УДК 004.056.5:004.8

Любченко Максим Сергійович,

здобувач другого (магістерського) рівня вищої освіти
Харківського національного університету радіоелектроніки

Смірнов Антон Олександрович,

к.т.н. доцент кафедри безпеки інформаційних технологій
Харківського національного університету радіоелектроніки

ГІБРИДНИЙ ПІДХІД ДО АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ПРОГРАМНОГО КОДУ НА ОСНОВІ CODEBERT

Анотація. Розглянуто підхід до автоматизованого виявлення потенційних вразливостей у програмному коді C/C++, який поєднує контекстний аналіз нейромережевою моделлю CodeBERT та евристичну перевірку ознак безпечного програмування. Описано підготовку навчальних даних на основі набору Draper VDISC, донавчання моделі для розпізнавання класів CWE та формування результатів програмним сканером. Визначено призначення індивідуальних порогів класифікації та перевірки захисних конструкцій для зниження кількості необґрунтованих сповіщень. Розмежовано результати функціонального тестування на контрольних прикладах і необхідність незалежного кількісного оцінювання якості виявлення. Обґрунтовано використання розробленого рішення як допоміжного інструмента статичного аналізу в процесах безпечної розробки програмного забезпечення.

Ключові слова: вразливість, програмний код, CodeBERT, машинне навчання, Transformer, CWE, статичний аналіз, DevSecOps.

Своєчасне виявлення вразливостей у вихідному коді є складовою забезпечення кіберстійкості програмних систем. Для застосунків, написаних мовами C та C++, особливо небезпечними залишаються помилки

керування пам’яттю: вихід за межі буфера, некоректні операції з покажчиками та розіменування нульового покажчика. Такі дефекти не завжди можна надійно визначити простими текстовими шаблонами, оскільки значення окремої інструкції залежить від її програмного контексту. Традиційний статичний аналіз безпеки (SAST) є необхідним, проте може формувати хибні сповіщення або пропускати окремі контекстно-залежні помилки.

Проблема управління вразливостями як складова політики інформаційної безпеки висвітлена в дослідженні [1]. Для аналізу вихідного коду доцільно доповнювати правила статичної перевірки моделями, які враховують взаємозв’язки між програмними конструкціями. Архітектура CodeBERT, запропонована Z. Feng та співавторами [2], формує контекстні представлення програмних послідовностей на основі Transformer, однак для завдання виявлення вразливостей потребує спеціалізованого донавчання та незалежної перевірки.

Метою дослідження є обґрунтування та реалізація гібридного підходу до пошуку потенційних вразливостей програмного коду, у якому прогноз CodeBERT поєднується з контрольованою евристичною постобробкою. На відміну від суто нейромережевої класифікації, запропонований підхід передбачає врахування захисного контексту, наприклад перевірки вказівника перед зверненням до пам’яті або контролю довжини даних перед копіюванням. Водночас наявність такої конструкції не є безумовним доказом відсутності вразливості.

Для навчання було використано набір Draper VDISC із фрагментами програм C/C++. У процесі попередньої обробки даних сформовано вибірку зі 100 000 функцій програмного коду мовами C/C++ із мітками класів вразливостей CWE-119 (порушення меж буфера пам’яті), CWE-120 (копіювання даних без перевірки розміру буфера) [3], CWE-469 (використання віднімання покажчиків для визначення розміру) та CWE-476

(розіменування нульового покажчика) [4]. Вхідні фрагменти очищувалися та перетворювалися токенізатором CodeBERT у послідовності, придатні для нейромережевого опрацювання.

Донавчання моделі microsoft/codebert-base організовано засобами Python, бібліотек PyTorch і Transformers. Вихід моделі інтерпретується як оцінка належності фрагмента до відповідних класів CWE. Для кожного класу передбачено окремий поріг прийняття рішення. Такий вибір обґрунтований різною поширеністю класів у навчальних даних і відмінною вартістю хибнопозитивних та хибнонегативних результатів. Під час оцінювання використовуються Precision, Recall і F1-score, вони характеризують відповідно частку правильних позитивних прогнозів, повноту виявлення та їх збалансоване співвідношення.

Реалізований сканер приймає вихідний файл, виконує підготовку його вмісту, застосовує модель і пороги класифікації, а потім формує звіт щодо потенційно небезпечних конструкцій. Додатковий евристичний етап аналізує ознаки безпечного програмування, які можуть пояснювати певні спрацювання. Результати аналізу програмного коду представляються у структурованому вигляді із зазначенням типу потенційної вразливості, розташування підозрілого фрагмента коду та оцінки впевненості нейромережевої моделі у сформованому прогнозі». Така оцінка є виходом моделі, а не самостійним доказом експлуатованості знайденої помилки.

Функціональне тестування проведено на двох контрольних прикладах мовою C: навмисно вразливому фрагменті з небезпечним копіюванням даних і зверненням до NULL-покажчика, а також прикладі із захисними перевірками. Для першого випадку сканер сформував попередження про потенційні дефекти; для другого критичних попереджень не сформовано та зафіксовано фільтрацію окремих спрацювань. Ці результати підтверджують працездатність програмного прототипу в перевірених сценаріях, але не доводять універсальної точності методу на довільних проєктах.

Науково-прикладний результат полягає в узгодженні двох різних механізмів аналізу, статистичного контекстного прогнозування CodeBERT та інтерпретованих правил постобробки з індивідуальними порогами для класів CWE. Застосування підходу може бути корисним для попереднього ранжування підозрілих фрагментів у конвеєрах Secure SDLC і DevSecOps.

Отже, розглянутий гібридний підхід забезпечує технологічну основу для допоміжного автоматизованого аналізу безпеки програмного коду та формування пояснюваних сповіщень. Подальше дослідження має передбачати вимірювання Precision, Recall, F1-score й частоти хибних спрацьовувань на незалежних вибірках, перевірку стійкості до змін стилю програмування та оцінювання часу аналізу.

Список використаних джерел

1. Поддубний В.О., Сєвєрінов О.В., Пустомельник О.С. Менеджмент вразливостей як складова частина політики безпеки ІТС. Системи управління, навігації та зв'язку. 2020. Т. 4, № 62. С. 55–58. DOI: https://doi.org/10.26906/SUNZ.2020.4.055.

2. Feng Z., Guo D., Tang D. et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. Findings of the Association for Computational Linguistics: EMNLP 2020. 2020. P. 1536–1547. DOI: https://doi.org/10.18653/v1/2020.findings-emnlp.139.

3. MITRE. CWE-120: Buffer Copy without Checking Size of Input (Classic Buffer Overflow). URL: https://cwe.mitre.org/data/definitions/120.html (дата звернення: 29.10.2026).

4. MITRE. CWE-476: NULL Pointer Dereference. URL: https://cwe.mitre.org/data/definitions/476.html