УДК 004.056.5

ВИЯВЛЕННЯ НЕДЕКЛАРОВАНИХ МОЖЛИВОСТЕЙ АНДРОЇД-ЗАСТОСУНКІВ МЕТОДОМ LLM-КЕРОВАНОЇ ПЕРЕВІРКИ ГІПОТЕЗ: ФОРМАЛЬНА МОДЕЛЬ

Д. О. Кригін[1,a], М. І. Ільїн[1]

[1] Навчально-науковий Фізико-технічний інститут

Анотація

У процесі пошуку недекларованих можливостей у програмному забезпеченні постановка та перевірка гіпотез — систематичний збір доказів на користь або проти потенційно наявного недекларованого функціоналу — є найбільш часозатратним етапом аналітичного процесу. У цій статті пропонується гібридна модель аналізу, в якій велика мовна модель (LLM) виступає як автоматизований тестувальник гіпотез: аналітик формулює гіпотези вразливостей або їх ланцюжків природною мовою, а LLM-агент перетворює кожний із них на стратегію збору доказів, проводить аналіз програмного застосунку для збагачення контексту та, на основі гіпотези і розширеного контексту, спростовує гіпотезу, створює Proof of Concept, що підтверджує її, або мутує для подальшого дослідження. Представлений підхід формалізовано за допомогою LTS (Labelled Transition System), поведінкових предикатів та скінченного автомата. Представлена модель дозволяє одному аналітику паралельно підтримувати та перевіряти десятки гіпотез, щодо наявного недекларованого функціоналу — включно зі складними міждоменними ланцюжками вразливостей та місконфігурацій, що охоплюють не лише Андроїд-застосунок і простір користувача, а архітектуру мобільного програмного забезпечення, включно з IoT та веб-технологіями, що не стає обмежувальним фактором у дослідженні. Дана гібридна модель дозволяє ефективно виконувати задачі, для яких раніше потребувалися команди з багатьох інженерів.

Ключові слова: безпека Android, аналіз на основі гіпотез, LLM, динамічне інструментування, ланцюжок вразливостей, поведінковий предикат, Frida.

Вступ

Сучасні Android-застосунки функціонують у комплексних технологічних стеках: вони взаємодіють із вбудованим IoT-обладнанням через пропрієтарні бездротові протоколи, синхронізуються із вбудованими прошивками, нативно відображають веб-контент та взаємодіють з бекендом у хмарі через власні канали оновлень [1]. Виявлення недокументованої або шкідливої поведінки в таких екосистемах вимагає одночасної компетентності в proximity протоколах, внутрішніх механізмах Android, реверс-інжинірингу, веббезпеці та мережевому аналізі — поєднанні, яке жоден окремий аналітик не зможе підтримувати на належному рівні.

У процесі пошуку недекларованих можливостей *перевірка гіпотез* є основним фактором часових витрат. Після того як аналітик підозрює наявність вразливості — наприклад, що у застосунку є витік персональних даних користувача — він повинен спланувати експеримент, налаштувати багаторівневе інструментування, зібрати й інтерпретувати трасування, виключити альтернативні пояснення та отримати відтворюваний підтверджуючий доказ. Цей цикл може тривати дні або тижні для однієї гіпотези; складний міждоменний ланцюжок вразливостей може потребувати десятків спростованих або незалежно підтверджених гіпотез. У практиці, при тестуванні на проникнення, перевірка гіпотез займає переважну частку робочого часу — і ця частка зростає разом із складністю екосистем.

Великі мовні моделі [2] з можливостями використання інструментів [3] докорінно змінюють цю проблематику. Запропонована модель дозволяє аналітику описати ланцюжок вразливостей природною мовою, тоді як LLM проводить перевірку гіпотези. Це звільняє аналітика для зосередження на творчому семантичному завданні побудови ланцюжків гіпотез, дозволяючи одному аналітику паралельно опрацьовувати десятки ланцюжків без виснаження та у прийнятний час.

Наукова новизна. Вперше запропоновано формальну модель, в якій LLM виступає як *тестувальник гіпотез* на всіх фазах робочого процесу безпекового аналізу Android-застосунків, а не як пасивний консультант. Модель інтегрує статичне отримання контексту, формулювання ланцюжків гіпотез аналітиком, автоматизоване багатоджерельне інстру-

[a]krydmy-ipt24@lll.kpi.ua

ментування та генерацію підтверджуючих доказів у єдиний формально специфікований конвеєр.

1. Формальні основи

Нехай Σ — множина спостережуваних подій: виклики API, IPC-повідомлення, proximity-комунікації, мережеві запити, звернення до файлової системи та події пам'яті. Скінченне трасування виконання:

$$\tau = \langle e_1, \dots, e_n \rangle \in \Sigma^*.$$

Поведінка застосунку $\mathcal{A}$ моделюється як мічена система переходів [4]:

$$\mathcal{A} = (S,\ Act,\ \rightarrow,\ s_\emptyset,\ AP,\ L), \tag{1}$$

де S — множина станів, Act — множина дій,
$\rightarrow\ \subseteq S \times Act \times S$ — відношення переходів,
$s_\emptyset$ — початковий стан,
$L : S \rightarrow 2^{AP}$ — функція маркування.

Поведінковий предикат $\phi \in \mathcal{L}_\phi$ обчислюється над τ за допомогою $\tau \vDash \phi$.

Означення 1 (Недокументована можливість). Недокументована можливість F — це предикат $\phi_F \in \mathcal{L}_\phi$ такий, що:

(1) ϕ_F виконується для деякого досяжного трасування $\mathcal{A}$, та
(2) $\phi_F \notin Cn(\Phi_{spec})$.

Означення 2 (Гіпотеза ланцюжка вразливостей). *Ланцюжок гіпотез* $\mathbf{H} = \langle \phi_1, \dots, \phi_k \rangle$ з причинними залежностями $\phi_i \rightarrow \phi_{i+1}$, рівнем впевненості $\kappa \in [\emptyset, 1]$ та контекстом доказів ρ. Ланцюжок підтверджується, коли кожне ϕ_i спостережено та їхній причинний порядок зафіксовано у спільному трасуванні.

2. Запропонований гібридний метод аналізу

2.1. Розподіл аналітичних ролей

Запропонований метод ґрунтується на суворому розподілі ролей (рис. 1). *Аналітик* привносить професійну інтуїцію, обізнаність в домені та семантичне розуміння. Працюючи з початковим корпусом даних $\mathcal{K}(\mathcal{A})$ — структурованим як *статична частина* (декомпільований Java/Kotlin-код, ресурси застосунку, маніфест і графи викликів) та *динамічна частина* (спостереження під час виконання, записи logcat та перехоплений мережевий трафік), що поповнюється з кожним проходом Фази 3 — аналітик формулює ланцюжки гіпотез природною мовою, описуючи підозрювані причинно-наслідкові послідовності поведінки застосунку.

LLM виступає тестувальником гіпотез. Отримавши ланцюжок $\mathbf{H}$ та корпус $\mathcal{K}$, вона обирає стратегію інструментування для кожного ϕ_i, генерує виконавані тестові артефакти, інтерпретує докази, оновлює оцінку впевненості та після підтвердження синтезує PoC. Такий розподіл дозволяє одному аналітику підтримувати черги з десятків ланцюжків і просувати їх паралельно, не стаючи пляшковим горлечком у дослідженні. Після кожного проходу Фази 3 структуровані записи доказів додаються до $\mathcal{K}$: трасування методів з мітками предикатів гіпотез, мережеві перехоплення з анотаціями шаблонів навантаження та знімки пам'яті з перехресними посиланнями на статичні рядкові константи.

Модель є стійкою до збоїв виконання LLM. Коли згенерований скрипт інструментування стикається з помилкою, LLM отримує вивід помилки та намагається самостійно виправитися перед повторним виконанням. LLM інформує аналітика про кожну дію природною мовою — який інструмент використовується, які докази знайдено та яким буде наступний крок. Така прозорість дозволяє аналітику проводити налагодження, якщо LLM застряє на нестандартній помилці або входить у рекурсивний цикл: аналітик може надати відсутній контекст, виправити помилкове припущення або перенаправити ланцюжок — без необхідності мати глибокий досвід інструментування конкретного програмного компоненту.

2.2. Фаза 3: Автоматизоване тестування з кількох джерел

Ключовим внеском запропонованого методу є організація Фази 3 як *середовища збору доказів із кількох джерел*, яким керує виключно LLM. Замість покладання на єдиний інструмент, LLM обирає та комбінує такі канали доказів:

1. **Журнали логів на пристрої.** ADB logcat фіксує вивід журналів застосунку та системи; strace записує послідовності системних викликів для нативних процесів, включно з операціями файлів, сокетів та спільної пам'яті. LLM перевіряє logcat за ключовими предикатами перед застосуванням важчого інструментування, дозволяючи швидко відхиляти низькоймовірні зв'язки.
2. **Перехоплення мережевого трафіку.** Прозорий HTTPS-проксі (mitmproxy) встановлюється шлюзом піддослідного пристрою із сертифікатом у системному сховищі довіри, надаючи повністю розшифровані пари запит/відповідь без модифікації застосунку. Немодифіковані TCP/UDP-сесії та DNS-запити перехоплюються tcpdump для не-HTTP-протоколів, включно з власними бінарними proximity-каналами.
3. **Динамічне інструментування.** Скрипти Frida [5] перехоплюють Java-, Kotlin- та нативні методи під час виконання, фіксуючи аргументи, повернуті значення, ланцюжки викликів та часові мітки. Кілька неконфліктуючих перехоплень виконуються одночасно, їхній порядок спрацювання корелюється для встановлення причинно-наслідкових зв'язків ($\phi_i \rightarrow \phi_{i+1}$).
4. **Налагодження процесу.** JDWP-сумісний відлагоджувач приєднується для покрокового виконання байт-коду Java/Kotlin з умовними точками зупину, похідними від предикатів гіпотез. Нативні бібліотеки відлагоджуються gdb/lldb на цільовому процесі або в ізольованому емуляторі.

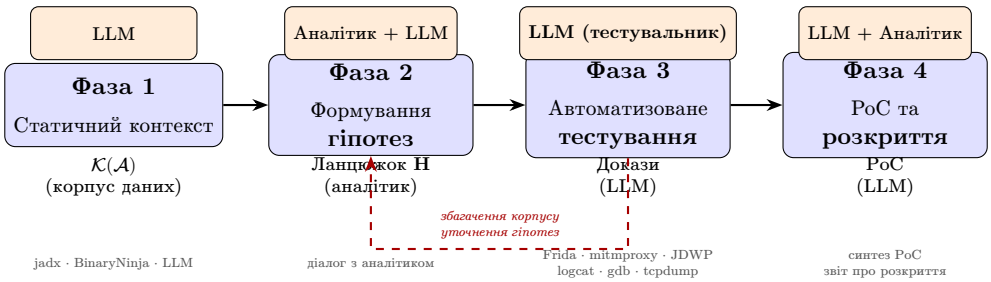

Рис. 1. Конвеєр гібридного аналізу. У Фазі 3 LLM виступає автономним тестувальником гіпотез; участь аналітика обмежена перевіркою.

5. **Аналіз пам'яті.** Аналіз оперативної пам'яті розкриває структури даних, ключі шифрування, чи такі, що не записуються на файлову систему та кешовані облікові дані, відсутні у трасуваннях рівня API. LLM шукає у пам'яті об'єкти, ідентифіковані у попередніх фазах аналізу, поєднуючи статичні та динамічні докази.

LLM синтезує міжрівневі докази для перевірки повних причинно-наслідкових зв'язків: подія ініціалізації внутрішнього компонента (канал 3), запис logcat про відкриття вебсокету (канал 1), отримання даних через вебсокет іншим компонентом (канал 3) та наступний HTTPS-запит на сервер (канал 2) разом підтверджують $\phi_1 \to \phi_2 \to \phi_3$ в межах одного сеансу — кореляція, що вимагала б значних ручних зусиль з розрізненого виводу інструментів.

Алгоритм 1: Цикл тестування Фази 3

Вхід: ланцюжок $\mathbf{H} = \langle \phi_1, \dots, \phi_k \rangle$, корпус $\mathcal{K}$
Вихід: підтверджені гіпотези, докази $\mathcal{E}$, PoC

for $i \leftarrow 1$ **to** k **do**
  $tools \leftarrow \mathsf{LLM.SelectChannels}(\phi_i, \mathcal{K})$
  $cfg \leftarrow \mathsf{LLM.GenerateConfig}(\phi_i, tools)$
  try: $\mathcal{E}_i \leftarrow \mathsf{Execute}(cfg,\ emulator)$
  except ε: $cfg \leftarrow \mathsf{LLM.SelfCorrect}(cfg, \varepsilon)$
  $\kappa \leftarrow \mathsf{UpdateConf}(\kappa, \mathcal{E}_i)$; $\mathcal{K} \leftarrow \mathcal{K} \cup \mathsf{Annotate}(\mathcal{E}_i, \phi_i)$
  if $\kappa \geq \theta_c$: $\mathsf{Confirm}(\phi_i)$; $\mathsf{SynthPoC}(\phi_i, \mathcal{E}_i)$
  elif $\kappa < \theta_r$: $\mathsf{Discard}(\phi_i)$; $\mathsf{Notify}(analyst)$
  else: $\phi_i \leftarrow \mathsf{LLM.Refine}(\phi_i, \mathcal{E}_i)$; **goto** step 2

Рис. 2. Спрощений псевдокод циклу тестування Фази 3.

3. Формальна модель

Конвеєр моделюється як скінченний автомат $\mathcal{M} = (Q, \Sigma_{\mathcal{M}}, \delta, q_0, F)$ [6], де стани відповідають фазам конвеєра. Впевненість підтримується як логарифмічна оцінка:

$$\operatorname{logit}(\kappa') = \operatorname{logit}(\kappa) \ + \sum_{e \in \mathcal{E}_i} w_e, \qquad (2)$$

де $w_e > 0$ для підтверджуючих доказів, $w_e < 0$ — для спростовуючих.

Апостеріорна ймовірність $\kappa' = \sigma(\operatorname{logit}(\kappa'))$ обмежується $[\varepsilon, 1-\varepsilon]$. Якщо $\kappa' < \theta_r$ — ланка ланцюжка відкидається; якщо $\kappa' \geq \theta_c$ — LLM синтезує PoC; інакше LLM уточнює ϕ_i до суворо специфічнішого предиката та повторює. Кількість ітерацій обмежена $N_{\max}$ (табл. 1).

Таблиця 1. Ключові правила переходів скінченного автомата.

Від	Подія	До	Умова
q_{init}	corpus_built	q_{ctx}	$\mathcal{K} \neq \emptyset$
q_{ctx}	chain_formed	q_{rank}	$\lvert\mathbf{H}\rvert > 0$
q_{rank}	predicate_sel.	q_{sel}	$\phi_i \in \mathbf{H}$
q_{sel}	tools_ready	q_{instr}	cfg valid
q_{instr}	evidence_coll.	q_{tr}	$\lvert\tau\rvert > 0$
q_{tr}	confirmed	q_{poc}	$\kappa \geq \theta_c$
q_{tr}	refine	q_{sel}	iter $< N_{\max}$
q_{tr}	rejected	q_{rej}	$\kappa < \theta_r$
q_{poc}	poc_verified	q_{done}	аналітик підтверджує

Теорема 1 (Коректність). Якщо $\mathcal{M}$ досягає q_{done} для ланцюжка $\mathbf{H}$, то кожне ϕ_i виконується на спостережуваному трасуванні зі зафіксованим причинним порядком.

Теорема 2 (Скінченність). $\mathcal{M}$ завершується не більш ніж за $k \cdot N_{\max}$ проходів для будь-якого ланцюжка довжини k.

Теорема 3 (Збіжність). При суворому відношенні специфічності на $\mathcal{L}_\phi$ послідовність уточнень збігається за скінченну кількість кроків [4, 6].

4. Переваги та оцінювання

Таблиця 2 порівнює запропонований метод із наявними підходами за вимірами, релевантними для аналізу з інтенсивним використанням гіпотез.

Таблиця 2. Порівняння підходів до аналізу.

Критерій	Ручний	Пісочниця	Запропон.
Перевірка гіпотез	Дні	N/A	Години
Паралельні ланцюжки	1	N/A	Багато
Міждоменна експертиза	Низьке	Немає	Високе
Джерела доказів	Повний	Фіксов.	Повний
Генерація PoC	Ручна	Немає	Авто.
Компетентність	Експерт	Немає	Помірна
Багаторівневі ланцюжки	Можливо	Ні	Так

Основною перевагою є стиснення циклу перевірки. На практиці, у повсякденній роботі фахівця, перевірка одного міжкомпонентного ланцюжка вимагає налаштування окремих інструментів для кожного рівня, кореляції розрізнених виводів, пошуку навчальних ресурсів та ітерацій після кожного негативного результату — у запропонованій моделі все це делеговано LLM. Роль аналітика зводиться до високорівневої творчої роботи — формулювання гіпотез та остаточної перевірки PoC, що дозволяє паралельно опрацьовувати кілька ланцюжків без необхідності мати експертний рівень знань у кожному з інструментів дослідження та технологічних компонентів застосунку.

4.1. Практичний приклад: спростування гіпотези та відкриття

Наступний приклад ілюструє формування ланцюжка гіпотез, його спростування та випадкове відкриття — закономірність, поширена в реальних дослідженнях.

Аналітик досліджує супутній IoT-застосунок для концентратора розумного будинку. Фаза 1 надає зведення потоків даних: LLM відображає три зовнішні поверхні введення — GATT-характеристику BLE з показниками сенсорів, рядок оголошення пристрою та поле виробничих даних — та простежує кожну через граф викликів. Аналітик зауважує, що вхідні дані конкатенуються у запит до бази даних, та формулює ланцюжок

$$\mathbf{H_1} = \langle \phi_{parse} \to \phi_{inject} \to \phi_{exec} \rangle,$$

підозрюючи ін'єкцію команд через дані сенсорів.

Для тестування $\mathbf{H_1}$ LLM будує автомат стану BLE-протоколу пристрою з витягненого GATT-профілю та керує підробним периферійним пристроєм, що генерує систематично мутовані навантаження — граничні значення, рядки формату, метасимволи оболонки — через кожну характеристику. Перехоплення Frida розміщуються на всіх викликах конкатенації рядків та API бази даних із статичного графу; logcat моніторується на предмет сигнатур збоїв.

Фаза 3 не знаходить ін'єкції команд: шлях даних застосовує санітизацію перед досягненням приймача бази даних. Рівень впевненості для $\mathbf{H_1}$ падає нижче θ_{reject}, і ланцюжок відкидається.

Однак під час симуляції BLE-сканування LLM спостерігає, що ім'я оголошення підробного пристрою передається до компонента WebView без HTML-екранування. Розпізнаючи нову можливість, LLM формулює

$$\mathbf{H_2} = \langle \phi_{scan} \to \phi_{render} \to \phi_{xss} \rangle$$

та перепрограмовує периферійний пристрій на оголошення імені із впровадженим скриптом. WebView виконує навантаження, витягуючи автентифікований токен сесії з `localStorage` та надсилаючи його на тестовий вебсервер. Рівень впевненості досягає $\kappa = 0.96$. LLM синтезує PoC — конфігурацію підробного BLE-периферійного пристрою та сервер захоплення — який аналітик перевіряє у ручному режимі, підтверджуючи витік ключів сесії.

4.2. Обмеження

Попри переваги, три суттєвих обмеження потребують уваги.

Масштабування токенів. Для застосунків з інтенсивною обфускацією статичний корпус насичується перейменованими ідентифікаторами та синтетичним потоком керування; споживання контекстного вікна зростає з розміром застосунку, збільшуючи вартість та затримку Фази 1.

Наївність у визначенні серйозності. LLM схильна надмірно класифікувати знахідки: API-ключі у публічних файлах конфігурації, закодовані URL серверів продакшену або ідентифікатори реклами можуть бути позначені як критичні, хоча є публічними за дизайном або низькоризиковими. Валідація оцінок критичності аналітиком після аналізу залишається необхідною.

Переоцінка впливу. Після початкового аналізу корпусу даних LLM часто продукує надто оптимістичні оцінки ризику — приписуючи високу придатність до експлуатації вразливостям, що за реальних обмежень (розмежування доступу, рівні автентифікації, обмеження сторонніх механізмів захисту)

можуть бути непридатні до експлуатації. Оцінки впливу LLM слід розглядати як верхні межі, що потребують перевірки перед звітуванням.

Висновки

У цій роботі запропоновано гібридну модель, в якій LLM виступає автоматизованим тестувальником гіпотез для пошуку недекларованих можливостей у Андроїд-застосунках. Основними внесками є:

(1) формалізм ланцюжка гіпотез вразливостей на основі поведінкових предикатів над трасуваннями міченої системи переходів (Рівн. (1));

(2) багатоджерельне середовище інструментування (logcat, HTTPS-проксі, Frida, JDWP, аналіз пам'яті), яким повністю керує LLM (Алгоритм 2);

(3) баєсівська логарифмічна модель впевненості (Рівн. (2)), що керує уточненням гіпотез у формально специфікованому скінченному автоматі; та

(4) формальні гарантії коректності, скінченності та збіжності. Практичний приклад демонструє, що модель коректно спростовує хибний ланцюжок гіпотез, паралельно автономно відкриваючи неочевидну міждоменну вразливість — ланцюжок BLE-to-WebView XSS з витоком ключів сесії — без додаткового втручання аналітика.

Подальша робота включає розширення інструментарію LLM у запропонованій моделі для визначення високоефективних підходів на різних етапах аналізу.

Перелік використаних джерел

1. *Google LLC.* Google Play Protect. — 2026. — URL: https://developers.google.com/android/play-protect.
2. A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly / Y. Yao, J. Duan, K. Xu, Y. Cai, E. Sun, Y. Zhang // High-Confidence Computing. — 2024. — Vol. 4, no. 2. — DOI: 10.1016/j.hcc.2024.100211.
3. *Anthropic.* Claude: Tool Use and Agentic Capabilities. — 2024. — URL: https://docs.anthropic.com/en/docs/build-with-claude/tool-use.
4. Reactive Systems: Modelling, Specification and Verification / L. Aceto, A. Ingólfsdóttir, K. G. Larsen, J. Srba. — Cambridge University Press, 2007. — DOI: 10.1017/CBO9780511814105.
5. *Frida Contributors.* Frida: Dynamic Instrumentation Toolkit for Developers, Reverse-Engineers, and Security Researchers. — 2024. — URL: https://frida.re.
6. *Clarke E. M., Grumberg O., Peled D. A.* Model Checking. — Cambridge, MA : MIT Press, 1999.