УДК 004.056.55

Драгін Д., Садченко А.

Національний університет «Одеська політехніка»

РОЗРОБКА ЛОКАЛЬНОЇ МОДЕЛІ МАШИННОГО НАВЧАННЯ ЩОДО ЗАХИСТУ КОНФІДЕНЦІЙНОЇ ІНФОРМАЦІЇ У ВІДКРИТОМУ ПРОГРАМНОМУ КОДІ

Вступ. В сучасному цифровізованому світі, ми кожного дня використовуємо велику кількість різних застосунків, які використовують, зберігають або передають конфіденційну інформацію. Безпека цих даних з кожним роком стає все більш важливим питанням і розглядається не тільки, як проблема особистої безпеки або компанії, а інколи може досягати загальносвітового рівня.

Загалом, добре побудова інформаційна система в технічному плані немає слабких місць, які зловмисник міг би використати для незаконного доступу до конфіденційної інформації. Проте під час створення застосунку, розробники можуть залишити у відкритому доступі конфіденційну інформацію (API ключі, токени доступу, паролі до баз даних), що робить їх вразливими до автоматизованих сканерів, які постійно переглядають платформи розробки, такі як GitHub, GitLab, npm або PyPi, у пошуках витоків, для несанкціонованого доступу до персональної інформації користувача або злому системи.

Мета: розробка локальної моделі машинного навчання для захисту конфіденційної інформації у відкритому програмному коді для забезпечення превентивного захисту від витоку секретної інформації.

Основна частина

Згідно зі звітом GitGuardian за 2023 рік, у публічних репозиторіях було виявлено 12 778 599 нових секретів, з яких 3 698 686 є унікальними, і ця кількість з кожним роком лише зростає. Для порівняння у 2021 році - 6 мільйонів, а у 2022 році - 10 мільйонів, що на 28% менше, ніж у 2023 році [1].

Згідно зі звітом Агентства з кібербезпеки та інфраструктури безпеки США (CISA) за 2022 рік, у більш ніж 54% випадків незаконного доступу до систем було отримано через компрометацію конфіденційної інформації, в той час як експлуатація вразливостей становила лише 1% [2].

Для вирішення цієї проблеми було створено багато рішень, але більшість ініціатив в цій проблемі фокусуються вже на аналізі репозиторіїв після завантаження коду, що в деяких випадках може бути запізно. Про що свідчіть дослідження Cybenari, компанії що спеціалізується на кібербезпеці та тестуванні ПО, для виявлення та використання чутливої інформації на GitHub треба лише 127 секунд. Набагато гірші результати має менеджер пакетів і репозиторій для JavaScript і Node.js – npm, для якого зловмисникам треба лише 60 секунд.

Також важливо пам'ятати, що npm і PyPi не надають функцію автоматичного сканування коду та оповіщення та знайдену чутливу інформацію в коді, що робить захист від витоку інформації на цих платформах майже неможливим [3].

Для побудови цієї моделі використовується логістична регресія, яка була спеціально розроблена для задач класифікації, що дозволяє їй ефективно розподіляти об'єкти між класами на основі ймовірнісного підходу.

Для визначення моделі логістичної регресії, вводиться певна випадкова величина Y, що набуває значення від 0 до 1. Найчастіше 0 відповідає за те, що певний об'єкт не відповідає певному класу, а 1 – навпаки. Об'єкт класифікується шляхом порівняння отриманої ймовірності з пороговим значенням. Логістична регресія дозволяє оцінити вплив змінних на ймовірність належності до певного класу. Результатом є ймовірності для кожного класу, що допомагають приймати рішення про класифікацію [3].

Ця величина залежить від певної множини змінних, які впливають на те, яке значення буде приймати змінна Y.

$$x = (1, x_1, \dots, x_{n)^T}, \tag{1}$$

Для того, щоб отримати залежність змінної Y від вектору пояснювальних змінних, вводиться додаткова прихована змінна y*, яка відповідає за лінійний результат.

$$y^* = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n + \varepsilon, \tag{2}$$

Ця зміна є лінійною комбінацією параметрів θ, що визначають вплив кожної ознаки x, до яких додається випадкова похибка ε, що зазвичай є підпорядкованою логістичному розподілу та є випадковою величиною з певним логістичним розподілом ймовірностей.

Тому залежність Y від y^* має вигляд:

$$Y = \begin{cases} 0, & y^* \le 0 \\ 1, & y^* > 0 \end{cases}, \tag{3}$$

Для перетворення лінійного результату у вірогідність використовується сигмоїдна функція. Сигмоїдна функція є математичною функцією, яка має S-подібну (sigmoid) форму. Її основна мета - перетворення будь-якого дійсного числа в значення в інтервалі від 0 до 1 [4].

$$\sigma(y^*) = \frac{1}{1+e^{-y^*}} \tag{4}$$

Додатково ефективність системи було підвищено за рахунок гібридних підходів, що поєднують методи обробки тексту (TF-IDF, регулярні вирази) з алгоритмами машинного навчання.

Під час тестування системи було виявлено, що для сканування 38000 файлів(розмір проєкту сягає 263 МБ та містить 30 конфіденційних токенів) займає приблизно 64 секунди, в залежності від розмірів файлів, а також кількості секретної інформації в них, що є кращим результатом ніж той, що надає GitHub. Повідомлення про наявність секретної інформації в коді надійшла приблизно через 43 секунди після завантаження.

Ще одним недоліком GitHub є те, що він не проводить сканування закритих репозиторіїв, а лише відкритих, що дає можливість зловмисникам отримати секретну інформацію до того, як буде отримано повідомлення про неї.

Також GitHub не проводить евристичний аналіз, через що не було знайдено всі паролі в програмному коді, які в свою чергу знайшла локальна модель.

Після сканування директорії було визначено, що моделі потрібно 1.5 мс для сканування 1 файлу. Модель відмітила 42 токени, які потенційно можуть містити конфіденційну інформацію, з яких 25 токенів дійсно є такою інформацією, проте 17 токенів були відмічені хибно, шо свідчить про те, що локальна модель демонструю певні ознаки надмірності в свої рішеннях, коли рядки які не є конфіденційною інформацію вона помічає такими, що потенційно можуть бути такими.

Вирішенням цієї проблеми на перших етапах є збільшення навчальної вибірки кількість даних для навчання моделі, а також додавання нових функцій обробки тексту та модернізація вже наявних методів. Проте вже отримані результати свідчать про певний успіх в розробці моделі.

Висновок. Таким чином, можна зробити висновок про те, що в сучасних системах увагу треба приділяти не тільки класичним методам захисту інформації, а також шляхам вирішення проблеми людського фактору. Дана модель забезпечує захист та аналіз відкритого програмного коду в умовах обмежених ресурсів, а також з забезпечення потрібної швидкості обробки та надійності, про що свідчать результати тестування моделі, за допомогою поєднання різних методів обробки тексту з алгоритмами машинного навчання.

Перелік використаних джерел.

1. The State of Secrets Sprawl Report. GitGuardian. 2024.

2. Active Adversary for Tech Leaders. Sophos News. 2022.

3. You Have One Minute to Save Your Leaked AWS Credentials ThreatDown Blog. 2023.

4. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. – Springer, 2001. – 533 p.