Аналіз вразливостей великих мовних моделей

Analysis of vulnerabilities in large language models

Віктор Кольченко [A]
Corresponding author: аспірант, асистент кафедри захисту інформації, e-mail: viktor.v.kolchenko@lpnu.ua, ORCID: 0009-0002-0718-6859

Дмитро Сабодашко [A]
доктор філософії, старший викладач кафедри захисту інформації, e-mail: dmytro.v.sabodashko@lpnu.ua, ORCID: 0000-0003-1675-0976

Марія Швед [A]
к.т.н, старший викладач кафедри захисту інформації, e-mail: mariia.y.shved@lpnu.ua, ORCID: 0000-0003-0428-7777

Юрій Хома [A]
д.т.н., доцент кафедри інформаційно-вимірювальних технологій, e-mail: yurii.v.khoma@lpnu.ua, ORCID: 0000-0002-4677-5392

Назар Максимів [A]
студент кафедри захисту інформації, e-mail: nazar.maksymiv.mkbas.2023@lpnu.ua, ORCID: 0009-0009-6496-4056

Viktor Kolchenko [A]
Corresponding author: Postgraduate student, Assistant Lecturer, e-mail: viktor.v.kolchenko@lpnu.ua, ORCID: 0009-0002-0718-6859

Dmytro Sabodashko [A]
Doctor of Philosophy, Senior Lecturer of Department, e-mail: dmytro.v.sabodashko@lpnu.ua, ORCID: 0000-0003-1675-0976

Mariia Shved [A]
Candidate of Technical Sciences, Senior Lecturer of Department, e-mail: mariia.y.shved@lpnu.ua, ORCID: 0000-0003-0428-7777

Yuriy Khoma [A]
Dr of Technical Sciences, Associate Professor of Department, e-mail: yurii.v.khoma@lpnu.ua, ORCID: 0000-0002-4677-5392

Nazar Maksymiv [A]
Student of the Department of Information Security, e-mail: nazar.maksymiv.mkbas.2023@lpnu.ua, ORCID: 0009-0009-6496-4056

[A] Національний університет "Львівська політехніка", м. Львів, Україна
[A] Lviv Polytechnic National University, Lviv, Ukraine

Received: October 18, 2024 | Revised: October 28, 2024 | Accepted: October 31, 2024

DOI: 10.33445/sds.2024.14.5.22

Мета роботи: аналіз вразливостей великих мовних моделей (LLM) на основі класифікації OWASP Top 10 для застосунків LLM, проведення оцінки потенційних загроз та розробка рекомендацій щодо підвищення рівня безпеки цих моделей.

Метод: кількісні методи, зокрема використання інструменту Garak для ідентифікації та аналізу вразливостей у великих мовних моделях.

Результати дослідження: проаналізовано декілька критичних вразливостей, серед яких ін'єкції запитів, небезпечна обробка виводів, отруєння навчальних даних, розкриття конфіденційної інформації та інші. Навіть найсучасніші моделі, зокрема GPT-4, не забезпечують повної захищеності від цих загроз.

Теоретична цінність дослідження: дослідження поглиблює розуміння безпекових ризиків, пов'язаних з використанням великих мовних моделей, і пропонує нові підходи до їх аналізу. Це може стати основою для подальших теоретичних розробок у галузі кібербезпеки стосовно LLM.

Практична цінність дослідження: дослідження пропонує рекомендації для розробників, науковців і організацій, які використовують LLM. Впровадження безпечних практик розробки та постійний аудит моделей можуть суттєво підвищити захист використання LLM.

Цінність дослідження: використано інноваційний підхід для тестування безпеки LLM через імітацію різноманітних атак, зокрема ін'єкції запитів, отруєння даних, галюцинації та витоки інформації, що дозволяє виявити потенційні слабкі місця у функціонуванні моделей.

Майбутні дослідження: у подальших дослідженнях планується розширення спектра моделей і методів, що спрямовані на оптимізацію безпеки. Це сприятиме формуванню більш комплексного розуміння проблеми та вдосконаленню стратегій забезпечення безпеки.

Тип статті: Концептуальне дослідження.

Ключові слова: великі мовні моделі, LLM, OWASP, кібербезпека, вразливості.

Purpose: analyzing the vulnerabilities of large-scale language models (LSLMs) based on the OWASP Top 10 classification for LSLM applications, assessing potential threats, and developing recommendations for improving the security of these models.

Method: quantitative methods, including the use of the Garak tool to identify and analyze vulnerabilities in large language models.

Findings: Several critical vulnerabilities have been identified, including query injection, insecure output processing, training data poisoning, disclosure of confidential information, and others. Even the most advanced models, including GPT-4, do not provide full protection against these threats.

Theoretical implications: The study deepens the understanding of security risks associated with the use of large language models and suggests new approaches to their analysis. This can serve as a basis for further theoretical developments in the field of cybersecurity with respect to LLMs.

Practical implications: The study offers recommendations for developers, researchers, and organizations that use LLMs. Implementing safe development practices and continuous model auditing can significantly increase the security of LLMs.

Value: an innovative approach was used to test the security of LLMs by simulating various attacks, including query injection, data poisoning, hallucinations, and information leaks, which allows identifying potential weaknesses in the functioning of the models.

Future research: In further research, we plan to expand the range of models and methods aimed at optimizing security. This will contribute to a more comprehensive understanding of the problem and improve security strategies.

Paper type: Conceptual research.

Key words: large language models, LLM, OWASP, cybersecurity, vulnerabilities.

Вступ

У сучасному світі штучний інтелект і машинне навчання набувають все більшого значення в різних сферах, зокрема в охороні здоров'я, освіті, бізнесі та наукових дослідженнях. Однією з найважливіших технологій у цьому контексті є великі мовні моделі (LLM від англ. large language model), які є інноваційною технологією, яка забезпечує машинне розуміння та генерацію тексту на рівні, близькому до людського, з безпрецедентною масштабністю [1]. Глибокі нейронні мережі, що складаються з мільярдів параметрів, на прикладі моделі GPT-3, використовуються для виявлення складних лінгвістичних шаблонів і контекстуальних залежностей [2].

Найбільш поширенішими LLM є GPT-3.5, GPT-4, які демонструють високі показники продуктивності в задачах генерації тексту, природньо-мовної обробки та адаптивного навчання, що робить їх ефективними інструментами для різноманітних застосувань в галузях штучного інтелекту та обчислювальної лінгвістики [3]. Проте, ефективність великих мовних моделей у застосунках генеративного штучного інтелекту робить їх привабливими цілями для зловмисників, що створюють унікальні виклики у виявленні, попередженні та мінімізації вразливостей [4].

Тому дослідження вразливостей великих мовних моделей (LLM), зокрема таких, як ін'єкція запитів, небезпечна обробка виводу, отруєння навчальних даних, відмова в обслуговуванні моделі, вразливості ланцюга поставок, розкриття конфіденційної інформації, небезпечне проектування плагінів, надмірні повноваження агентів, надмірна залежність і викрадення моделі, а також шляхи їхнього вирішення, є актуальним і потребує ретельного вивчення.

Теоретичні основи дослідження

Великі мовні моделі (LLM) представляють собою клас моделей глибокого навчання, які набули значної популярності завдяки своїм видатним можливостям в обробці природної мови. Ці моделі, засновані на архітектурах нейронних мереж, використовуються для аналізу та генерації людської мови, що дозволяє досягати високого рівня точності в таких завданнях, як автоматичний переклад, генерація текстів, розпізнавання емоцій та інших аспектів аналізу текстових даних. Навчання LLM відбувається на масивних наборах даних, які включають текстові дані різного типу, що підвищує здатність моделей адаптуватися до різноманітних мовних контекстів [5].

Серед найбільш поширених та перспективних великих мовних моделей (LLM) виокремлюються GPT-3.5, GPT-4 та Mistral 7B, які підлягатимуть детальному аналізу в наступних розділах цієї статті.

GPT-3.5 є міжгенераційною моделлю, розробленою OpenAI, яка стоїть між GPT-3 та GPT-4 [6]. Ця модель є вдосконаленою версією GPT-3 з додатковими оптимізаціями, що покращують її здатність до розуміння тексту та генерації відповідей. GPT-3.5 має велику кількість гіперпараметрів, хоча їх менше, ніж у GPT-4, що дозволяє їй ефективно обробляти мовні структури та генерувати тексти, які є релевантними та зв'язними. Модель була навчена на великому обсязі даних з різних джерел, що охоплюють широкий спектр тем і стилів. В ході експериментів використовувалася остання версія GPT-3.5 Turbo, яка була опублікована в січні 2024 року.

GPT-4 є четвертим поколінням трансформерної архітектури, розробленої OpenAI [7]. Це покращена версія попередніх моделей, яка має більшу кількість параметрів, що дозволяє їй генерувати більш точні та зв'язні тексти, краще розуміти контекст та виконувати різноманітні завдання, такі як переклад, резюмування текстів та відповіді на запитання. GPT-4 була навчена

на величезному обсязі даних, що охоплюють різні галузі знань, що робить її універсальною в застосуванні.

Mistral 7B є моделлю машинного навчання з відкритим кодом, яка також була розроблена для задач обробки природної мови. На відміну від моделей GPT від OpenAI, Mistral 7B менш відома, але вона також має значний потенціал для вирішення конкретних задач у галузі NLP (Natural Language Processing) [8]. Модель поширюється під ліцензією Apache 2.0, що робить її доступною для модифікації та використання дослідниками та розробниками для реалізації власних потреб. У дослідженні використовувалася версія Mistral 7Bv-0.1, опублікована у грудні 2023 року.

Однією з основних причин проведення досліджень у цій сфері є те, що великі мовні моделі, мають величезний потенціал для розвитку багатьох галузей, але одночасно вони є складними системами, що вимагають особливого підходу до забезпечення їхньої безпеки. Наприклад, вони можуть бути використані для автоматизації бізнес-процесів, створення систем штучного інтелекту, що відповідають на запитання користувачів, надають аналітичні висновки, допомагають у медичних діагнозах або виконують переклад текстів з однієї мови на іншу. У той же час, вразливість до атак, таких як ін’єкція запитів або витік даних, може спричинити серйозні наслідки, особливо в умовах застосування цих моделей у високочутливих системах, таких як банківська справа або охорона здоров’я.

Проте, із розвитком генеративних мовних моделей зростає небезпека використання їх для дезінформаційних кампаній, маніпуляції громадською думкою або автоматизованого створення шкідливого контенту. Вразливості в таких моделях можуть бути використані для підриву довіри до інформаційних систем, що ставить під загрозу не лише конфіденційність даних, але й надійність сучасної інформаційної екосистеми.

Постановка проблеми

Основною проблемою, розглянутою в цій статті, є недостатній рівень безпеки великих мовних моделей (LLM), які широко застосовуються у різних галузях, але мають низку критичних вразливостей, що можуть бути використані зловмисниками. Такі моделі, як GPT-3.5, GPT-4 та Mistral 7B, не гарантують повного захисту, що створює ризики, пов'язані з ін’єкціями запитів, витоками конфіденційної інформації та генерацією небажаного або шкідливого контенту.

Окрім цього, деякі мовні моделі мають функції, які не завжди чітко описані в технічній документації, що може бути використано для потенційних атак. Ця проблема набуває особливої важливості в умовах зростаючого використання LLM у критично важливих системах, де будь-які вразливості можуть призвести до серйозних наслідків.

Методологія дослідження

У даному дослідженні для аналізу вразливостей великих мовних моделей було використано сканер Garak [9]. Який використовує обширний діапазон тестів та запитів до великої мовної моделі, імітуючи атаки, та використовує ряд детекторів для аналізу результатів відповіді моделі, щоб побачити, чи була модель вразлива до будь-якої з цих атак. Garak перевіряє наявність галюцинацій, витоку даних, ін’єкції запитів, поширення неправдивої інформації, генерації токсичності та багатьох інших вразливостей. Після завершення тестування моделі, garak надає звіти з результатами у форматі HTML та JSONL, які в подальшому були проаналізовано.

Класифікацію вразливосте проводили на основі проєкту OWASP Top 10 for Large Language Model Applications (Top 10 вразливостей для застосунків, побудованих на основі великих мовних моделей) [10]. Даний проєкт ідентифікує десять найкритичніших вразливостей безпеки в застосунка.

Результати

В ході дослідження було проведено аналіз вразливостей та яким чином вони можуть реалізовуватися та розроблено рекомендації по захисту від них [4].

Аналіз вразливості LLM01: Prompt Injection (Ін'єкція запитів)

Ця вразливість виникає, коли зловмисник вводить спеціально підготовлені запити (промпти) або дані, маніпулюючи LLM для отримання непередбачуваних або шкідливих результатів [11]. Це використовує можливості моделі генерувати відповіді, що призводить до порушення цілісності у відповідях моделі.

Ін'єкції запитів можуть бути двох видів:

- Пряма ін'єкція запитів: Зловмисник перезаписує або розкриває базовий системний запит. Це може дозволити нападникам експлуатувати системи на серверній частині, взаємодіючи з ненадійними функціями та сховищами даних, до яких можна отримати доступ через LLM.
- Непряма ін'єкція запитів: LLM приймає вхідні дані з зовнішніх джерел, який контролюються нападником, наприклад, з вебсайтів або файлів. Нападник може вставити ін'єкцію запиту в зовнішній вміст, захоплюючи контекст розмови. Це дозволить нападнику маніпулювати користувачем або додатковими системами, до яких може отримати доступ LLM. Крім того, для непрямих ін'єкцій запитів не обов'язково бути видимими чи читабельними для людини, досить, щоб текст оброблявся LLM.

Приклади вразливості:

- Зловмисник створює пряму ін'єкцію запиту до LLM, яка наказує йому ігнорувати системні запити застосунку та виконати запит, який повертає приватну, небезпечну або іншу небажану інформацію.
- Користувач використовує LLM для підсумовування вебсторінки, яка містить непряму ін'єкцію запиту. Це приводить до того, що застосунок отримує чутливу інформацію у користувача та здійснює витік даних через JavaScript або Markdown.

Рекомендації по захисту:

- Забезпечте контроль привілеїв доступу LLM до бекенд-систем. Надайте LLM власні API токени для розширюваної функціональності, такої як плагіни, доступ до даних і дозволи на рівні функцій. Дотримуйтесь принципу найменших привілеїв, обмежуючи LLM лише мінімально необхідним рівнем доступу для його призначених операцій.
- Додайте людину в процес для розширеної функціональності. При виконанні привілейованих операцій, таких як відправлення або видалення електронних листів, зробіть так, щоб додаток вимагав попередньої згоди користувача на таку дію. Це зменшує можливість непрямої ін'єкції запитів, які можуть призвести до несанкціонованих дій від імені користувача без його відома або згоди.
- Відокремлюйте зовнішній контент від запитів користувача. Виділяйте та позначайте, де використовується ненадійний контент, щоб обмежити його вплив на запити користувачів.
- Встановіть довірчі межі між LLM, зовнішніми джерелами та плагінами. Ставтеся до LLM як до ненадійного користувача та зберігайте остаточний контроль користувача над процесами прийняття рішень. Скомпрометована LLM може діяти як посередник (man-in-the-middle) між API вашого застосунку та користувачем, оскільки вона може

приховувати або маніпулювати інформацією перед її представленням користувачеві. Виділяйте потенційно ненадійні відповіді візуально для користувача.

- Періодично вручну проводьте моніторинг вхідних та вихідних даних LLM, щоб перевірити, чи вони відповідають очікуванням. Хоча це не є засобом пом'якшення, це може надати необхідні дані для виявлення слабкостей і їх усунення.

Аналіз вразливості LLM02: Insecure Output Handling (Небезпечна обробка виводу)

Проблема ризикованої обробки результатів стосується неадекватної перевірки, очищення та керування виводами, що створюються великими мовними моделями, перед їх передачею іншим елементам та системам. Це пов’язано з тим, що вміст, сформований за допомогою LLM, можуть регулювати зовнішні запити, що подібне до надання користувачам доступу до додаткового функціоналу. Відмінність ризикованої обробки виводів від надмірно залежності (вразливість LLM09) полягає в тому, що вона стосується результатів, генерованих LLM, перш ніж вони передаються далі, тоді як надмірна залежність зосереджується на ширших питаннях, пов’язаних із занадто великою залежністю від точності та доречності виводів LLM. Успішне використання вразливості небезпечної обробки виводу може призвести до XSS (міжсайтовий скриптинг) та CSRF (міжсайтова підробка запиту) у веб-браузерах, а також SSRF (підробка запитів на стороні сервера), підвищення привілеїв або виконання віддаленого коду на бекенд-системах [12].

Наступні умови можуть збільшити вплив цієї вразливості:

- Застосунок надає LLM привілеї, що виходять за межі того, що призначено для кінцевих користувачів, дозволяючи підвищення привілеїв або виконання віддаленого коду.
- Застосунок вразливий до атак через непряму ін’єкцію запитів, що може дозволити нападнику отримати привілейований доступ до середовища цільового користувача.
- Плагіни третіх сторін не забезпечують належної валідації вхідних даних.

Приклади вразливостей:

- Вивід LLM безпосередньо вводиться в системний командний рядок або функцію, таку як exec або eval, що призводить до виконання віддаленого коду.
- JavaScript або Markdown, згенеровані LLM, повертаються користувачеві. Потім код інтерпретується браузером, що призводить до XSS.

Рекомендації по захисту:

- Ставтеся до моделі як до будь-якого іншого користувача, застосовуючи підхід нульової довіри та застосовуйте належну валідацію вхідних даних відповідей, що надходять від моделі до функцій бекенду.
- Дотримуйтесь рекомендацій OWASP ASVS (Стандарт верифікації безпеки застосунків) [13] для забезпечення ефективної валідації та санітації вводу або інших рекомендацій з пост-процесінгу відповідей моделі [14].

Аналіз вразливості LLM03: Training Data Poisoning (Отруєння навчальних даних)

Отруєння навчальних даних відбувається, коли дані, що використовуються для тренування LLM, змінюються зі зловмисною метою, що призводить до упередженої, некоректної або шкідливої поведінки моделі [15]. Це може кардинально підірвати надійність та точність моделі, а також нанести шкоду репутації компанії.

Приклади вразливостей:

- Модель навчається використовуючи дані, джерело яких не було перевірене за походженням або змістом на будь-якому з етапів навчання, що може призвести до помилкових результатів, якщо дані є забрудненими (отруєними) або некоректними.
- Відсутність контролю доступу до інфраструктури або недостатня ізоляція може дозволити моделі отримувати небезпечні навчальні дані, що призведе до упереджених або шкідливих результатів.

Рекомендації по захисту:

- Перевіряйте ланцюг постачання даних для навчання, особливо коли вони отримані з зовнішніх джерел.
- Перевіряйте легітимність цільових джерел даних та отриманих даних під час навчання та тонкого налаштування.
- Перевіряйте ваші варіанти використання LLM та застосунок, з яким модель буде інтегровано. Створіть різні моделі за допомогою окремих навчальних даних або тонкого налаштування для різних варіантів використання, щоб отримати більш детальний і точний вихід генеративного ШІ відповідно до його визначеного варіанту використання.
- Забезпечити достатню ізоляцію за допомогою налаштувань прав доступу, щоб запобігти моделі зчитувати ненавмисні джерела даних, які можуть перешкодити результатам машинного навчання.
- Використовуйте суворий відбір або фільтри введення для певних навчальних даних або категорій джерел даних, щоб контролювати обсяг сфальсифікованих даних. Очищення даних, за допомогою таких методів, як виявлення статистичних аномалій, дозволяє виявити та видалити отруєні дані, які можуть бути використані на етапах тренування або тонкого налаштування.

Аналіз вразливості LLM04: Model Denial of Service (Відмова у обслуговуванні моделі)

Це означає атаки, спрямовані на перевантаження LLM, або шляхом перевантаження обчислювальних ресурсів інфраструктури, або шляхом спричинення поведінки моделі, що веде до її збою або значного сповільнення, тим самим перешкоджаючи її доступності та надійності.

Приклади вразливостей:

- Створення ресурсного голоду. Зловмисник надсилає запити, які призводять до збільшеного використання ресурсів через масове створення завдань для моделі.
- Зловмисник надсилає неочікувані запити, що споживають багато ресурсів, використовуючи нестандартну орфографію або послідовності.
- Зловмисник надсилає LLM потік вхідних даних, який перевищує вікно контексту, змушуючи модель споживати надмірні обчислювальні ресурси.
- Зловмисник створює вхідні дані, які запускають рекурсивне розширення контексту, змушуючи LLM повторно розширювати та обробляти вікно контексту.

Рекомендації по захисту:

- Валідація та очищення вхідних даних. Цей метод гарантує, що вхідні дані від користувачів відповідають встановленим обмеженням, а також фільтрує будь-який шкідливий вміст.
- Обмеження ресурсів, що виділяються на кожен запит або крок, сповільнюючи виконання запитів, які залучають складні елементи.

- Обмеження кількість запитів до API, які може зробити окремий користувач (авторизований, або ідентифікований за IP-адресою) за певний проміжок часу.
- Обмеження черги дій: Цей метод обмежує кількість дій, що очікують на виконання, та загальну кількість дій у системі, які реагують на відповіді LLM.
- Постійний моніторинг використання ресурсів LLM для виявлення аномальних сплесків, які можуть вказувати на атаку типу “відмова в обслуговуванні” (DoS).
- Обмеження вхідних даних: Цей метод встановлює строгі обмеження на розмір вхідних даних на основі вікна контексту LLM, запобігаючи перевантаженню та виснаженню ресурсів.
- Підвищення обізнаності серед розробників щодо потенційних DoS-вразливостей у моделях та затосунках.

Аналіз вразливості LLM05: Supply Chain Vulnerabilities (Вразливості ланцюга поставок)

В ланцюгу постачання великих мовних моделей існують вразливості, що негативно впливають на цілісність навчальних даних, моделей машинного навчання та платформ для розгортання. Такі слабкі місця можуть спричинити виникнення упередженості в результатах, порушення у сфері безпеки або навіть катастрофічні збої у системі. Раніше акцент зазвичай робився на вразливостях програмних компонентів, однак у сфері машинного навчання ця проблематика розширюється через використання попередньо тренованих моделей і навчальних даних від сторонніх постачальників, які можуть бути піддані атакам, спрямованим на їхню зміну чи “отруєння” [16].

Приклади вразливостей:

- Традиційні вразливості сторонніх програмних пакетів, бібліотек, включаючи застарілі або непідтримувані компоненти.
- Використання вразливої попередньо навченої моделі для тонкого налаштування може призвести до вразливості всієї моделі.
- Використання забруднених даних з відкритих джерел для навчання моделі може призвести до того, що модель буде генерувати шкідливий або неправдивий вихід.
- Використання застарілих або непідтримуваних моделей, які більше не оновлюються, може призвести до проблем із безпекою.
- Нечіткі умови використання та політика конфіденційності даних операторів моделі можуть призвести до того, що конфіденційні дані програми будуть використані для навчання моделі, а потім оприлюднені. Це також може стосуватися ризиків використання моделі, яка містить матеріали, захищені авторським правом.

Рекомендації по захисту:

- Ретельно перевіряйте джерела даних і постачальників, включаючи їхні умови використання та політику конфіденційності. Співпрацюйте лише з надійними постачальниками. Переконайтеся, що в них запроваджено належні та незалежно перевірені заходи безпеки, а політика оператора моделі відповідає вашій політиці захисту даних (тобто ваші дані не використовуються для навчання їхніх моделей). Також отримайте гарантії та правові заходи щодо неналежного використання захищеного авторським правом матеріалу від обслуговуючих організацій моделі.
- Ознайомтеся та застосуйте методи запобігання вразливостей, описані в розділі A06:2021 – “Вразливі та застарілі компоненти” стандарту OWASP Top 10 [13]. До цього належить сканування вразливостей, управління ними та оновлення компонентів.

Також застосовуйте ці методи контролю в середовищах розробки, які мають доступ до конфіденційних даних.

- Використовуйте цифровий підпис для моделей і коду під час роботи із зовнішніми моделями та постачальниками.
- Виявлення аномалій у даних та тестування на стійкість до спроб ввести в оману моделей можуть допомогти виявити фальшування та отруєння.
- Реалізуйте належний моніторинг для виявлення вразливостей компонентів та середовища, використання неавторизованих плагінів та застарілих компонентів, включаючи модель та її артефакти.
- Впровадьте політику патчів для усунення вразливих або застарілих компонентів. Переконайтеся, що програма використовує підтримувані версії API та базової моделі.
- Регулярно переглядайте та аудітуйте безпеку та доступ постачальників, щоб переконатися, що їхній рівень безпеки та умови використання не змінилися.

Аналіз вразливості LLM06: Sensitive Information Disclosure (Розкриття конфіденційної інформації)

Застосунки на основі LLM можуть ненавмисно розкривати конфіденційну інформацію, власні алгоритми або інші конфіденційні дані через свої відповіді. Це може призвести до несанкціонованого доступу до чутливих даних, порушення інтелектуальної власності, порушення конфіденційності та інших проблем безпеки.

Приклади вразливостей:

- Оператор непрямо вводить чутливі або конфіденційні дані в процеси навчання моделі, які потім повертаються у результатах генерації LLM.
- Неповне або неправильне фільтрування конфіденційної інформації в відповідях LLM.
- Надмірне навчання або запам’ятовування конфіденційних даних під час навчання моделі.
- Ненавмисне розкриття конфіденційної інформації через неправильне тлумачення моделлю, відсутність методів очищення даних або як результат помилки.

Рекомендації по захисту:

- Застосуйте належні методи очищення даних, щоб запобігти потраплянню конфіденційних даних до навчальних даних моделі.
- Впровадьте надійні методи перевірки та очищення вхідних даних для виявлення та фільтрації потенційно шкідливих елементів, щоб запобігти отруєнню моделі.
- Будь-яка інформація, що вважається конфіденційною в даних для тренування або тонкого налаштування, може бути потенційно розкрита користувачеві. Тому застосовуйте принцип мінімальних привілеїв і не навчайте модель на інформації, до якої має доступ користувач із найвищими привілеями, якщо вона може бути відображена користувачеві з нижчими привілеями.
- Слід обмежити доступ до зовнішніх джерел даних
- Застосуйте суворі методи контролю доступу до зовнішніх джерел даних та ретельний підхід до підтримки безпечного ланцюга поставок.

Аналіз вразливості LLM07: Insecure Plugin Design (Небезпечне проектування плагінів)

У контексті LLM, небезпечне проектування плагінів стосується інтеграції додаткових функцій або розширень, які належним чином не захищені, потенційно вносячи вразливості в безпечну базову модель.

Приклади вразливостей:

- Плагін приймає всі параметри в одному текстовому полі замість окремих параметрів вводу.
- Плагін приймає рядки конфігурації замість параметрів, які можуть перезаписати всі налаштування конфігурації.
- Плагін приймає невідформатовані SQL-запити або частини коду замість параметрів.
- Автентифікація виконується без явної авторизації для конкретного плагіна.
- Плагін розглядає весь вміст LLM як повністю створений користувачем і виконує будь-які запити без додаткової авторизації.

Рекомендації по захисту:

- Плагіни повинні, за можливості, використовувати суворий параметризований ввод та включати перевірку типу та діапазону вхідних даних. Якщо це неможливо, слід запровадити додатковий рівень, що здійснює аналіз запитів та застосовує валідацію та очищення. У випадках, коли за логікою програми необхідно приймати неформатовані дані, їх слід ретельно перевіряти, щоб гарантовано запобігти виклику потенційно шкідливих методів.
- Розробники плагінів повинні застосовувати рекомендації OWASP ASVS (Application Security Verification Standard) [13] для забезпечення належної валідації та очищення вхідних даних.
- Плагіни повинні пройти ретельне тестування та перевірку для забезпечення належної валідації.
- Плагіни повинні використовувати відповідні протоколи авторизації, такі як OAuth2, для застосування ефективного контролю доступу.
- Вимагайте ручного дозволу користувача та підтвердження будь-якої дії, виконаної чутливими плагінами.
- Оскільки плагіни зазвичай це REST API, розробники повинні застосовувати рекомендації, наведені в OWASP Top 10 API Security Risks – 2023 [17], щоб мінімізувати загальні вразливості.

Аналіз вразливості LLM08: Excessive Agency (Надмірні повноваження агенту)

В розробці програмного забезпечення “агент” зазвичай означає програму або програмний компонент, який діє автономно від імені користувача, іншої програми або системи. Агенти призначені для виконання певних завдань або прийняття рішень на основі заздалегідь визначених правил, алгоритмів або моделей машинного навчання. Вони можуть працювати незалежно або співпрацювати з іншими агентами для досягнення спільних цілей. Надмірні повноваження агенту в застосунках заснованих на LLM, є вразливістю, спричиненою надмірними дозволами, надмірною функціональністю або занадто великою автономією. Ця вразливість може призвести до широкого спектру наслідків, що впливають на конфіденційність, цілісність та доступність даних, і залежить від того, з якими системами може взаємодіяти застосунок.

Приклади вразливостей:

- Надмірна функціональність: агент має доступ до плагінів, які містять функції, що не потрібні для належної роботи системи. Наприклад, розробник повинен надати сервісу LLM можливість читати документи зі сховища, але сторонній плагін, який він обирає використовувати, також дозволяє редагувати та видаляти документи.
- Надмірні дозволи: Плагін має дозволи на інших системах, які не потрібні для належної роботи програми. Наприклад, плагін, призначений для читання даних, підключається

до сервера бази даних за допомогою облікового запису, який має не тільки дозволи SELECT, але й дозволи UPDATE, INSERT та DELETE.

- Надмірні дозволи: Плагін, призначений для виконання операцій від імені користувача, отримує доступ до нижчестоящих систем за допомогою загальної облікового запису з високими правами. Наприклад, плагін для читання сховища документів поточного користувача підключається до сховища документів за допомогою облікового запису з високими правами, який має доступ до файлів усіх користувачів.
- Надмірна автономія: Програма або плагін на основі LLM не може самостійно перевіряти та підтверджувати дії з високим впливом результату. Наприклад, плагін, який дозволяє видаляти документи користувача, виконує видалення без будь-якого підтвердження з боку користувача.

Рекомендації по захисту:

- Обмежте плагіни або інструменти, які дозволено викликати агентам, лише мінімально необхідними функціями. Наприклад, якщо система не потребує можливості отримувати вміст URL-адреси, то такий плагін не слід пропонувати агенту LLM.
- Обмежте функціонал, реалізовані в плагінах/інструментах, мінімально необхідним. Наприклад, плагіну, який отримує доступ до поштової скриньки користувача для аналізу електронних листів, може знадобитися лише можливість читати листи, тому плагін не повинен містити інших функцій, таких як видалення або надсилання повідомлень.
- За можливості слід уникати функцій з гнучкою поведінкою (наприклад, виконання команди системної оболонки, отримання URL-адреси тощо) та використовувати плагіни/інструменти з більш детальною функціональністю. Наприклад, програмі може знадобитися записати вихідні дані у файл. Якщо б це було реалізовано за допомогою плагіна для виконання функції системної оболонки, тоді сфера застосування небажаних дій була б дуже широкою (можна виконати будь-яку іншу команду оболонки). Більш безпечною альтернативою було б створити плагін для запису файлів, який міг би підтримувати лише цю конкретну функцію.
- Відстежуйте авторизацію користувача та область безпеки, щоб гарантувати, що дії, які виконуються від імені користувача, здійснюються на нижчестоящих системах у контексті цього конкретного користувача та з мінімально необхідними привілеями. Наприклад, плагін, який читає репозиторій коду користувача, має вимагати від користувача автентифікації через OAuth із мінімально необхідною областю безпеки.

Аналіз вразливості LLM09: Overreliance (Надмірна залежність)

Надмірна залежність може виникнути, коли LLM генерує помилкову інформацію і подає її як достовірну. Хоча LLM можуть створювати креативний та інформативний контент, вони також можуть генерувати контент, який є фактично неточним, недоречним або небезпечним. Це називається галюцинацією або конфабуляцією [18]. Коли люди або системи довіряють цій інформації без контролю чи підтвердження, це може призвести до витоку безпеки, дезінформації, непорозумінь, юридичних проблем та шкоди репутації.

Приклади вразливостей:

- Неточність під маскою авторитетності: LLM може надавати невірну інформацію, при цьому уявляючи її як абсолютно достовірну та правдиву. Системи, розроблені без належних перевірок та контролю якості, можуть сприймати цю інформацію як істину, що вводить користувачів в оману та потенційно призводить до шкоди.

- Небезпечні підказки: LLM може пропонувати небезпечний або помилковий код, який, якщо його інтегрувати в програмне забезпечення без належної перевірки та підтвердження, може створити вразливості в системі.

Рекомендації по захисту:

- Регулярно контролюйте та переглядайте вихідні дані LLM. Узгоджуйте і відфільтровуйте несумісний текст. Порівняння відповідей декількох моделей для одного запиту може допомогти краще оцінити якість та узгодженість вихідних даних.
- Перевіряйте вихідні дані LLM з надійними зовнішніми джерелами. Цей додатковий рівень перевірки допоможе гарантувати, що інформація, надана моделлю, є точною та достовірною.
- Підвищення якості виведення за допомогою тонкого налаштування. Загальні попередньо навчені моделі частіше генерують неточну інформацію порівняно з тонко налаштованими моделями в конкретній області. Для цього можна застосувати такі методи, як проектування запитів (prompt engineering) [19], повне налаштування моделі та підказки щодо ланцюга міркувань.
- Впровадження механізмів автоматичної перевірки для порівняння згенерованого виводу з відомими фактами або даними. Це може забезпечити додатковий рівень безпеки та зменшити ризики, пов’язані з галюцинаціями або конфабуляцією.
- Створіть API та інтерфейси користувача, які сприяють відповідальному та безпечному використанню LLM. Це може включати такі заходи, як:
    - Фільтри вмісту: Запобігання створенню або поширенню шкідливого чи образливого контенту.
    - Попередження користувачів про можливі неточності: Інформування користувачів про те, що LLM може генерувати неточну інформацію, та заохочення до критичної оцінки вихідних даних.
    - Чітке маркування контенту, створеного штучним інтелектом: Інформування користувачів про те, що певний контент був створений LLM, а не людиною.

Аналіз вразливості LLM10: Model Theft (Викрадення моделі)

Викрадення моделі включає несанкціоноване копіювання або екстракцію LLM, або їхні ваги та параметри витягуються для створення функціонального еквівалента. Така крадіжка може бути використана для конкурентної переваги, зловмисних цілей або для обходу ліцензійних обмежень, а також отримання несанкціонованого доступу до чутливої інформації, що міститься в навчальних даних [20].

Приклади вразливостей:

- Зловмисник використовує вразливість в інфраструктурі компанії, щоб отримати несанкціонований доступ до сховища LLM через помилки конфігурації мережевих або програмних налаштувань безпеки.
- Сценарій внутрішньої загрози, коли незадоволений співробітник викрадає модель або пов’язані з нею артефакти.
- Зловмисник надсилає запити до API моделі, використовуючи ретельно підготовлені вхідні дані та методи ін’єкції запитів (LLM01), щоб зібрати достатню кількість вихідних даних для створення “тіньової” моделі. Тіньова модель – це спроба відтворити оригінальну модель шляхом аналізу її вихідних даних.
- Зловмисник може обійти методи фільтрації вхідних даних LLM, щоб здійснити атаку з побічним каналом і зрештою викрасти ваги та інформацію про архітектуру моделі. Атака з побічним каналом – це техніка, яка дозволяє витягувати конфіденційну

інформацію з комп'ютерної системи, аналізуючи її неявні витоки, такі як час виконання, споживання пам'яті або інші показники.

Рекомендації по захисту:

- Впровадьте надійний контроль доступу (наприклад, RBAC та принцип мінімальних привілеїв) та потужні механізми аутентифікації, щоб обмежити несанкціонований доступ до сховищ моделей LLM та середовищ навчання.
- Обмежте доступ LLM до мережевих ресурсів, внутрішніх служб та API.
- Регулярно здійснюйте моніторинг та перевіряйте журнали доступу та дії, пов'язані зі сховищами моделей, щоб швидко виявляти та реагувати на будь-яку підозрілу або несанкціоновану поведінку.
- Автоматизуйте розгортання з керуванням робочими процесами відстеження та затвердження для посилення контролю доступу та розгортання в межах інфраструктури.
- Впровадьте засоби керування та стратегії усунення ризиків, щоб зменшити ризик використання методів ін'єкції підказок, що призводять до атак з побічним каналом.
- Застосовуйте обмеження швидкості викликів API, де це можливо, або фільтри для зменшення ризику викрадення даних із програм LLM.
- Посилення заходів фізичної безпеки.

Впровадьте фреймворк водяних знаків на етапах вбудовування та тонкого налаштування життєвого циклу LLM.

Висновки

У даній роботі було розглянуто питання вразливостей великих мовних моделей (LLM) у контексті їхнього застосування в сучасних інформаційних системах. Основна увага була зосереджена на аналізі критичних вразливостей, серед яких ін'єкція запитів, небезпечна обробка виводів, отруєння навчальних даних та розкриття конфіденційної інформації. В ході дослідження особливу увагу було приділено вивченню методів захисту, заснованих на рекомендаціях OWASP Top 10 для застосувань LLM, які є найбільш перспективними та теоретично придатними для впровадження у сучасні моделі.

Під час дослідження із застосуванням інструменту Garak було знайдено можливість ідентифікації низки базових вразливостей у великих мовних моделях, таких як GPT-3.5, GPT-4 та Mistral 7B. Основним досягненням цього дослідження стало те, що за допомогою інструменту вдалося ідентифікувати потенційні загрози, такі як ін'єкція запитів та небезпечна обробка виводів, що можуть призвести до витоку інформації або генерації небажаного контенту.

Отримані результати відкривають широкі можливості для подальшого дослідження та впровадження нових методів захисту мовних моделей. Використання методів захисту, таких як фільтрація виводів та обмеження доступу до навчальних даних, допоможе суттєво підвищити безпеку великих мовних моделей, зокрема для застосувань у галузях з високими вимогами до конфіденційності та надійності даних. Таким чином, дане дослідження показало важливість впровадження комплексних заходів для підвищення захищеності великих мовних моделей, що дозволить забезпечити їх безпечне використання в реальних умовах, мінімізуючи ризики несанкціонованого доступу та викрадення інформації.

Фінансування

Це дослідження не отримало конкретної фінансової підтримки.

Конкуруючі інтереси

Автори заявляють, що у них немає конкуруючих інтересів.

Список використаних джерел

1. Sreerakuvandana, S., Pappachan, P., Arya, V. (2024). Understanding large language models. In Advances in computational intelligence and robotics book series (pp. 1–24). https://doi.org/10.4018/979-8-3693-3860-5.ch001
2. Chen, Z., Xu, L., Zheng, H., Chen, L., Tolba, A. et al. (2024). Evolution and prospects of foundation models: from large language models to large multimodal models. Computers, Materials & Continua, 80(2), 1753-1808. https://doi.org/10.32604/cmc.2024.052618.
3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2005.14165
4. Brown, H., Lee, K., Mireshghallah, F., Shokri, R., Tramèr, F. (2022). What Does it Mean for a Language Model to Preserve Privacy? 2022 ACM Conference on Fairness, Accountability, and Transparency. https://doi.org/10.1145/3531146.3534642
5. Gholami, N. Y. (2024b). Large Language Models (LLMs) for Cybersecurity: A Systematic review. World Journal of Advanced Engineering Technology and Sciences, 13(1), 057–069. https://doi.org/10.30574/wjaets.2024.13.1.0395
6. Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., …Cui, Y. (2023). A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.10420
7. OpenAI. (2023). GPT-4 Technical Report. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.08774
8. Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., De Las Casas, D., …Bressand, F. (2023). Mistral 7B. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.06825
9. Derczynski, L., Galinkin, E., Majumdar, S. (2024). Garak: A framework for large language model red teaming. [Електронний ресурс] // Garak Ai – URL : https://garak.ai
10. OWASP. (2023). OWASP top 10 for LLM applications, version 1.1.[Електронний ресурс] // OWASP – URL : https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-2023-v1_1.pdf
11. Capitella, D. (n.d.). Forging thoughts and observations in REACT-based LLM agents via prompt injection. [Електронний ресурс] // Wi Labs – URL : https://labs.withsecure.com/publications/llm-agent-prompt-injection
12. Петрів, П., Опірський, І. (2023). АНАЛІЗ ПРОБЛЕМАТИКИ ВИКОРИСТАННЯ ІСНУЮЧИХ СТАНДАРТІВ З ВЕБ-ВРАЗЛИВОСТЕЙ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(22), 96–112. https://doi.org/10.28925/2663-4023.2023.22.96112
13. OWASP Application Security Verification Standard (ASVS) [Електронний ресурс] // OWASP. URL : https://owasp.org/www-project-application-security-verification-standard/
14. Goldstein, J. A., Sastry, G., Musser, M., DiResta, R., Gentzel, M., Sedova, K. (2023). Generative language models and automated influence operations: emerging threats and potential mitigations. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2301.04246
15. Aghakhani, H., Dai, W., Manoel, A., Fernandes, X., Kharkar, A., Kruegel, C., …Vigna, G. (2023). TrojanPuzzle: Covertly Poisoning Code-Suggestion Models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2301.02344

16. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., Zhang, Y. (2023). A survey on Large Language Model (LLM) Security and Privacy: The Good, the bad, and the ugly. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2312.02003
17. Team, O. W. A. S. P. (n.d.). OWASP Top 10 API Security Risks – 2023 - OWASP API Security Top 10. [Електронний ресурс] // OWASP URL : https://owasp.org/API-Security/editions/2023/en/0x11-t10/
18. Zhang, M., Press, O., Merrill, W., Liu, A., Smith, N. (2023). How language model hallucinations can snowball. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2305.13534
19. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, …Schmidt, D. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2302.11382
20. Shayegani, E., Mamun, A., Fu, Y., Zaree, P., Dong, Y., Abu-Ghazaleh, N. (2023). Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.10844

References

1 Sreerakuvandana, S., Pappachan, P., Arya, V. (2024). Understanding large language models. In Advances in computational intelligence and robotics book series (pp. 1–24). https://doi.org/10.4018/979-8-3693-3860-5.ch001
2. Chen, Z., Xu, L., Zheng, H., Chen, L., Tolba, A. et al. (2024). Evolution and prospects of foundation models: from large language models to large multimodal models. Computers, Materials & Continua, 80(2), 1753-1808. https://doi.org/10.32604/cmc.2024.052618.
3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2005.14165.
4. Brown, H., Lee, K., Mireshghallah, F., Shokri, R., Tramèr, F. (2022). What Does it Mean for a Language Model to Preserve Privacy? 2022 ACM Conference on Fairness, Accountability, and Transparency. https://doi.org/10.1145/3531146.3534642
5. Gholami, N. Y. (2024b). Large Language Models (LLMs) for Cybersecurity: A Systematic review. World Journal of Advanced Engineering Technology and Sciences, 13(1), 057–069. https://doi.org/10.30574/wjaets.2024.13.1.0395
6. Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., …Cui, Y. (2023). A comprehensive capability analysis of GPT-3 and GPT-3.5 series models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.10420
7. OpenAI. (2023). GPT-4 Technical Report. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.08774
8. Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., De Las Casas, D., …Bressand, F. (2023). Mistral 7B. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.06825
9. Derczynski, L., Galinkin, E., Majumdar, S. (2024). Garak: A framework for large language model red teaming. // Garak Ai – Available from : https://garak.ai
10. OWASP. (2023). OWASP top 10 for LLM applications, version 1.1. // OWASP Available from : https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-2023-v1_1.pdf
11. Capitella, D. (n.d.). Forging thoughts and observations in REACT-based LLM agents via prompt injection. // Wi Labs – Available from : https://labs.withsecure.com/publications/llm-agent-prompt-injection

12. Petriv, P., Opirskyi, I. (2023). Analysis of the problems of using existing web vulnerability standards. *Cybersecurity: education, science, technology*, 2(22), 96–112. https://doi.org/10.28925/2663-4023.2023.22.96112
13. OWASP Application Security Verification Standard (ASVS) [Електронний ресурс] // OWASP - Available from : https://owasp.org/www-project-application-security-verification-standard/
14. Goldstein, J. A., Sastry, G., Musser, M., DiResta, R., Gentzel, M., Sedova, K. (2023). Generative language models and automated influence operations: emerging threats and potential mitigations. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2301.04246
15. Aghakhani, H., Dai, W., Manoel, A., Fernandes, X., Kharkar, A., Kruegel, C., …Vigna, G. (2023). TrojanPuzzle: Covertly Poisoning Code-Suggestion Models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2301.02344
16. Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., Zhang, Y. (2023). A survey on Large Language Model (LLM) Security and Privacy: The Good, the bad, and the ugly. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2312.02003
17. Team, O. W. A. S. P. (n.d.). OWASP Top 10 API Security Risks – 2023 - OWASP API Security Top 10. [Електронний ресурс] // OWASP – Available from : https://owasp.org/API-Security/editions/2023/en/0x11-t10/
18. Zhang, M., Press, O., Merrill, W., Liu, A., Smith, N. (2023). How language model hallucinations can snowball. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2305.13534
19. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, …Schmidt, D. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2302.11382
20. Shayegani, E., Mamun, A., Fu, Y., Zaree, P., Dong, Y., Abu-Ghazaleh, N. (2023). Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2310.10844