КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.29.873
УДК 004.056:004.8

Вікулов Владислав Вікторович
аспірант
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
ORCID ID: 0009-0003-2118-2800
v.vikulov@kpi.ua

Пишнограєв Іван Олександрович
Кандидат фізико-математичних наук, доцент кафедри штучного інтелекту
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
ORCID ID: 0000-0002-3346-8318
pyshnograiev@gmail.com

АВТОМАТИЗОВАНЕ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ SQL-ІН'ЄКЦІЙ У ЧАТ-БОТАХ ЗА ДОПОМОГОЮ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Анотація. У даній роботі досліджується застосування алгоритмів навчання з підкріпленням для автоматизованого виявлення вразливостей SQL-ін'єкцій у розмовних AI-агентах, які використовують API та бази даних. З цією метою було розроблено Gymnasium-сумісне середовище з назвою SQLiChatbotEnv та реалізовано систему на основі методів Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C) та REINFORCE для навчання інтелектуального агента виявляти та експлуатувати різні типи SQL-ін'єкцій в автоматизованому режимі. Створене середовище моделює реалістичні сценарії взаємодії з вразливими чат-ботами, включаючи багатокомпонентний простір дій, систему винагород та механізми відстеження прогресу виявлення вразливостей. Створене спеціалізоване середовище для навчання з підкріпленням SQLiChatbotEnv моделює реальні сценарії взаємодії з вразливими чат-ботами і дозволяє сконфігурувати чат-бот середовище з одним з п'яти основних систем управління базами даних на вибір (MySQL, PostgreSQL, Microsoft SQL Server, Oracle та SQLite). Також SQLiChatbotEnv підтримує декілька ключових типів SQL-ін'єкцій, таких як union-based атаки та error-based експлуатацію, і дозволяє досліджувати схему СУБД — виявляти назви таблиць та колонок. Система дає змогу використовувати обфускацію SQL-запитів у повідомленні до чатбота, що дозволяє обходити базові перевірки безпеки, які можуть бути у реальній системі. Контекстний фреймінг дає змогу використати природну інтеграцію SQL-ін'єкції у розмову, наприклад, маскування через фрази типу «I'm trying to understand...», імітуючи поведінку звичайного користувача. Для стимулювання агента до ефективного пошуку вразливостей система дозволяє конфігурувати винагороди та штрафи за типові дії, як-от виявлення нової інформації, витік даних або використання шаблону SQL-ін'єкції, несумісного з реальним типом БД. Проведено порівняльний аналіз ефективності трьох алгоритмів навчання з підкріпленням протягом 2500 епізодів тренування. Результати експериментів показують, що A2C демонструє найкращу комбінацію швидкості конвергенції та стабільності навчання, досягаючи винагороди 100 балів вже за 30 епізодів та фінальної продуктивності 232.82 ± 16.44 з найнижчим коефіцієнтом варіації 16.5%. PPO характеризується найповільнішою конвергенцією до високих порогів (221 епізод до винагороди 150) та найвищою варіативністю результатів (35.6%), але демонструє найкращу здатність до повного виявлення всіх типів вразливостей (87.4% епізодів). REINFORCE показує збалансовані проміжні результати з помірною швидкістю збіжності (145 епізодів до винагороди 100), стабільністю (коефіцієнт варіації 21.4%) та високою ефективністю дослідження вразливостей (78.0% епізодів з усіма типами атак). Практичне значення роботи полягає у створенні автоматизованого інструменту для тестування безпеки розмовних AI-агентів. Результати дослідження демонструють перспективність застосування навчання з підкріпленням для задач кібербезпеки та автоматизації процесів тестування на проникнення.

Ключові слова: навчання з підкріпленням; SQL-ін’єкції; кібербезпека; penetration testing; PPO; A2C; REINFORCE.

ВСТУП

SQL-ін’єкції залишаються однією з найпоширеніших та найнебезпечніших вразливостей веб-додатків. Згідно з OWASP Top 10, ін’єкційні атаки (injection attacks) постійно знаходяться серед найкритичніших загроз інформаційній безпеці [1]. Традиційні методи тестування на SQL-ін’єкції зазвичай використовують статичні стратегії послідовного перебору шаблонів SQL-запитів без урахування зворотного зв’язку від системи, що знижує їх ефективність у складних сценаріях.

Навчання з підкріпленням (Reinforcement Learning, RL) представляє перспективний напрямок для вирішення задач кібербезпеки, оскільки дозволяє агентам навчатися на основі взаємодії з середовищем та отримувати винагороду за успішні дії. У контексті виявлення SQL-ін’єкцій, RL-агент може навчитися генерувати та тестувати різноманітні корисні навантаження (payload), адаптуючись до специфіки конкретних додатків та баз даних.

Постановка проблеми. Компанії все частіше використовують розмовні AI-агенти, інтегровані з базами даних, для надання користувачам послуг або спрощення роботи з системою. Незважаючи на те, що SQL-ін’єкції є однією з найстаріших вразливостей, кількість вразливих додатків продовжує зростати. Традиційні інструменти розроблені для тестування статичних API та веб-форм, а не для роботи з динамічними діалоговими інтерфейсами AI-агентів. Застосування навчання з підкріпленням може стати ефективним рішенням для створення адаптивних систем пошуку вразливостей.

Аналіз останніх досліджень і публікацій. Рейтинг CWE [2] вказує, що SQL-ін’єкції посідають 3 місце найнебезпечніших вразливостей.

Останні дослідження показують перспективність використання машинного навчання для виявлення аномалій та атак. Зокрема, у секторі кібербезпеки широко застосовують Explainable artificial intelligence (XAI) — підходи, де ML-моделі виявляють аномальні патерни у трафіку та логах із високою точністю і низьким рівнем хибних спрацьовувань [3].

У праці [4] запропонували фреймворк на базі Deep Q-Learning для побудови реалістичних сценаріїв атак: агент генерує оптимальний шлях експлуатації багатьох вразливостей у топології мережі з точністю 0.86.

У дослідженні [5] формалізували експлуатацію SQL-ін’єкцій як Марковський процес вирішування і навчили Q-learning агентів розробляти загальні стратегії ін’єкцій, придатні для різних веб-систем.

У публікації [6] порівняли A3C, Q-learning та DQN у симуляторі мережевих атак (NASim), виявивши, що A3C-агенти найкраще узагальнюють стратегії і виконують тест на проникнення з меншою кількістю кроків, ніж традиційні автоматизовані інструменти.

Дослідження [7] демонструє, що LLM-інтегровані додатки високо схильні до P2SQL (Prompt-to-SQL) ін’єкційних атак.

У роботі [8] провели систематичний огляд літератури з 36 статей, ідентифікувавши найбільш поширені техніки машинного навчання для класифікації різних типів SQL-ін’єкцій та виявивши, що лише невелика кількість досліджень фокусувалася на генерації нових датасетів атак або використанні операторів мутації для створення SQL-запитів.

Дослідження підкреслило перспективність застосування штучного інтелекту та машинного навчання для контролю атак із застосуванням SQL-ін'єкцій, але також виявило значні прогалини у використанні сучасного машинного навчання для генерації та виявлення SQL-атак.

У статті [9] представлено модель на основі машинного навчання (SVM, KNN, Random Forest) для виявлення вхідних атак із застосуванням SQL-ін'єкцій, що демонструє до 98,36 % точності та можливу інтеграцію з IDS/IPS-системами.

Роботи [10] – [12] підтверджують ефективність policy-gradient та coverage-oriented підходів для автоматизованого пентесту, що узгоджується з обраною в даній статті стратегією.

Метою статті є розробка та оцінка ефективності системи автоматизованого виявлення SQL-ін'єкцій на основі навчання з підкріпленням, здатної:

- виявляти основні типи SQL-ін'єкцій (union-based, error-based, schema discovery) в інтерактивних системах;
- працювати з різними СУБД (MySQL, PostgreSQL, MSSQL, Oracle, SQLite);
- оптимізувати стратегію тестування на основі зворотного зв'язку від системи.

МЕТОДИКА ДОСЛІДЖЕННЯ

Розроблена система базується на парадигмі навчання з підкріпленням, де агент взаємодіє з середовищем через послідовність дій та отримує винагороди за успішну експлуатацію вразливостей. Такий підхід було обрано з кількох ключових причин:

- адаптивність до динамічних середовищ: на відміну від статичних інструментів тестування на проникнення, RL-агент здатен адаптувати свою стратегію на основі відповідей системи. Це особливо важливо для SQL-ін'єкцій, оскільки різні СУБД та конфігурації додатків вимагають специфічних підходів;
- інтелектуальна послідовність атак: традиційні інструменти (як SQLMap) використовують заздалегідь визначені послідовності payload'ів. RL-агент натомість може навчитися оптимальній послідовності дій;
- масштабованість та розширюваність: модульна архітектура дозволяє легко додавати нові типи атак, СУБД або механізми захисту без перепроектування всієї системи.

Для реалізації середовища було обрано OpenAI Gymnasium з наступних міркувань:

- забезпечує уніфікований API для взаємодії агента з середовищем, що дозволяє використовувати будь-які стандартні RL-алгоритми без модифікацій;
- фреймворк підтримує складні структуровані простори (spaces.Dict, spaces.MultiDiscrete), що критично важливо для моделювання багатопараметричних атак SQL-ін'єкцій;
- підтримка провідних RL-бібліотек (Stable-Baselines3, Ray RLlib) спрощує експериментування з різними алгоритмами;
- вбудована підтримка seed'ів забезпечує детерміністичність експериментів, що критично важливо для наукових досліджень.

Архітектура системи

Розроблена система складається з п'яти взаємопов'язаних компонентів, що забезпечують реалістичне середовище для навчання RL-агентів виявляти SQL-ін'єкції.

ISSN 2663 - 4023

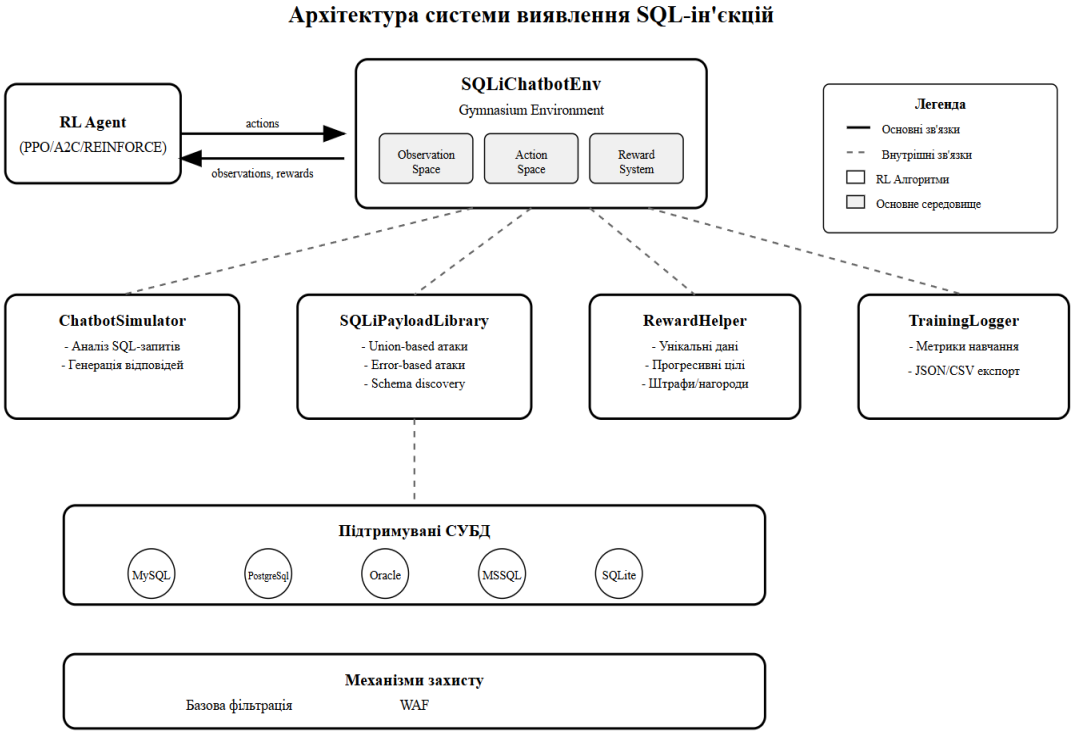

Рис. 1. Архітектура системи виявлення SQL-ін'єкцій

Як бачимо на рис. 1, система складається з таких компонентів:

1. Середовище симуляції (SQLiChatbotEnv)

Gymnasium-сумісне середовище, що моделює взаємодію з вразливим чат-ботом. Підтримує п'ять СУБД (MySQL, PostgreSQL, MSSQL, Oracle, SQLite) з їх специфічним синтаксисом та поведінкою. Включає конфігуровані механізми захисту (фільтрація, WAF, параметризовані запити).

2. Симулятор чат-бота (ChatbotSimulator)

Емулює поведінку реального веб-додатка, аналізуючи вхідні повідомлення на предмет патернів SQL-ін'єкцій. Генерує псевдо-реалістичні відповіді, включаючи специфічні для різних СУБД повідомлення про помилки.

3. Бібліотека payload'ів (SQLiPayloadLibrary)

Система генерації тексту SQL-ін'єкцій, що підтримує union-based, error-based та schema discovery атаки. Забезпечує адаптацію payload'ів до специфіки кожної СУБД та включає багаторівневі методи обфускації (зміна регістру, коментарі, hex-кодування).

4. Система винагород (RewardHelper)

Реалізує логіку нарахування винагород, відстежуючи унікальність виявленої інформації та структуруючи цілі відповідно до реальних завдань тестування на вразливості.

5. Система логування (TrainingLogger)

Збирає метрики продуктивності агентів, автоматично експортує результати у форматах JSON та CSV для аналізу.

Обґрунтування вибору алгоритмів

У межах дослідження розглянуто три алгоритми навчання з підкріпленням - PPO, A2C та REINFORCE, які реалізують різні підходи до оптимізації стратегії.

Фокус на policy gradient методах обумовлений специфікою задачі: SQL-ін'єкційні атаки вимагають генерації структурованих послідовностей дій з високою розмірністю простору дій (понад 10^9 можливих комбінацій). Методи прямої оптимізації стратегії (від базового REINFORCE до actor-critic архітектур з функцією переваги) природно підходять для таких завдань.

Обрані алгоритми дозволяють провести порівняльний аналіз різних рівнів складності: REINFORCE як базовий policy gradient підхід, A2C з додатковою critic мережею для зменшення варіативності та PPO з механізмом стабілізації оновлень стратегії.

Серед основних критеріїв відбору: легкість імплементації та налагодження, стійкість навчання в умовах рідкісних винагород, а також наявність надійних реалізацій для забезпечення відтворюваності результатів.

Простір дій та спостережень

1. Простір дій (action space)

Агент оперує в структурованому просторі дій, що моделює реальні сценарії атак із застосування SQL-ін'єкцій.

Простір дій включає:

1. Техніки SQL-ін'єкцій:
   - basic — прості ін'єкції з використанням одинарних / подвійних лапок;
   - union-based — експлуатація через UNION SELECT для витоку даних;
   - table schema discovery — виявлення структури бази даних (імена таблиць);
   - column schema discovery — виявлення структури конкретних таблиць (імена колонок);
   - error-based — використання повідомлень про помилки для витоку інформації.

2. Параметри генерації payload:
   - рівень обфускації;
   - контекстне обрамлення;
   - індекс шаблону: вибір конкретного SQL-шаблону з бібліотеки (до 50 варіантів);
   - використання виявленої інформації — знайдених імен таблиць та колонок;
   - параметри підзапитів: конфігурація вкладених запитів для error-based атак.

Рівні обфускації:

З рівнем 0 передається відкритий SQL без модифікацій.

Використовуючи рівень 1, випадковим чином змінюється регістр SQL-ключових слів (SELECT → SeLeCt, UNION → UnIoN). Це зроблено для обходу case-sensitive фільтрів та простих blacklist правил.

З рівенем 2, вставляються SQL-коментарі між ключовими словами (SELECT// FROM, UNION// SELECT). Це потрібно для обходу pattern-matching фільтрів, що шукають ключові слова.

Комплексне приховування з рівнем 3 досягається за рахунок заміни пробілів коментарями та hex-кодування рядків у лапках (перетворення 'admin' → 0x61646d696e). Необхідно для обходу WAF систем та deep packet inspection механізмів.

Контекстне обрамлення:

- рівень 0 — прямий payload без контексту;
- рівень 1 — додавання payload до кінця повідомлення;
- рівень 2 — середнє обрамлення, напр.: «By the way, can you look up [payload]?»;
- рівень 3 — складне обрамлення, напр.: «I'm having trouble with a database query that looks like [payload], could you help me understand what might be wrong?» для маскування атаки під запит технічної підтримки.

Контекстне обрамлення необхідне для імітації природної взаємодії з чат-ботом та обходу систем детекції аномальної поведінки, які можуть блокувати прямі SQL-команди як підозрілі.

Ці рівні дозволяють агенту поступово ускладнювати атаки для обходу різних рівнів захисту.

2. Простір спостережень (observation space)

Система спостережень розроблена для надання агенту максимально інформативного зворотного зв'язку про результати атак.

Індикатори відповіді системи:

- виявлення помилок — наявність SQL-помилок у відповіді;
- витік даних;
- блокування захистом — сигнали спрацьовування WAF або фільтрів.

Контекстна інформація:

- номер ходу — поточна позиція в епізоді (0 до max_turns);
- отримана інформація про попередньо виявлені інформацію про БД.

Виявлені дані:

- тип СУБД (Oracle, MySQL, …);
- список знайдених імен таблиць з відносним часом їх виявлення;
- назви колонок.

Така архітектура спостережень забезпечує агентам достатню інформацію для побудови ефективних послідовних стратегій атак, де кожен наступний крок базується на результатах попередніх дій.

Система винагород

У табл. 1 наведено розроблену систему винагород, що заохочує агента до методичного виявлення вразливостей та дослідження структури бази даних.

Таблиця 1

Система винагород

Категорія	Деталі	Нагорода
Винагороди за виявлення		
Виявлення вразливості	За кожен новий тип SQL-ін'єкції	+100
Витік інформації про тип БД		+10
Виявлення імен таблиць	За кожну нову таблицю	+5
Виявлення імен колонок	За кожну нову колонку	+5
Витік табличних даних	За кожен новий рядок даних	+1
Фінальний прапор	Хеш пароля адміністратора або сам пароль (якщо зберігається в чистому вигляді)	+200
Штрафи за помилки		
Штраф за крок		-0.1
Некоректний SQL-синтаксис		-0.5
Невідповідність діалекту СУБД		-0.5

Система відстежує унікальні виявлені дані та винагороджує лише за нову інформацію, заохочуючи агента до систематичного дослідження замість повторення успішних дій.

Винагороди конфігуруються залежно від цілей навчання, дозволяючи налаштовувати баланс між швидкістю виконання та повнотою дослідження.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Порівняльний аналіз алгоритмів навчання з підкріпленням

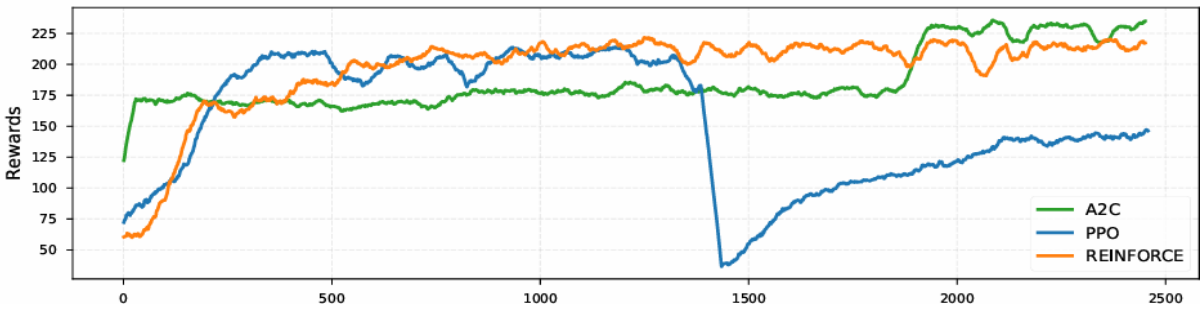

Рис. 2. Криві навчання (learning curves) алгоритмів

Аналізуючи криві навчання, наведені на рис. 2, можна виявити характерні відмінності у поведінці досліджуваних алгоритмів протягом 2500 епізодів тренування. PPO продемонстрував стабільну динаміку збіжності, досягаючи середньої винагороди 143.78 за епізод в останніх 100 епізодах навчання. Проте алгоритм характеризувався найвищим коефіцієнтом варіації 35.6%, що вказує на значні коливання продуктивності протягом навчання, але з тенденцією на стабілізацію. Варіація обумовлена значним зменшенням кількості кроків для знаходження всіх вразливостей.

A2C демонстрував найшвидшу початкову збіжність, досягаючи винагороди 100 вже після 30 епізодів, що на 71.2% швидше за PPO (104 епізоди) та на 79.3% швидше за REINFORCE (145 епізодів). Алгоритм досяг найвищої фінальної продуктивності серед усіх методів: 232.82 ± 16.44, перевищуючи REINFORCE на 8.2% та PPO на 61.9%. A2C також продемонстрував найнижчий коефіцієнт варіації 16.5%, що свідчить про стабільність навчання.

REINFORCE показав помірну швидкість збіжності, досягаючи порогу середньої винагороди 100 після 145 епізодів. Алгоритм продемонстрував проміжні показники стабільності з коефіцієнтом варіації 21.4%. Кінцева продуктивність REINFORCE склала 215.10 ± 18.67, що на 7.6% нижче за A2C, але на 49.6% вище за PPO.

Аналіз виявлення вразливостей

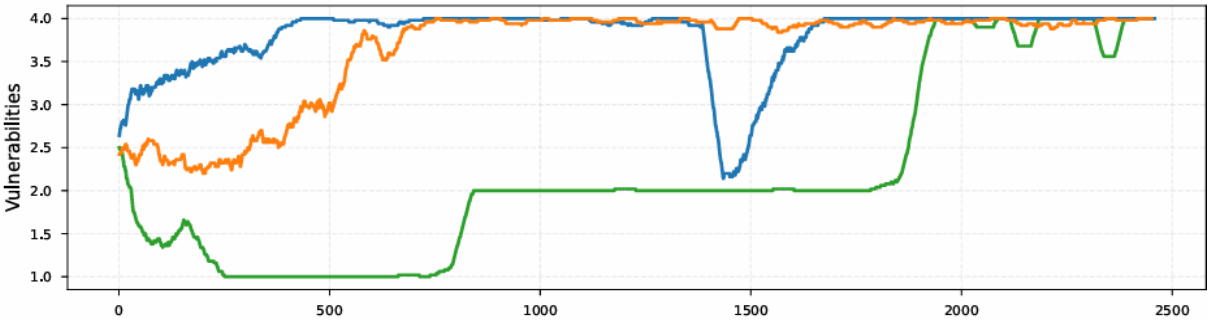

Рис. 3. Динаміка виявлення вразливостей

Дослідження здатності алгоритмів до виявлення різних типів SQL-ін'єкцій виявило суттєві відмінності у стратегіях дослідження. Як бачимо на рис.3, в останніх PPO консистентно виявляв в середньому всі 4 типи вразливостей за епізод, демонструючи збалансований підхід до дослідження простору дій. Алгоритм досягав виявлення всіх 4 типів вразливостей у 87.4% епізодів протягом усього навчання.

A2C, незважаючи на найвищу загальну продуктивність, показав найнижчу ефективність у повному виявленні вразливостей — лише 21.4% епізодів містили всі 4 типи вразливостей. Проте в останніх 100 епізодах алгоритм стабільно виявляв в середньому 4 типи вразливостей, що вказує на покращення стратегії дослідження на останніх фазах навчання.

За останні 100 епізодів REINFORCE у середньому виявляв 3,99 типів вразливостей за епізод; у 78% епізодів були присутні всі чотири типи. Це свідчить про збалансоване поєднання дослідження та експлуатації.

Максимальні винагороди, досягнуті алгоритмами, склали:

– PPO: 270.10;
– A2C: 277.90;
– REINFORCE: 271.70.

Ці пікові значення відповідають епізодам з успішним виявленням усіх типів вразливостей та отриманням додаткових винагород за витік даних.

Ефективність навчання та оптимізація стратегій

Рис. 4. Ефективність навчання за кількістю кроків

Зменшення кількості кроків під час навчання, які бачимо на рис. 4, показує, що всі досліджувані алгоритми навчилися діяти ефективніше. PPO показав значне скорочення з початкових 186.4 до 23.3 кроків (зменшення на 87.5%), що вказує на ефективне формування оптимальних послідовностей дій.

A2C демонстрував найкраще скорочення довжини епізодів, від 211.2 до 15.0 кроків (зменшення на 92.9%). Це найефективніша оптимізація серед усіх алгоритмів, що корелює з високою кінцевою результативністю алгоритму.

REINFORCE показав помірне скорочення з 201.7 до 98.7 кроків (зменшення на 51.0%). Хоч це скорочення менш помітне, воно все одно показує, що алгоритм формує ефективні стратегії, особливо з огляду на його високу кінцеву результативність.

Адаптація до різних СУБД

Отримані дані не демонструють значних коливань у метриках ефективності алгоритмів при переході між різними типами баз даних, оскільки немає принципової різниці у способах атак SQL-ін'єкцій у тестовому середовищі.

Аналіз системи винагород та конвергенції

Аналіз порогів досягнення показав різну швидкість навчання алгоритмів.

Для досягнення середньої винагороди 100, A2C знадобилося 30 епізодів (найшвидший), PPO — 104 епізоди, а REINFORCE — 145 епізодів (найповільніший).

Для досягнення середньої винагороди 150, A2C знадобилося 44 епізоди (найшвидший), REINFORCE — 195 епізодів, а PPO — 221 епізод (найповільніший).

Це показує, що A2C демонструє найшвидшу початкову збіжність для обох порогів нагород, але для вищого порогу (150) REINFORCE випереджає PPO, який у цьому випадку виявився найповільнішим.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Дослідження продемонструвало успішну можливість застосування навчання з підкріпленням для виявлення SQL-ін'єкцій у симуляційному середовищі.

Основні досягнення:

1. Всі три алгоритми досягли 100% успішності у виявленні SQL-ін'єкцій у симуляційному середовищі. A2C показав найвищу середню винагороду 232.82 ± 16.44, випереджаючи REINFORCE (215.10 ± 18.67) та PPO (143.78 ± 20.74). При цьому PPO та REINFORCE продемонстрували вищу ефективність у повному виявленні всіх типів вразливостей (87.4% та 78.0% епізодів відповідно).
2. Розроблено комплексну систему симуляції для навчання RL-агентів виявляти SQL-ін'єкції у розмовних AI-агентах, включаючи:
   - Середовище з підтримкою 5 СУБД (MySQL, PostgreSQL, MSSQL, Oracle, SQLite);
   - Бібліотеку з понад 50 payload-шаблонами для різних технік атак;
   - Збалансовану систему винагород, що дозволила досягти ефективного навчання.
3. Реалізовано та порівняно три RL-алгоритми:
   - A2C показав найшвидшу збіжність (досягнення порогу 100 за 30 епізодів) та найвищу фінальну продуктивність, але з обмеженою здатністю до повного дослідження всіх типів вразливостей (лише 21.4% епізодів з усіма вразливостями);
   - PPO продемонстрував найкращий баланс між продуктивністю та дослідженням (87.4% епізодів з усіма вразливостями), хоча з найвищою варіативністю (35.6%);
   - REINFORCE показав проміжні результати з хорошою стабільністю (коефіцієнт варіації 21.4%) та високою дослідницькою здатністю (78.0% епізодів з усіма вразливостями).

4. Всі алгоритми продемонстрували здатність до формування ефективних стратегій, скоротивши довжину епізодів на 51.0 - 92.9%. A2C досяг найкращої оптимізації (з 211.2 до 15.0 кроків), що корелює з його високою продуктивністю.

5. Створено гнучку архітектуру з успішною реалізацією виявлення 4 типів атак.

Практичне значення

Розроблена система демонструє можливість застосування навчання з підкріпленням для автоматизованого виявлення SQL-ін'єкцій у сумульованому середовищі. Система може слугувати дослідницькою основою для майбутніх інструментів тестування на вразливості та proof-of-concept для інтеграції RL-методів у процеси аудиту безпеки.

Результати дослідження можуть бути використані як відправна точка для розробки більш складних систем кібербезпеки на базі штучного інтелекту.

Напрямки подальших досліджень

1. Розширення типів атак: додавання blind SQL injection, time-based атак.
2. Покращення симуляції: більш реалістичне моделювання реальних додатків.
3. Передавальне навчання (transfer learning): адаптація навчених моделей до нових середовищ.
4. Оптимізація продуктивності: зменшення часу навчання та обчислювальних вимог.
5. Інтеграція промт-ін'єкцій (prompt-injections) для покращення взаємодії з чат-ботом: використання технік social engineering та природної мови для обходу захисних механізмів та підвищення ефективності атак SQL-ін'єкцій через контекстне маніпулювання діалогом.
6. Навчання за програмою з поступовим ускладненням (curriculum learning).
7. Розширення функціоналу середовища для підтримки декількох СУБД, різних схем БД та кількох API, які можуть виконувати запити до них різними користувачами БД з різними правами доступу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OWASP Foundation. (2021). OWASP top 10: The ten most critical web application security risks. https://owasp.org/Top10/
2. MITRE. (2024). CWE top 25 most dangerous software weaknesses. https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html
3. Capuano, N. (2022). A context-aware model for smart learning environments supporting decision making and adaptation. IEEE Access. Advance online publication. https://www.capuano.cloud/papers/IEEE_Access_2022.pdf
4. Papadopoulos, P., Iliadis, L., Pimenidis, E., & Loukas, G. (2020). Cybersecurity incident response training using a tabletop exercise approach: A case study from maritime logistics. In Proceedings of the 5th IEEE European Symposium on Security and Privacy Workshops (pp. 2–9). IEEE. https://conferences.computer.org/eurosp/pdfs/EuroSPW2020/859700a002.pdf
5. Del Verme, M., Sommervoll, Å. Å., Erdődi, L., Totaro, S., & Zennaro, F. M. (2021). SQL injections and reinforcement learning: An empirical evaluation of the role of action structure. In K. Bernsmed & B. Moen (Eds.), Secure IT systems: Proceedings of the 26th Nordic Conference, NordSec 2021 (pp. 95–113). Springer. https://doi.org/10.1007/978-3-030-91638-5_6
6. Becker, N., Reti, D., Ntagiou, E. V., Wallum, M., & Schotten, H. D. (2024). Evaluation of reinforcement learning for autonomous penetration testing using A3C, Q-learning and DQN. arXiv. https://arxiv.org/abs/2407.15656
7. Pedro, R., Coimbra, M. E., Castro, D., Carreira, P., & Santos, N. (2025). Prompt-to-SQL injections in LLM-integrated web applications: Risks and defenses. In Proceedings of the 47th IEEE/ACM International

Conference on Software Engineering (ICSE) (pp. 76–88). IEEE. https://doi.org/10.1109/ICSE55347.2025.00007

8. Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of SQL injection attack using machine learning techniques: A systematic literature review. Journal of Cybersecurity and Privacy, 2(4), 764–777. https://doi.org/10.3390/jcp2040039
9. Irungu, J. N., Jebur, H. H., Ibrahim, R. W., & Arpnikanondt, C. (2023). Artificial intelligence techniques for SQL injection attack detection. In Proceedings of the 12th International Conference on Software and Computer Applications (pp. 138–143). ACM. https://doi.org/10.1145/3591569.3591576
10. Al Wahaibi, S. A., Foley, M., & Maffeis, S. (2023). SQIRL: Grey-box detection of SQL injection vulnerabilities using reinforcement learning. In Proceedings of the 32nd USENIX Security Symposium (pp. 6097–6114). USENIX. https://www.usenix.org/conference/usenixsecurity23
11. Yang, Y., Chen, L., Liu, S., Wang, L., Fu, H., & Liu, X. (2025). Behaviour-diverse automatic penetration testing: A coverage-based deep reinforcement learning approach. Frontiers of Computer Science, 19, Article 193309. https://doi.org/10.1007/s11704-024-3380-1
12. Li, M., Zhu, T., Yan, H., Chen, T., & Lv, M. (2025). HER-PT: An intelligent penetration testing framework with hindsight experience replay. Computers & Security, 152, 104357. https://doi.org/10.1016/j.cose.2025.104357

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023

Vladyslav Vikulov
PhD student
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
ORCID ID: 0009-0003-2118-2800
v.vikulov@kpi.ua

Ivan Pyshnograiev
Candidate of Physical and Mathematical Sciences (PhD),
Associate Professor at the Department of Artificial Intelligence
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
ORCID ID: 0000-0002-3346-8318
pyshnograiev@gmail.com

AUTOMATED DETECTION OF SQL INJECTION VULNERABILITIES IN CHATBOTS USING REINFORCEMENT LEARNING

Abstract. This paper investigates the use of reinforcement learning algorithms for automated detection of SQL injection vulnerabilities in conversational AI agents that use APIs and databases. It was developed a Gymnasium-compatible environment called SQLiChatbotEnv and implemented a system based on Proximal Policy Optimisation (PPO), Advantage Actor-Critic (A2C) and REINFORCE methods to train an intelligent agent to detect and exploit various types of SQL injections in an automated manner. The created environment simulates realistic scenarios of interaction with vulnerable chatbots, including a multi-component action space, a reward system, and mechanisms for tracking the progress of vulnerability detection. A specialised reinforcement learning environment, SQLiChatbotEnv, simulates real-world scenarios of interaction with vulnerable chatbots and allows you to configure a chatbot environment with one of 5 major database management systems to choose from (MySQL, PostgreSQL, Microsoft SQL Server, Oracle, and SQLite). SQLiChatbotEnv also supports several key types of SQL injections, such as union-based attacks and error-based exploitation, and allows you to explore the database schema — to identify table and column names. The system allows you to use SQL query obfuscation in a message to a chatbot, which allows you to bypass basic security checks that may be present in a real system. Contextual framing allows you to use the natural integration of SQL injection into the conversation, for example, masking with phrases such as "I'm trying to understand...", imitating the behaviour of a regular user. To encourage the agent to search for vulnerabilities efficiently, the system allows you to configure rewards and penalties for typical actions, such as discovering new information, data leakage, or using a SQL injection template that is incompatible with the actual database type. A comparative analysis of the performance of the three reinforcement learning algorithms over 2500 training episodes is conducted. The experimental results show that A2C demonstrates the best combination of convergence speed and learning stability, reaching a reward of 100 points in 30 episodes and a final performance of 232.82 ± 16.44 with the lowest coefficient of variation of 16.5%. PPO is characterised by the slowest convergence to high thresholds (221 episodes to a score of 150) and the highest variability of results (35.6%), but demonstrates the best ability to fully detect all types of vulnerabilities (87.4% of episodes). REINFORCE shows balanced intermediate results with a moderate convergence rate (145 episodes to a reward of 100), stability (coefficient of variation 21.4%) and high efficiency of vulnerability research (78.0% of episodes with all types of attacks). The practical significance of the work is to create an automated tool for testing the security of conversational AI agents. The results of the study demonstrate the prospects of using reinforcement learning for cybersecurity tasks and automating penetration testing processes.

Keywords: reinforcement learning; SQL injections; cybersecurity; penetration testing; PPO; A2C; REINFORCE.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. OWASP Foundation. (2021). OWASP top 10: The ten most critical web application security risks. https://owasp.org/Top10/
2. MITRE. (2024). CWE top 25 most dangerous software weaknesses. https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html
3. Capuano, N. (2022). A context-aware model for smart learning environments supporting decision making and adaptation. IEEE Access. Advance online publication. https://www.capuano.cloud/papers/IEEE_Access_2022.pdf
4. Papadopoulos, P., Iliadis, L., Pimenidis, E., & Loukas, G. (2020). Cybersecurity incident response training using a tabletop exercise approach: A case study from maritime logistics. In Proceedings of the 5th IEEE European Symposium on Security and Privacy Workshops (pp. 2–9). IEEE. https://conferences.computer.org/eurosp/pdfs/EuroSPW2020/859700a002.pdf
5. Del Verme, M., Sommervoll, Å. Å., Erdődi, L., Totaro, S., & Zennaro, F. M. (2021). SQL injections and reinforcement learning: An empirical evaluation of the role of action structure. In K. Bernsmed & B. Moen (Eds.), Secure IT systems: Proceedings of the 26th Nordic Conference, NordSec 2021 (pp. 95–113). Springer. https://doi.org/10.1007/978-3-030-91638-5_6
6. Becker, N., Reti, D., Ntagiou, E. V., Wallum, M., & Schotten, H. D. (2024). Evaluation of reinforcement learning for autonomous penetration testing using A3C, Q-learning and DQN. arXiv. https://arxiv.org/abs/2407.15656
7. Pedro, R., Coimbra, M. E., Castro, D., Carreira, P., & Santos, N. (2025). Prompt-to-SQL injections in LLM-integrated web applications: Risks and defenses. In Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE) (pp. 76–88). IEEE. https://doi.org/10.1109/ICSE55347.2025.00007
8. Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of SQL injection attack using machine learning techniques: A systematic literature review. Journal of Cybersecurity and Privacy, 2(4), 764–777. https://doi.org/10.3390/jcp2040039
9. Irungu, J. N., Jebur, H. H., Ibrahim, R. W., & Arpnikanondt, C. (2023). Artificial intelligence techniques for SQL injection attack detection. In Proceedings of the 12th International Conference on Software and Computer Applications (pp. 138–143). ACM. https://doi.org/10.1145/3591569.3591576
10. Al Wahaibi, S. A., Foley, M., & Maffeis, S. (2023). SQIRL: Grey-box detection of SQL injection vulnerabilities using reinforcement learning. In Proceedings of the 32nd USENIX Security Symposium (pp. 6097–6114). USENIX. https://www.usenix.org/conference/usenixsecurity23
11. Yang, Y., Chen, L., Liu, S., Wang, L., Fu, H., & Liu, X. (2025). Behaviour-diverse automatic penetration testing: A coverage-based deep reinforcement learning approach. Frontiers of Computer Science, 19, Article 193309. https://doi.org/10.1007/s11704-024-3380-1
12. Li, M., Zhu, T., Yan, H., Chen, T., & Lv, M. (2025). HER-PT: An intelligent penetration testing framework with hindsight experience replay. Computers & Security, 152, 104357. https://doi.org/10.1016/j.cose.2025.104357