Ризики компрометації API-ключів у сервісах Google Cloud із використанням генеративного ШІ та методи їх мінімізації

УДК 004.056.5:004.8

Михайло Рокош[1]

Тернопільський національний технічний університет імені Івана Пулюя,
[1]mykhailo.rokosh@tntu.edu.ua

Активне впровадження генеративного штучного інтелекту у бізнес-процеси, розроблення прототипів і внутрішню автоматизацію створює не лише нові можливості, а й нові ризики для інформаційної та фінансової безпеки. У зоні найбільшого ризику перебувають малі команди, стартапи та користувачі без глибоких технічних знань, які користуються хмарними сервісами на кшталт Google AI Studio, Firebase або Google Cloud Console. У таких випадках API-ключ часто сприймається як допоміжний технічний параметр, хоча насправді він може створювати значні фінансові наслідки у разі компрометації.

Один із найпоширеніших сценаріїв пов'язаний із компрометацією API-ключа Google Maps, вбудованого у клієнтську частину вебсайту чи мобільного застосунку без належних обмежень. Якщо згодом у тому самому проєкті активується Gemini API, цей ключ може бути використаний для платних запитів до сервісів генеративного ШІ. У такому разі зловмисник після отримання доступу до ключа може виконувати запити за рахунок власника проєкту, що часто призводить до значних фінансових втрат, які можуть вимірюватися сотнями або навіть тисячами доларів за годину.

Дослідження Truffle Security у 2026 році показало, що старі Google API-ключі, які раніше сприймалися як відносно безпечні ідентифікатори для клієнтських сервісів, можуть створювати значно більші ризики після підключення Gemini. Дослідники повідомили про 2863 публічні Google API-ключі з доступом до Gemini [1]. Подібну проблему описала компанія CloudSEK: у 22 Android-застосунках було виявлено 32 жорстко вбудовані Google API-ключі, які потенційно відкривали доступ до сервісів Gemini. Сукупна аудиторія цих застосунків перевищувала 500 млн користувачів [2].

Окремою проблемою є неправильне розуміння бюджетних сповіщень. У Google Cloud бюджет за замовчуванням слугує механізмом сповіщення, а не автоматичним припиненням витрат. Документація Google Cloud прямо вказує, що встановлення бюджету не обмежує автоматично використання або витрати [3]. Для користувача без досвіду роботи з хмарною інфраструктурою слово "бюджет" може створити враження жорсткого фінансового ліміту, хоча насправді воно часто означає лише повідомлення про досягнення певної межі.

Для зменшення ризиків слід дотримуватись принципу "один ключ – одне призначення". Ключ, створений для Google Maps або Places API, не повинен використовуватися для інших сервісів. Для кожного середовища, сервісу та сценарію доцільно створювати окремий ключ із чітким найменуванням, що відображає його призначення. Також важливим є встановлення двох типів обмежень: обмеження за API та обмеження за джерелом запиту. Перше визначає, до яких саме API має доступ ключ, наприклад тільки до Places API або Maps JavaScript API. Друге означає, звідки дозволено використовувати ключ: з

конкретного домену вебсайту, IP-адреси сервера або мобільного застосунку. Google Maps Platform також рекомендує обмежувати API-ключі за застосунком і за доступними API [4].

Для запитів до Gemini API небажано використовувати пряме звернення з клієнтської частини вебзастосунку. Безпечнішою є архітектура, у якій ключ зберігається на сервері, у безсерверній функції або в захищеному сховищі секретів. Клієнтський застосунок має звертатися до власної серверної частини, а вона вже робить запит до Gemini API після перевірки користувача, обмеження частоти запитів та інших правил доступу. Якщо ключ потрапляє у зібраний JavaScript-код або мобільний застосунок без належних обмежень, його можуть знайти автоматизовані сканери.

Важливим доповненням до бюджетних сповіщень є квоти. Якщо API підтримує обмеження за кількістю запитів, їх потрібно встановлювати відповідно до реального сценарію використання, особливо для прототипів і тестових проєктів. Документація Google Cloud зазначає, що для окремих API можна явно обмежувати кількість запитів, щоб контролювати оплачуване використання [5]. Крім того, доцільно регулярно перевіряти список активованих API, історію використання, налаштування сповіщень, ролі доступу та застарілі тестові проєкти. Ключі, які більше не використовуються або могли бути відкриті у публічному коді, потрібно видаляти або замінювати.

Отже, розвиток генеративного ШІ суттєво прискорює створення цифрових продуктів, але також підвищує ризики, пов’язані з API-ключами. Основними заходами захисту є обмеження ключів за API і джерелом запиту, розділення проєктів за призначенням, серверне зберігання ключів для ШІ-сервісів, встановлення квот, регулярний аудит і видалення застарілих ключів. У контексті масового використання генеративного ШІ безпека API-ключів має розглядатися не як другорядне технічне налаштування, а як основа інформаційної та фінансової безпеки.

1. Truffle Security. Google API Keys Weren’t Secrets. But then Gemini Changed the Rules. URL: https://trufflesecurity.com/blog/google-api-keys-werent-secrets-but-then-gemini-changed-the-rules (дата звернення: 06.05.2026).
2. CloudSEK. Hardcoded Google API Keys in Top Android Apps Now Expose Gemini AI. URL: https://www.cloudsek.com/blog/hardcoded-google-api-keys-in-top-android-apps-now-expose-gemini-ai (дата звернення: 06.05.2026).
3. Google Cloud. Create, edit, or delete budgets and budget alerts. URL: https://docs.cloud.google.com/billing/docs/how-to/budgets (дата звернення: 07.05.2026).
4. Google Maps Platform. API security best practices. URL: https://developers.google.com/maps/api-security-best-practices (дата звернення: 07.05.2026).
5. Google Cloud. Capping API usage. URL: https://docs.cloud.google.com/apis/docs/capping-api-usage (дата звернення: 07.05.2026).