УДК 004.056:004.451.3

***Гребенюк Д. М., магістрант***
*Черкаський державний технологічний університет*

## HONEYPOT-ПЛАТФОРМА ДЛЯ ВИЯВЛЕННЯ АТАК

**Актуальність.** Honeypot-системи - це «приманки» в кібербезпеці, що імітують важливі ресурси - сервери або сервіси з відкритими вразливостями з метою привабити зловмисників. Вони ізольовані від продуктивної мережі і постійно моніторяться, в результаті все, що робить атакуючий фіксується та аналізується. Основна мета - відволікти атаки від реальних критичних систем і визначити методи та інструменти зловмисників [1]. Витяги з активності в Honeypot допомагають отримати цінну розвідувальну інформацію про техніки зловмисників та оновити сигнатури захисту, виправити виявленні вразливості, одночасно зменшуючи кількість хибних спрацювань у системах виявлення. Сучасні дослідження підтверджують, що застосування honeypot-підходів дає змогу своєчасно виявляти реальні загрози та покращувати захист інформаційних ресурсів [5; 7].

**Мета і завдання дослідження.** Метою роботи є розробка комплексної Honeypot-системи на базі Ruby on Rails для реєстрації та аналізу мережевих атак. Завдання включають:

- розробку архітектури і реалізацію веб-інтерфейсу для налаштування пасток та перегляду логів;
- запуск відповідних служб (HTTP, SSH, FTP) для взаємодії з нападниками [4; 6];
- забезпечення зручного адміністрування на основі фреймворку Avo [3];
- організацію управління життєвим циклом пасток за допомогою бібліотеки Actor [4];
- проведення експериментів з реальними атаками та отримання емпіричних результатів (типи атак і поведінки зловмисників).

**Архітектура системи.** Система побудована на основі фреймворку Ruby on Rails, що забезпечує стандартні MVC-підходи і швидку розробку веб-додатків [2]. Rails надає готові структури для роботи з базою даних, веб-сервісами й сторінками, а також підтримує зручні механізми міграцій і генерації основних моделей/контролерів для швидкого запуску проекту. Загальна архітектура включає:

- Веб-модуль (HTTP) - декілька Rails-контролерів і маршрутів, що обробляють всі вхідні HTTP-запити (до фіктивних сторінок/інтерфейсів). Кожен запит (URL, заголовки, тіло)

фіксується у відповідних лог-файлах в базі даних для подальшого аналізу.

- Honeypot служби (SSH, FTP) - система запускає окремі фонові сервіси, які імітують реальні SSH- та FTP-сервери з фіктивними обліковими записами. Усі спроби підключення, введення пароля, виконання команд тощо ретельно записуються.
- База даних і моделі - у сховищі зберігається інформація про події (HTTP-запити, SSH/FTP-сесії), налаштування пасток і метадані. Rails-моделі представляють користувачів, пастки і логи.
- Менеджмент процесів - для керування запуском та зупинкою пасток використано бібліотеку Ruby-гем actor, що дозволяє описувати сервіси як поетапні об'єкти (service objects) з можливістю обробки помилок. Це спрощує послідовне виконання дій (запуск демона, налаштування ) і їх відкат у разі помилок [4].
- Адміністративний інтерфейс (Avo) - поверх Rails реалізовано зручну панель керування з використанням фреймворку Avo. Avo дозволяє декларативно описати ресурси (моделі) і автоматично генерує CRUD-інтерфейс із зручним UX. Це суттєво скорочує час розробки панелі адміністрування: Avo позиціонує себе як «фреймворк адміністрування для Rails, який економить командам місяці розробки» [3].

**Експериментальні результати.** Після розгортання на сервері були проведені тестові атаки та збір реального трафіку. Основні спостереження:

- HTTP: зафіксовано численні сканування веб-додатків (перевірка шляхів /admin, /login, /phpinfo тощо), а також атаки на форми (SQL-ін'єкція у полях логіна, XSS-ключі).
- SSH: домінують brute-force-атаки - автоматизовані ботнети підбирали комбінації для ідентифікації за зразками root:password, admin:admin тощо. У вдалих випадках системи відслідковували запуск сценарію та передачу шкідливих бінарних файлів (інколи різних архітектур).
- FTP: аналогічні шаблони - багато підборів логінів/паролів, спроби завантаження сценаріїв, що є класичною поведінкою ботів із використанням стандартних словників.

**Висновки.** Отримані результати демонструють, що система успішно виконує роль «пастки» для зловмисників. Зібрані дані дозволяють детально аналізувати тактики атак і коригувати захист. В цілому, Honeypot-підхід довів високу ефективність, що обумовлено генеруванням малої кількості хибних спрацювань. Передбачається, що

легітимні користувачі навряд чи навмисно потрапляють у пастку (на відміну від IDS, яка іноді може давати хибні спрацьовування - сигналізувати про небезпеку і у випадках нормального трафіку). Таким чином, презентована Honeypot-система підтвердила практичну користь застосування Ruby on Rails у сфері кібербезпеки. Подальший розвиток може включати додавання штучного інтелекту для автоматичного аналізу зібраних атак, розширення переліку пасток на інші служби (Telnet, IoT-пристрої тощо) та інтеграцію з SIEM-аналізаторами загроз. Такі кроки дозволять перетворити атаку на цінну розвідку і зроблять захист більш проактивним.

**Список використаних джерел:**

1. The Honeynet Project. Know Your Enemy: Honeynets. URL: https://www.honeynet.org (дата звернення: 20.11.2025).

2. Ruby on Rails Guides. Ruby on Rails Documentation. URL: https://guides.rubyonrails.org (дата звернення: 20.11.2025).

3. Avo. Avo Admin Panel for Ruby on Rails. URL: https://avohq.io (дата звернення: 20.11.2025).

4. Sunny. Actor Ruby Gem: Lightweight Actor Model Implementation. URL: https://github.com/sunny/actor (дата звернення: 20.11.2025).

5. Kaur M., Singh S. Analysis of Honeypot Systems for Network Security. International Journal of Computer Applications. 2014. Vol. 96, No. 18. P. 1-4.

6. Gonzalez A., Smith J. Techniques for HTTP Attack Detection Using Application-Layer Honeypots. Journal of Cybersecurity Research. 2020. Vol. 12(3). P. 45-57.

7. Brown L., Patel R. Deployment of Multi-Protocol Honeypots for Attack Profiling. International Journal of Information Security Science. 2019. Vol. 8(2). P. 63-74.