. 054, № 08. URL: https://doi.org/10.2505/4/sc17_054_08_68 (дата звернення: 11.03.2026).
2. Segura T. The nightmare of hard-coded credentials. Network Security. 2023. Т. 2023, № 2. URL: [https://doi.org/10.12968/s1353-4858\(23\)70010-x](https://doi.org/10.12968/s1353-4858(23)70010-x) (дата звернення: 11.03.2026).
3. Biringa C., Kul G. Detecting Hard-Coded Credentials in Software Repositories via LLMs. Digital Threats: Research and Practice. 2025. URL: <https://doi.org/10.1145/3744756> (дата звернення: 11.03.2026).
4. CredData. Samsung Electronics Co., Ltd. URL: <https://github.com/Samsung/CredData> (дата звернення: 11.03.2026).
5. QLoRA: Efficient Finetuning of Quantized LLMs / T. Dettmers та ін. Advances in Neural Information Processing Systems 36, м. New Orleans, Louisiana, USA, 10–16 груд. 2023 р. San Diego, California, USA, 2023. С. 10088–10115. URL: <https://doi.org/10.52202/075280-0441> (дата звернення: 11.03.2026).
6. Dutta S., Ganapathy S. LLM supervised Pre-training for Multimodal Emotion Recognition in Conversations. ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), м. Hyderabad, India, 6–11 квіт. 2025 р. 2025. С. 1–5. URL: <https://doi.org/10.1109/icassp49660.2025.10889998> (дата звернення: 11.03.2026).

Ігор Володимирович Сітніков – студент групи 2БС-226, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: ihor.sitnikov05@gmail.com.

Віталій Володимирович Лукічов – канд. техн. наук, доцент кафедри захисту інформації, Вінницький національний технічний університет, м. Вінниця.

Ihor Sitnikov – Department of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: ihor.sitnikov05@gmail.com.

Vitalii Lukichov – Ph.D., Associate Professor of the Department of Information Protection, Vinnytsia National Technical University, Vinnytsia.