ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

NGUYỄN PHÚC NHI
ĐINH BẠCH KIỀU PHƯƠNG

BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH

PHƯƠNG PHÁP KIỂM THỬ TỰ ĐỘNG LINH HOẠT DỰA TRÊN HỌC TĂNG CƯỜNG VÀ DANH MỤC CHỈ DẪN

DYNAMIC AUTOMATED PENTESTING METHOD BASED ON REINFORCEMENT LEARNING AND GUIDELINES CATALOG

TP. HỒ CHÍ MINH, 2025

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

NGUYỄN PHÚC NHI – 22521041
ĐINH BẠCH KIỀU PHƯƠNG – 21520406

BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH

PHƯƠNG PHÁP KIỂM THỬ TỰ ĐỘNG LINH HOẠT DỰA TRÊN HỌC TĂNG CƯỜNG VÀ DANH MỤC CHỈ DẪN

DYNAMIC AUTOMATED PENTESTING METHOD BASED ON REINFORCEMENT LEARNING AND GUIDELINES CATALOG

GIẢNG VIÊN HƯỚNG DẪN
ThS. NGÔ KHÁNH KHOA

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến thầy Ngô Khánh Khoa đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được đồ án này. Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho đồ án của mình.

Trong quá trình thực hiện đồ án, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong tạo ra được sản phẩm chất lượng nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng thầy sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai.

Nhóm xin chân thành cảm ơn thầy!

TP. Hồ Chí Minh, ngày 30 tháng 06 năm 2025

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ đầy đủ
1	SQLi	SQL Injection
2	RL	Reinforcement Learning
3	DQL	Deep Reinforcement Learning
4	ML	Machine Learning
5	DL	Deep Learning
6	DRL	Deep Q-Learning
7	MDP	Markov Decision Process
8	CVSS	Common Vulnerability Scoring System
9	ANNs	Artificial Neural Networks
10	NLP	Natural Language Processing
11	GAN	Natural Language Generation
12	WAF	Web Application Firewall

TÓM TẮT

Trong bối cảnh tấn công mạng ngày càng tinh vi, đặc biệt là tấn công SQL Injection – một trong những lỗ hổng nguy hiểm phổ biến nhất theo chuẩn OWASP, do đó nhu cầu về các phương pháp kiểm thử xâm nhập tự động hiệu quả ngày càng trở nên cấp thiết. Đồ án này đề xuất một hệ thống kiểm thử linh hoạt dựa trên học tăng cường, cụ thể là hai thuật toán Q-Learning và Deep Q-Learning nhằm tự động hóa quá trình khai thác lỗ hổng SQL Injection trong môi trường giả lập.

Hệ thống được xây dựng dưới dạng một framework mô phỏng, nơi agent học cách gửi các payload SQL độc hại để tương tác với ứng dụng web, phân tích phản hồi và tối ưu chiến lược tấn công thông qua quá trình huấn luyện. Các thuật toán được triển khai và so sánh trong cùng một điều kiện môi trường, với bộ hành động tấn công được xây dựng thủ công và phần thưởng phản ánh mức độ thành công hoặc thất bại của hành động.

Kết quả thực nghiệm cho thấy mô hình Deep Q-Learning vượt trội hơn về hiệu suất, với tỷ lệ khai thác thành công đạt 100% sau 2000 tập huấn luyện, số truy vấn trung bình giảm còn 2.38, và thời gian xử lý mỗi tập được tối ưu rõ rệt. Trong khi đó, mô hình Q-Learning tuy đơn giản hơn nhưng vẫn đạt tỷ lệ thành công gần 99.8% sau 10,000 tập. Các kết quả này cho thấy tiềm năng rõ rệt của việc ứng dụng Deep Reinforcement Learning trong tự động hóa kiểm thử bảo mật, đồng thời mở ra hướng phát triển cho các hệ thống kiểm thử xâm nhập thế hệ tiếp theo.

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu vấn đề

Năm 2025, tình hình an toàn thông tin toàn cầu đang đối mặt với nhiều thách thức phức tạp và đa dạng, trong số đó tấn công SQL Injection tiếp tục là mối đe dọa nghiêm trọng đối với an toàn thông tin toàn cầu.

Trong thực tế, đã có nhiều vụ tấn công nghiêm trọng do khai thác lỗ hổng SQL Injection. Tiêu biểu như vụ tấn công Sony Pictures năm 2011, nhóm hacker LulzSec đã sử dụng SQLi để lấy cắp dữ liệu của hơn 1 triệu người dùng, gây ảnh hưởng lớn đến uy tín của Sony. Tại Anh, vụ tấn công vào TalkTalk năm 2015 khiến 157.000 khách hàng bị rò rỉ thông tin, dẫn đến khoản phạt 400.000 bảng Anh vì sai sót trong bảo mật. Ngoài ra, vụ tấn công một công ty xử lý thanh toán lớn tại Mỹ là Heartland Payment Systems năm 2008 đã khiến hơn 130 triệu thẻ tín dụng bị lộ, gây ra một trong những sự cố bảo mật lớn nhất lịch sử.

Bên cạnh tổ chức thì đối với cá nhân việc mất thông tin cá nhân hay bị xâm phạm quyền riêng tư đang trở thành nỗi lo thường trực trong thời đại 4.0. Điều này không chỉ đe dọa sự an toàn của hệ thống thông tin mà còn đặt ra những thách thức lớn trong việc bảo vệ dữ liệu và tài sản trong kỷ nguyên kỹ thuật số.

Việc áp dụng học máy trong khai thác lỗ hổng SQL Injection đang trở thành một phương pháp hiệu quả để đánh giá bảo mật của các trang web hoặc hệ thống. Bằng cách sử dụng các mô hình học máy, mô hình có thể phân tích dữ liệu đầu vào, xác định mẫu tấn công tiềm ẩn và tự tạo ra các truy vấn độc hại nhằm kiểm tra mức độ dễ bị tấn công của hệ thống. Từ đó giúp phát hiện lỗ hổng nhanh và chính xác hơn so với phương pháp kiểm thử truyền thống. Sau khi xác định được các điểm yếu, doanh nghiệp có thể nhanh chóng triển khai các bản vá bảo mật, cải thiện khả năng phòng thủ trước các cuộc tấn công SQL Injection trong thực tế. Việc kết hợp AI và học máy không chỉ giúp nâng cao an toàn hệ thống mà còn tối ưu hóa quy trình kiểm thử, giảm tải công việc cho các chuyên gia bảo mật trong thời kỳ chuyển đổi số.

1.2 Thách thức và phương pháp

1.2.1 Thách thức

Trong lĩnh vực an toàn thông tin, việc khai thác lỗ hổng SQL Injection dựa trên xây dựng một framework học tăng cường (Reinforcement Learning - RL) kiểm thử xâm nhập tự động đối mặt với khá nhiều thách thức.

Thách thức hàng đầu đến từ việc lựa chọn và tối ưu mô hình học tăng cường. Hệ thống cần tìm ra thuật toán RL phù hợp nhất, ở đây chúng tôi lựa chọn Q-learning và Deep Q-learning để dễ dàng triển khai và so sánh. Quá trình huấn luyện RL đòi hỏi phải đánh giá khả năng khai thác của hệ thống bằng cách thử nghiệm trên các bộ dữ liệu khác nhau. Việc lựa chọn các challenge kiểm thử từ các nền tảng như CTF SQLi, Zixem và các hệ thống khai thác lỗ hổng khác cũng là một thách thức quan trọng nhằm đảm bảo tính khách quan và thực tế trong quá trình đánh giá.

Bên cạnh đó, một vấn đề quan trọng khác là việc xây dựng và cải tiến mô hình mô phỏng môi trường tấn công. Hệ thống cần được thiết kế để có thể dễ dàng tích hợp các thuật toán khác nhau, đồng thời phải tối ưu code để đảm bảo khả năng vận hành trên GPU, góp phần giảm thời gian training và cải thiện hiệu suất mô hình từ đó nâng cao hiệu quả khai thác.

1.2.2 Phương pháp

Để giải quyết các thách thức trên, chúng tôi áp dụng phương pháp xây dựng mô hình kiểm thử linh hoạt kết hợp với học tăng cường và danh mục chỉ dẫn. Cụ thể, framework sẽ được thiết kế dưới dạng một môi trường mô phỏng có khả năng tự động sinh payload dựa trên RL. Hệ thống sẽ tận dụng các kỹ thuật hướng dẫn để định hướng quá trình tìm kiếm payload hiệu quả hơn, với mục tiêu tăng tỷ lệ khai thác thành công.

Bên cạnh đó, các thuật toán RL sẽ được thử nghiệm và so sánh bao gồm Q-learning và Deep Q-learning nhằm tìm ra phương pháp tối ưu cho bài toán khai thác lỗ hổng SQL Injection tự động. Ngoài ra, một bộ tiêu chí đánh giá sẽ được xây dựng để đo lường hiệu suất khai thác trong các điều kiện khác nhau. Các challenge từ các nền tảng như Zixem và CTF SQLi,... sẽ được lựa chọn để kiểm tra hiệu quả của hệ thống. Việc so sánh tỷ lệ khai thác thành công giữa các thuật toán khác nhau giúp đánh giá mức độ học tập của từng mô hình.

1.3 Mục tiêu, nội dung cụ thể

1.3.1 Mục tiêu cụ thể

Mục tiêu chính của đề tài là phát triển một framework kiểm thử xâm nhập tự động có khả năng khai thác lỗ hổng SQL Injection dựa trên học tăng cường. Hệ thống được thiết kế để hoạt động với các thuật toán khác nhau, bao gồm Q-learning và Deep Q-learning.

Ngoài ra, hệ thống sẽ được huấn luyện trên nhiều bộ dữ liệu và môi trường thực tế như CTF SQLi và Zixem,... bên cạnh việc tối ưu hiệu suất bằng cách xây dựng code chạy trên GPU nhằm rút ngắn thời gian training. Từ đó nhóm sẽ đánh giá khả năng thích nghi và tối ưu chiến lược khai thác của hệ thống thông qua quá trình thử nghiệm liên tục dựa trên thời gian huấn luyện và tỉ lệ thành công khi khai thác lỗ hổng SQL Injection khi triển khai ở các thuật toán khác nhau.

1.3.2 Nội dung cụ thể

Để đạt được các mục tiêu đề ra, đề tài sẽ được thực hiện theo các bước chính sau:

- **Nghiên cứu tài liệu và các công trình liên quan:** Giai đoạn đầu tiên tập trung vào việc nghiên cứu các tài liệu khoa học, bài báo và công trình nghiên cứu liên quan đến kiểm thử xâm nhập tự động, học tăng cường (Reinforcement Learning - RL) và các phương pháp khai thác lỗ hổng SQL Injection. Việc này giúp xây dựng nền tảng lý thuyết vững chắc, xác định các phương pháp tiếp cận hiệu quả và hiểu rõ những thách thức trong lĩnh vực này.
- **Phân tích và đánh giá mô hình hiện có:** Sau khi tìm hiểu cơ sở lý thuyết từ các tài liệu và các công trình liên quan, mô hình kiểm thử hiện có sẽ được thiết lập và chạy thử để đánh giá hiệu suất ban đầu. Quá trình này giúp kiểm tra tính ổn định của hệ thống, xác định các điểm hạn chế cần cải tiến và đánh giá mức độ phù hợp của mô hình với các yêu cầu thực tế.
- **Phát triển và tối ưu hóa mô hình khai thác tự động:** Ở bước này, hệ thống sẽ được mở rộng và cải tiến theo nhiều khía cạnh:
    + Triển khai và so sánh các phương pháp học tăng cường, bao gồm Q-Learning (học tăng cường cổ điển) và Deep Q-Learning (học tăng cường sử dụng mạng nơ-ron nhân tạo).

    + Tối ưu hóa hiệu suất tính toán, tập trung vào việc chuyển đổi mô hình để hỗ trợ huấn luyện trên GPU, từ đó giảm thời gian xử lý và tăng hiệu suất tổng thể.

- **Xây dựng bộ thử nghiệm và kịch bản kiểm tra thực tế:** Để đảm bảo tính toàn diện của quá trình đánh giá, cần thu thập và xây dựng bộ kịch bản thử nghiệm từ các nguồn thực tế. Các thử nghiệm sẽ bao gồm các challenge khai thác SQLi từ Zixem, CTF SQLi và các nền tảng khai thác khác. Việc mở rộng kịch bản giúp đánh giá khả năng tổng quát hóa của hệ thống, đảm bảo mô hình có thể hoạt động tốt trên nhiều tình huống khác nhau.
- **Huấn luyện, phân tích hiệu suất và đánh giá kết quả:** Sau khi hoàn thiện mô hình, hệ thống sẽ được huấn luyện và đánh giá dựa trên các tiêu chí sau:
    + Hiệu suất huấn luyện: Đánh giá thời gian training trên CPU và GPU, so sánh mức độ tối ưu giữa các thuật toán RL.
    + So sánh phương pháp Q-Learning và Deep Q-Learning: Phân tích sự khác biệt về hiệu suất khai thác giữa hai thuật toán để tìm ra phương pháp tối ưu nhất.

Kết quả thu được từ các thử nghiệm sẽ được phân tích, tổng hợp và trình bày trong báo cáo, đồng thời đề xuất những hướng phát triển tiếp theo để cải thiện mô hình trong tương lai.

1.4 Các liên quan

Trong quá trình nghiên cứu tài liệu, chúng tôi đã khảo sát nhiều công trình liên quan đến kiểm thử xâm nhập tự động và khai thác lỗ hổng SQL Injection bằng học tăng cường. Dưới đây là tổng hợp một số nghiên cứu với phương pháp tiêu biểu:

1. **Penetration Testing and Attack Automation Simulation: Deep Reinforcement Learning Approach (Jabr et al. [2024])**

   Nghiên cứu này xây dựng một hệ thống kiểm thử xâm nhập tự động bằng Deep Q-Learning, tận dụng các công cụ như Nmap, MulVAL và Docker để xây dựng môi trường tấn công mô phỏng. Mô hình được huấn luyện để tìm đường tấn công tối ưu dựa trên đồ thị tấn công (attack graph) và điểm CVSS.

- Ưu điểm: Khả năng phát hiện nhiều đường tấn công tiềm năng và hiệu quả hơn so với kiểm thử thủ công.
- Nhược điểm: Phụ thuộc vào dữ liệu từ CVSS và giới hạn trong môi trường mô phỏng.

2. **Sqirl: Grey-Box Detection of SQL Injection Vulnerabilities Using Reinforcement Learning (Al Wahaibi et al. [2023])**

   SQIRL sử dụng nhiều agent hoạt động song song để fuzz các đầu vào trên web ứng dụng, sử dụng mô hình RL sâu để sinh ra payload SQLi phù hợp với từng ngữ cảnh.

   - Ưu điểm: Khả năng sinh payload linh hoạt, phát hiện nhiều lỗ hổng hơn với số lượng request ít hơn.
   - Nhược điểm: Cần truy cập nhật ký CSDL và có cấu trúc hệ thống phù hợp để phản hồi trạng thái.

3. **XploitSQL: Advancing Adversarial SQL Injection Attack Generation with Language Models and Reinforcement Learning (Leung et al. [2024])**

   Bài báo sử dụng mô hình ngôn ngữ lớn (T5) kết hợp với Reinforcement Learning (Actor-Critic) để sinh truy vấn SQL Injection có khả năng né tránh hệ thống phát hiện như WAF và các mô hình học máy.

   - Ưu điểm: Hiệu quả cao trong việc vượt qua WAF, tạo payload mới mẻ và phù hợp ngữ cảnh.
   - Nhược điểm: Yêu cầu tài nguyên huấn luyện cao và thiết kế cẩn thận hàm phần thưởng.

4. **PenGym: Realistic training environment for reinforcement learning pentesting agents (Nguyen et al. [2025])**

   PenGym cung cấp một môi trường thực tế thay vì mô phỏng để huấn luyện tác tử học tăng cường, có khả năng thực hiện các hành động tấn công thật trên mạng ảo được thiết lập tự động.

   - Ưu điểm: Khắc phục khoảng cách thực tế – mô phỏng, tăng độ chính xác và khả năng tổng quát của tác tử.
   - Nhược điểm: Cài đặt phức tạp, yêu cầu hạ tầng mạng để chạy môi trường.

5. **Extending Q-Learning Agents in SQLi Environments (Marinelli et al. [2024])**

   Nghiên cứu này đánh giá các biến thể của Q-Learning như Double Q-Learning, N-Step Q-Learning và Upper Confidence Bound để cải thiện tác tử tấn công SQLi.

   - Ưu điểm: Tăng hiệu năng huấn luyện, giảm overfitting và cải thiện khả năng khám phá.
   - Nhược điểm: Chưa khai thác triệt để tính năng tổng quát hóa như trong Deep Q-Learning.

6. **Developing a SQL Injection Exploitation Tool with Natural Language Generation (Boekweg [2024])**

   Sử dụng mô hình Natural Language Generation (GAN) để tự động sinh payload SQL Injection và khai thác lỗ hổng trên các trang web mục tiêu.

   - Ưu điểm: Tăng tính đa dạng của payload, khai thác được nhiều lỗ hổng hơn so với các công cụ như SQLMap, dễ tích hợp học máy.
   - Nhược điểm: Payload còn trùng lặp với dữ liệu cũ, tốc độ xử lý chậm, phụ thuộc nhiều vào chất lượng dữ liệu đầu vào.

7. **Simulating SQL Injection Vulnerability Exploitation Using Q-Learning Reinforcement Learning Agents (Erdődi et al. [2021])**

   Mô hình hóa bài toán khai thác SQLi như một quá trình Markov và giải quyết bằng Q-Learning. Hệ thống tập trung vào môi trường đơn giản với chỉ một tham số đầu vào dễ bị tổn thương.

   - Ưu điểm: Thiết lập mô hình học tăng cường rõ ràng và có thể tái sử dụng, chứng minh được hiệu quả của Q-Learning trong môi trường giả lập.
   - Nhược điểm: Giả định đơn giản hoá cao như chỉ có một điểm đầu vào dễ bị tấn công, không có sự hiện diện của hệ thống phòng thủ, chỉ tập trung vào kiểu tấn công UNION-based, chưa bao phủ các dạng khai thác khác.

8. **Simulating All Archetypes of SQL Injection Vulnerability Exploitation Using Reinforcement Learning Agents (Sommervoll et al. [2024])**

Nghiên cứu này mở rộng mô hình từ nghiên cứu trước, khai thác nhiều kiểu SQLi như boolean-based, error-based, time-based,… đồng thời tích hợp khả năng học chuyển giao.

- Ưu điểm: Mô hình toàn diện bao phủ hầu hết các kiểu khai thác SQLi phổ biến, khả năng phân tích phản hồi từ máy chủ (giảm tiền xử lý), học chuyển giao hiệu quả giữa các dạng tấn công.
- Nhược điểm: Yêu cầu môi trường huấn luyện phức tạp và tài nguyên tính toán lớn, một số dạng khai thác phức tạp như time-based vẫn cho kết quả hạn chế do tính không xác định cao.

9. **Detection of SQL Injection Attacks: A Machine Learning Approach (Hasan et al. [2019])**

Nghiên cứu này sử dụng 23 bộ phân loại máy học khác nhau để phát hiện SQLi trên cơ sở dữ liệu tĩnh, tập trung vào phát hiện thay vì khai thác.

- Ưu điểm: Tỷ lệ phát hiện cao (trên 93.8%) với tập dữ liệu được chuẩn hóa, phù hợp cho các hệ thống phòng thủ.
- Nhược điểm: Không áp dụng cho việc khai thác mà chỉ dùng để phát hiện, không sử dụng mô hình học tăng cường do đó thiếu khả năng tương tác và thích ứng với môi trường.

Dựa trên nền tảng lý thuyết và phân tích các công trình liên quan, chúng tôi xây dựng một hệ thống học tăng cường nhằm huấn luyện một agent có khả năng tự động khai thác lỗ hổng SQL Injection trên một môi trường giả lập. Hệ thống được thiết kế với các đặc điểm sau:

- Môi trường thử nghiệm mô phỏng một ứng dụng web với lỗ hổng SQLi.
- Agent học thông qua tương tác như gửi các chuỗi payload, phân tích phản hồi từ máy chủ và tối ưu hóa chiến lược tấn công.

Hai thuật toán được sử dụng là:

- Q-Learning với Q-Table phù hợp với không gian trạng thái nhỏ, dễ kiểm soát.
- Deep Q-Learning sử dụng mạng nơ-ron sâu để xử lý không gian trạng thái phức tạp hơn.

Mục tiêu của agent là khai thác thành công để thu thập thông tin nhạy cảm như username và database version,...

Kết quả thử nghiệm bước đầu cho thấy agent có khả năng học được chiến lược khai thác hiệu quả đối với một số dạng SQLi cơ bản và tiềm năng mở rộng cho các dạng khai thác phức tạp hơn.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 SQL injection

Lỗ hổng SQL injection cho phép kẻ tấn công gắn mã độc hại (thường là các câu lệnh SQL) làm sai lệch câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ cơ sở dữ liệu. Thường xảy ra khi một ứng dụng web sử dụng các truy vấn SQL động mà không thực hiện đầy đủ các biện pháp kiểm tra và lọc dữ liệu đầu vào thích hợp thì mã độc hại sẽ được thực thi trực tiếp trên hệ thống cơ sở dữ liệu dẫn đến các cuộc tấn công SQL injection.

Tấn công SQL injection (SQLi) là một kỹ thuật tấn công mạng nguy hiểm, kẻ tấn công có thể chèn mã SQL độc hại vào các điểm nhập dữ liệu như biểu mẫu, URL hoặc tham số HTTP,... nhằm khai thác lỗ hổng bảo mật trong các ứng dụng web.

2.1.1 Tấn công Error-Based SQLi

Liên quan đến việc gửi các truy vấn SQL độc hại nhằm kích hoạt lỗi hoặc xác nhận lỗ hổng trong ứng dụng. Kẻ tấn công sử dụng những lỗi này để thu thập thông tin về cấu trúc cơ sở dữ liệu hoặc các chi tiết nhạy cảm khác.

Ví dụ tấn công Error-Based SQLi:

- Kẻ tấn công có thể sử dụng các lệnh SQL như dấu nháy đơn, dấu nháy kép hoặc các toán tử như AND, OR và NOT để khai thác lỗi.
- Khi nhập http://example.vn/index.php?title=1' có thể tạo ra một thông báo lỗi như: “You have an error in your SQL syntax; check the manual corresponding to your MySQL server version for the right syntax to use near ‘‘VALUE’’.
- Thông báo lỗi cung cấp cho kẻ tấn công các thông tin quan trọng như:
    + Cơ sở dữ liệu sử dụng là MySQL.
    + Lỗi trong cú pháp là dấu nháy kép.
    + Vị trí xảy ra lỗi nằm ở cuối tham số.

2.1.2 Tấn công Union-Based SQLi

Trong loại tấn công tiêm SQL này, kẻ tấn công lợi dụng lỗ hổng bằng cách sử dụng toán tử "UNION". Toán tử UNION được dùng để kết hợp hai bảng hoặc thực hiện đồng

thời hai truy vấn chọn lọc. Trong toán tử UNION, các hàng hoặc cột trùng lặp được loại bỏ, đây là điều mà kẻ tấn công cố gắng tận dụng.

Ví dụ tấn công Union-Based SQLi

- Giả sử một ứng dụng web xây dựng một truy vấn SQL như sau:

SELECT name, email, phone FROM users WHERE name = '[user_input]'

- Dữ liệu đầu vào của người dùng không được lọc đúng cách, vì vậy kẻ tấn công có thể chèn thêm mã SQL độc hại. Ví dụ, chúng có thể nhập giá trị sau làm tên:

' UNION SELECT password, NULL, NULL FROM users --

- Điều này sẽ dẫn đến việc thực thi truy vấn SQL sau:

SELECT name, email, phone FROM users WHERE name = '' UNION SELECT password, NULL, NULL FROM users --'

- Ký tự '--' ở cuối chuỗi được chèn là một ký tự comment (bình luận), nó comment phần còn lại của truy vấn gốc. Vì vậy, truy vấn kết quả tương đương với:

SELECT name, email, phone FROM users WHERE name = ''

UNION SELECT password, NULL, NULL FROM users

- Truy vấn này sẽ trả về một bảng chứa tên, email và số điện thoại của người dùng và một bảng với tất cả mật khẩu trong bảng người dùng. Kẻ tấn công sau đó có thể sử dụng thông tin này để tiếp tục xâm nhập vào hệ thống.

2.1.3 Tấn công Blind SQLi - Boolean-Based SQLi

Tấn công Blind SQLi xảy ra khi kẻ tấn công không thể trực tiếp xem nội dung cơ sở dữ liệu nhưng có thể suy đoán thông tin dựa trên phản hồi của ứng dụng. Tấn công Blind SQLi bao gồm hai loại chính: Boolean-Based SQLi và Time-Based SQLi.

Trong kiểu khai thác lỗ hổng SQL injection Boolean-Based SQLi, kẻ tấn công gửi một loạt các truy vấn SQL đánh giá kết quả đúng hoặc sai, tùy thuộc vào việc mã được chèn có được thực thi thành công hay không. Sau đó, kẻ tấn công có thể sử dụng phản hồi của ứng dụng để suy đoán thông tin về cơ sở dữ liệu bằng cách xây dựng các truy vấn phức tạp nhằm thăm dò thông tin cụ thể.

Ví dụ tấn công Boolean-Based SQLi:

- Một truy vấn cơ sở dữ liệu SQL phổ biến của một cửa hàng trực tuyến có thể như thế này:

SELECT ItemName, ItemDescription FROM Item WHERE ItemNumber = ItemNumber

- Vì vậy, URL sản phẩm trên cửa hàng trực tuyến có thể là: http://www.example.vn/items/items.asp?itemid=999 or 1=1.
- Truy vấn SQL có thể là:

SELECT ItemName, ItemDescription FROM Items WHERE ItemNumber = 999 OR 1=1

- Do truy vấn với điều kiện "1=1" luôn đúng, các truy vấn SQL sẽ trả về tất cả tên và mô tả sản phẩm trong cơ sở dữ liệu, ngay cả những sản phẩm mà kẻ tấn công không có quyền truy cập.

2.1.4 Tấn công Blind SQLi - Time-Based SQLi

Kỹ thuật tấn công SQL injection này liên quan đến việc chèn một truy vấn để gây ra việc ứng dụng phản hồi chậm trễ. Bằng cách đo thời gian phản hồi của hệ thống, kẻ tấn công có thể suy ra được dữ liệu có tồn tại không và cấu trúc cơ sở dữ liệu.

Ví dụ tấn công Time-Based SQLi:

- Giả sử có một biểu mẫu đăng nhập trên ứng dụng web sử dụng truy vấn SQL để kiểm tra xem thông tin đăng nhập của người dùng có hợp lệ hay không. Truy vấn có thể trông giống như thế này:

*SELECT * FROM users WHERE username = 'admin' AND password = 'password123'*

- Để thực hiện tấn công Time-based SQLi, kẻ tấn công có thể chèn một truy vấn như thế này:

SELECT CASE WHEN (1=1) THEN pg_sleep(10) ELSE pg_sleep(0) END;

- Truy vấn này sẽ khiến ứng dụng ngủ trong 10 giây nếu điều kiện (1=1) là đúng. Kẻ tấn công có thể xác định điều kiện là đúng hay sai bằng cách đo thời gian ứng dụng phản hồi cho truy vấn này.

- Nếu phản hồi mất 10 giây, kẻ tấn công biết điều kiện là đúng và ứng dụng dễ bị tấn công Time-based SQLi. Ngược lại, nếu phản hồi ngay lập tức, kẻ tấn công biết điều kiện là sai.
- Một khi kẻ tấn công xác nhận được khả năng tấn công Time-based SQLi, chúng có thể bắt đầu chèn các truy vấn phức tạp hơn để trích xuất thông tin nhạy cảm từ cơ sở dữ liệu.

2.2 Reinforcement learning

Reinforcement learning - RL hay học tăng cường là một nhánh của học máy (Machine learning - ML), RL nghiên cứu cách thức một tác nhân (Agent) trong một môi trường (Environment) đang ở một trạng thái (State) thực hiện một hành động (Action) để tối ưu hóa một phần thưởng (Reward) nhận được theo thời gian bằng cách chọn những hành động tốt nhất trong từng tình huống.

Môi trường thường được biểu diễn dưới dạng quá trình quyết định Markov (Markov decision process - MDP). MDP được coi là tiêu chuẩn để mô hình hóa các bài toán học tăng cường. Ngoài ra các thuật toán của học tăng cường cũng liên quan nhiều đến kỹ thuật quy hoạch động.

2.2.1 Các khái niệm chính

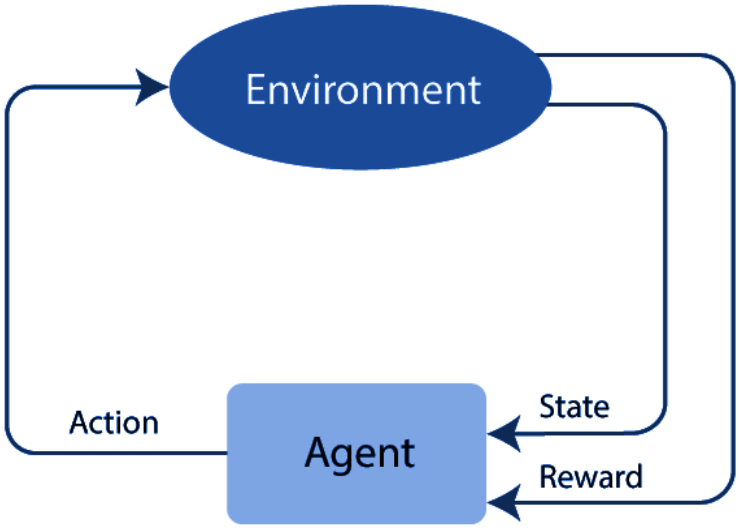
- Tác nhân (Agent): Thực thể tương tác với môi trường và đưa ra quyết định.
- Môi trường (Environment): Hệ thống bên ngoài hoặc thế giới mà tác nhân hoạt động trong đó. Môi trường cung cấp phản hồi dựa trên các hành động của tác nhân.
- Hành động (Action – A): Tập các hành động của tác nhân.
- Trạng thái (State – S): Tình trạng hiện tại của tác nhân trong môi trường.
- Phần thưởng (Reward – R): Đối với mỗi hành động được chọn bởi tác nhân, môi trường sẽ đưa ra một phần thưởng. Phần thưởng có giá trị dương, âm hoặc bằng không. Tác nhân hướng đến việc tối đa hóa phần thưởng này.
- Chính sách (Policy – π): Chiến lược (ra quyết định) mà tác nhân sử dụng để phản ứng trước môi trường giúp đạt được mục tiêu là tối đa hóa phần thưởng.
- Hàm giá trị (Value Function): Hàm ước tính phần thưởng tích lũy dự kiến từ một trạng thái nhất định, giúp tác nhân dự đoán giá trị dài hạn của các hành động.

2.2.2 Cách thức hoạt động

Trong học tăng cường, các nhà phát triển nghĩ ra một phương pháp khen thưởng các hành vi mong muốn và trừng phạt các hành vi tiêu cực. Phương pháp này gán các giá trị dương cho các hành động mong muốn để khuyến khích tác nhân và các giá trị âm cho các hành vi không mong muốn. Chương trình này giúp tác nhân tìm kiếm phần thưởng tổng thể dài hạn và tối đa để đạt được một giải pháp tối ưu.

Theo thời gian, tác nhân học cách tránh điều tiêu cực và tìm kiếm điều tích cực. Phương pháp học này đã được áp dụng trong trí tuệ nhân tạo (AI) như một cách chỉ đạo việc học máy không giám sát thông qua phần thưởng và hình phạt.

Tác nhân thực hiện hành động (A) trong trạng thái (S) nhất định của môi trường. Môi trường phản hồi bằng phần thưởng (R) và chuyển sang trạng thái mới (S'). Sau đó tác nhân sử dụng phản hồi này để cập nhật chiến lược (π) của mình, dần dần cải thiện khả năng ra quyết định bằng cách tối đa hóa phần thưởng đạt được cuối cùng.



Hình 2.1: Mô hình hoạt động của Reinforcement learning

2.2.3 Phân loại

Gồm hai loại chính là Model-based reinforcement learning và Model-free reinforcement learning. Model-based RL phù hợp với môi trường ổn định và có cấu trúc rõ ràng trong khi Model-free RL hữu ích cho môi trường phức tạp, biến đổi và không thể dự đoán chính xác:

- Reinforcement learning dựa trên mô hình (Model-based RL): Model-based RL được sử dụng khi môi trường được xác định rõ ràng, không thay đổi và thử nghiệm thực tế gặp khó khăn. Trong loại này, tác nhân (agent) xây dựng một mô hình nội bộ của môi trường. Các bước thực hiện bao gồm tác nhân thực hiện các hành động và ghi nhận trạng thái mới cùng giá trị phần thưởng và tác nhân liên kết chuyển đổi hành động-trạng thái với giá trị phần thưởng. Sau khi mô hình hoàn chỉnh, tác nhân mô phỏng các chuỗi hành động dựa trên xác suất đạt phần thưởng tích lũy tối ưu, từ đó phát triển các chiến lược khác nhau để đạt mục tiêu cuối cùng.
- Reinforcement learning không dựa trên mô hình (Model-free RL): Model-free RL thích hợp cho các môi trường lớn, phức tạp và khó mô tả, hoặc khi môi trường không được biết trước và thay đổi liên tục. Trong loại này, tác nhân không xây dựng mô hình môi trường mà sử dụng phương pháp thử và sai. Nó ghi nhận các cặp trạng thái và hành động từ đó phát triển chính sách từ các phản hồi nhận được.

2.2.4 Q-learning

Q-learning là một thuật toán học tăng cường (Reinforcement Learning - RL) không dựa trên mô hình (model-free), được sử dụng để học một chiến lược tối ưu cho một tác nhân trong một môi trường không biết trước đó thông qua bảng Q-table.

- Q-learning dựa vào công thức cập nhật Q-value (giá trị hành động trong một trạng thái nhất định):

$$Q(s, \alpha) = r(s, \alpha) + \gamma \, max_{\alpha} \, Q(s', a)$$

Trong đó:

$Q(s, \alpha)$: Giá trị Q-value khi thực hiện hành động α tại trạng thái s.

$r(s, \alpha)$: Phần thưởng (reward) nhận được sau khi thực hiện hành động α tại trạng thái s.

s': Trạng thái kế tiếp sau khi thực hiện hành động α.

γ: Hệ số chiết khấu, giúp điều chỉnh mức độ ảnh hưởng của các phần thưởng trong tương lai. Giá trị $0 \leq \gamma \leq 1$. Khi γ tiến tới 0, thuật toán chỉ quan tâm đến phần thưởng ngay lập tức, còn khi γ tiến tới 1, thuật toán ưu tiên các phần thưởng dài hạn.

$max_a Q(s', \alpha)$: Giá trị Q-value lớn nhất có thể đạt được từ trạng thái kế tiếp s'.

Công thức này thể hiện giá trị Q-value của hành động α tại trang thái s bằng tổng của phần thưởng trực tiếp nhận được khi thực hiện hành động α và giá trị kỳ vọng cao nhất có thể đạt được từ trạng thái kế tiếp s', sau khi thực hiện hành động tốt nhất.

Q-learning sử dụng công thức trên để xây dựng Q-table, trong đó mỗi ô chứa Q-value của một cặp trạng thái và hành động cụ thể. Khi agent cần quyết định hành động, nó chỉ cần chọn hành động có Q-value cao nhất tại trạng thái hiện tại.

Tuy nhiên, Q-learning là một quá trình ngẫu nhiên, nên Q-value sẽ thay đổi theo thời gian khi agent khám phá và học hỏi thêm từ môi trường. Sự khác biệt giữa giá trị Q trước và sau khi cập nhật gọi là Temporal Difference (TD).

- Công thức tính Temporal Difference (TD):

$$TD_t(s, a) = R(s, a) + \gamma max_{a'} Q(s', a') - Q_{t-1}(s, a)$$

Trong đó:

$R(s, \alpha)$**:** Phần thưởng nhận được khi thực hiện hành động $\boldsymbol{a}$ tại trạng thái **s**.

γ: Hệ số chiết khấu, điều chỉnh mức độ ảnh hưởng của phần thưởng tương lai.

maxa′Q(s′,*a′*): Giá trị Q tối đa có thể đạt được từ trạng thái tiếp theo **s′**.

Qt−1(s,*a*): Giá trị Q trước khi cập nhật.

- Cập nhật giá trị Q dựa trên Temporal Difference:

$$Q_t(s, a) = Q_{t-1}(s, a) + \alpha TD_t(s, a)$$

Trong đó:

α: Hệ số học, quyết định mức độ điều chỉnh giá trị Q dựa trên TD.

Qua các lần thực hiện hành động, cập nhật lặp lại dựa trên Temporal Difference (TD) giá trị Q sẽ tiến gần hơn đến giá trị mong đợi, dẫn đến sự hội tụ dần về giá trị tối ưu.

2.3 Deep Reinforcement learning

Deep Reinforcement learning - DRL hay học tăng cường sâu là một lĩnh vực con của học máy, kết hợp giữa học tăng cường và học sâu (Deep learning - DL).

DRL là một kỹ thuật học tổng quát, trong đó agent nhận thông tin trạng thái từ môi trường, lựa chọn hành động phù hợp theo chiến lược của mình, thay đổi trạng thái môi trường và nhận phần thưởng. Phần thưởng này xác định hiệu quả của hành động tác tử dựa trên trạng thái mới của môi trường. Thuật toán DRL có khả năng xử lý lượng đầu vào rất lớn và quyết định các bước đi phù hợp để đạt được mục tiêu. Khi DRL được áp dụng vào kiểm thử xâm nhập thì agent đóng vai trò là người kiểm thử và lựa chọn con đường tối ưu nhất để tối đa hóa phần thưởng.

DRL bao gồm ba nhóm thuật toán chính:

- Các phương pháp dựa trên giá trị (Value-based)
- Các phương pháp tìm kiếm dựa trên chiến lược (Policy-based)
- Các phương pháp dựa trên mô hình (Model-based)

2.3.1 Deep learning

Học sâu (Deep learning - DL) xuất phát từ một phương pháp học máy gọi là perceptron hoặc multilayer perceptron. Hiện nay, học sâu đã trở thành một lĩnh vực phổ biến và được công nhận rộng rãi trong ML, thu hút sự chú ý với những thành tựu xuất sắc trong nhiều ứng dụng, như thị giác máy tính, xử lý ngôn ngữ tự nhiên (natural language processing - NLP) và học tăng cường. Học sâu không chỉ giới hạn ở mạng nơ-ron nhân tạo tiêu chuẩn mà còn có tác động mạnh mẽ đến các mô hình mạng nơ-ron và perceptron được sử dụng trong học máy.

Học sâu dựa trên kiến trúc của mạng nơron nhân tạo (Artificial Neural Networks - ANNs). ANNs sử dụng các node được kết nối với nhau gọi là nơron, cùng nhau hoạt động để xử lý và học hỏi từ dữ liệu đầu vào.

Trong một mạng nơron được kết nối đầy đủ, có một layer đầu vào và một hoặc nhiều layer ẩn được kết nối theo thứ tự. Mỗi nơron trong các layer này nhận đầu vào từ nơron của layer trước hoặc trực tiếp từ layer đầu vào. Đầu ra của một nơron trở thành đầu vào cho các nơron trong layer tiếp theo, duy trì quá trình này cho đến khi layer cuối cùng tạo ra đầu ra của mạng. Qua một chuỗi các biến đổi phi tuyến tính, các layer mạng nơron tạo điều kiện thuận lợi cho việc học tập cách biểu diễn phức tạp của dữ liệu đầu vào.

2.3.2 Deep Q-learning

Deep Q-Learning (DQL) là phiên bản mở rộng của Q-learning sử dụng mạng nơ-ron nhân tạo (Neural Network) thay vì Q-table để ước lượng giá trị hành động (Q-value). Điều này giúp DQL xử lý các môi trường có không gian trạng thái lớn hoặc liên tục khi mà Q-table truyền thống không khả dụng.

DQL sử dụng một mạng nơ-ron sâu để xấp xỉ hàm giá trị hành động Q(s,a) với đầu vào là trạng thái s của môi trường và đầu ra là giá trị Q cho từng hành động khả thi a. Quá trình huấn luyện mạng nơ-ron được thực hiện dựa trên công thức cập nhật Q-learning truyền thống, trong đó giá trị target được tính bằng:

$$y = r + \gamma \bullet max_{a'} Q(s', a', \theta^{-})$$

Trong đó:

r: Phần thưởng nhận được sau khi thực hiện hành động α tại trạng thái s.

s′: Trạng thái tiếp theo.

γ: Hệ số chiết khấu tương lai.

θ^{-} : Tham số của mạng nơ-ron mục tiêu, được cập nhật định kỳ từ mạng nơ-ron chính θ.

Để cải thiện tính ổn định và hiệu quả trong quá trình huấn luyện, DQL sử dụng hai kỹ thuật chính:

- Replay Buffer (Experience Replay): Thay vì cập nhật mạng nơ-ron trực tiếp từ các bước gần nhất, (s,a,r,s′) được lưu vào một bộ nhớ và chọn ngẫu nhiên để huấn luyện. Việc này giúp giảm tính tương quan giữa các mẫu liên tiếp và tăng hiệu quả học.

- Target Network: Một bản sao của mạng nơ-ron chính được dùng để tính giá trị mục tiêu. Mạng này được cập nhật định kỳ để tránh tình trạng mục tiêu thay đổi liên tục làm mất ổn định quá trình học.

Nhờ việc kết hợp giữa Q-learning và Deep Learning, DQL có khả năng giải quyết các bài toán phức tạp với không gian trạng thái lớn như chơi game Atari, điều khiển robot, hoặc các bài toán tối ưu trong thực tế. DQL được xem là một trong những bước đột phá quan trọng trong lĩnh vực học tăng cường sâu.

2.4 Danh mục chỉ dẫn

Trong lĩnh vực kiểm thử phần mềm, đặc biệt là kiểm thử bảo mật tự động, danh mục chỉ dẫn (Guidelines Catalog) là tập hợp có cấu trúc các hướng dẫn, quy tắc, chiến lược hoặc mẫu hành vi được thiết kế nhằm hỗ trợ quá trình kiểm thử một cách hệ thống và hiệu quả hơn. Các danh mục này không chỉ đóng vai trò là cơ sở tri thức, mà còn có thể được dùng để dẫn dắt hành vi của agent học tăng cường trong môi trường kiểm thử tự động.

2.4.1 Khái niệm

Danh mục chỉ dẫn được hiểu là một bộ sưu tập các chỉ dẫn có thể định dạng dưới dạng luật, biểu mẫu hành động, pattern tấn công, hoặc nguyên tắc phân tích hệ thống. Mỗi chỉ dẫn trong danh mục thường bao gồm:

- Mục tiêu kiểm thử
- Mô tả hành vi/hành động cụ thể
- Điều kiện áp dụng
- Kết quả kỳ vọng hoặc hậu quả tiềm ẩn
- Mức độ ưu tiên hoặc trọng số

Danh mục chỉ dẫn có thể được xây dựng từ nhiều nguồn khác nhau, nhằm đảm bảo tính toàn diện và sát với thực tế kiểm thử. Cụ thể, các chỉ dẫn có thể được tạo ra dựa trên kinh nghiệm của chuyên gia kiểm thử bảo mật, từ các mẫu tấn công phổ biến như OWASP Top 10, kết quả thống kê được thu thập qua các chiến dịch kiểm thử trước đó cũng như dữ liệu phân tích từ các công cụ quét tự động hoặc hệ thống honeypot. Trong hệ thống đề xuất, các chỉ dẫn này có thể được biểu diễn dưới dạng các chính sách dựa

trên rule-based policy hoặc graph-based structure và được tích hợp trực tiếp vào tiến trình học tăng cường với ba vai trò chính:

- Heuristics định hướng việc chọn hành động.
- Constraint functions giới hạn không gian hành động.
- Reward shaping mechanism điều chỉnh phần thưởng nhằm phản ánh mức độ phù hợp với guideline.

2.4.2 Vai trò trong kiểm thử tự động dựa trên học tăng cường

Danh mục chỉ dẫn được tích hợp vào hệ thống nhằm: Trong phương pháp kiểm thử dựa trên học tăng cường, agent học cách tương tác với hệ thống mục tiêu thông qua quá trình thử – sai nhằm tối đa hóa phần thưởng nhận được. Tuy nhiên, nếu không có sự định hướng phù hợp, quá trình này có thể tiêu tốn đáng kể tài nguyên tính toán và thời gian huấn luyện. Việc tích hợp danh mục chỉ dẫn vào hệ thống đóng vai trò quan trọng trong việc khắc phục vấn đề này.

Cụ thể, danh mục chỉ dẫn giúp hướng dẫn quá trình học của agent bằng cách cung cấp các hành động khởi đầu hợp lý hoặc chiến lược tấn công đã được xác minh từ trước, từ đó rút ngắn thời gian hội tụ vì agent không cần khám phá toàn bộ không gian trạng thái một cách ngẫu nhiên. Đồng thời, danh mục cũng góp phần đảm bảo tính an toàn và hợp lệ của các hành động kiểm thử, giúp hạn chế những tác động tiêu cực có thể xảy ra với hệ thống mục tiêu. Ngoài ra, việc xây dựng chỉ dẫn từ kinh nghiệm thực tiễn và tri thức chuyên gia còn giúp tăng khả năng tổng quát hóa của hệ thống, nâng cao hiệu quả và độ tin cậy trong các tình huống kiểm thử đa dạng.

CHƯƠNG 3: PHƯƠNG PHÁP

3.1 Mô hình tổng quát

Trong nghiên cứu này, chúng tôi xây dựng một hệ thống học tăng cường để huấn luyện một agent tự động khai thác lỗ hổng SQL injection trên một trang web thử nghiệm. Mục tiêu của agent là tương tác với ứng dụng web, thông qua việc gửi các chuỗi payload (action), phân tích phản hồi từ máy chủ (state), và tối ưu hóa chiến lược tấn công để thu được thông tin nhạy cảm (cụ thể là username và database version). Mô hình học tăng cường được triển khai với hai thuật toán: Q-Learning (mô hình bảng Q-Table) và Deep Q-Learning (mô hình mạng nơ-ron sâu).

Chúng tôi tiến hành thiết lập môi trường thử nghiệm, định nghĩa các thành phần trong bài toán học tăng cường, sau đó triển khai hai thuật toán này trên cùng một cấu hình để so sánh hiệu quả.

3.1.1 Thành phần của môi trường và mô hình học máy

Bao gồm một trang web thử nghiệm và hai mô hình học máy với hai thuật toán khác nhau.

Trang web thử nghiệm được thiết lập với một trường nhập liệu dễ bị khai thác bằng kỹ thuật SQL injection. Trang web này phản hồi theo các kịch bản cụ thể tương ứng với các loại lỗi cú pháp, phản hồi không có dữ liệu, phản hồi có chứa username hoặc database version.

Mô hình học tăng cường bao gồm các thành phần:

- Agent: là thành phần trung tâm của hệ thống học tăng cường. Nó đóng vai trò là tác nhân ra quyết định, chọn các chuỗi payload từ không gian hành động để gửi đến trang web. Sau mỗi hành động, agent nhận được một phản hồi, rút ra trạng thái tiếp theo và cập nhật giá trị kỳ vọng của các hành động.
- Action Space: Tổng cộng có 91 hành động, mỗi hành động tương ứng với một chuỗi payload tấn công SQL được viết trước được xây dựng thủ công, chủ yếu là dạng tấn công union-based.
- State: Phản hồi từ máy chủ web được chuẩn hóa thành 5 loại trạng thái rời rạc:
    + syntax: Lỗi cú pháp SQL
    + different_collumn: lỗi không khớp số cột

+ waf_block: bị chặn bởi WAF
+ get_username: truy xuất thành công tên người dùng
+ get_version: truy xuất thành công phiên bản CSDL

3.1.2 Hàm phần thưởng (Reward function)

Môi trường mô phỏng một trang web có phản hồi thực tế khi bị khai thác SQL Injection. Phần thưởng được quy định như sau (có thể điều chỉnh linh hoạt):

Phản hồi	Phần thưởng
Truy xuất username	+10
Truy xuất phiên bản CSDL	+5
Lỗi cú pháp SQL	-2
Số lượng cột sai	-1
Không nhận được phản hồi	-3
Bị WAF chặn	-20

Bảng 3.1: Quy định phần thưởng tương ứng với từng loại phản hồi

Mỗi hành động được chuyển đổi thành chuỗi truy vấn SQL tương ứng, gửi lên trang web thông qua requests.get(), và nội dung HTML phản hồi được phân tích bằng BeautifulSoup để xác định loại phản ứng nhận được.

3.1.2 Chính sách lựa chọn hành động

Agent áp dụng chính sách ε-greedy với ε là hệ số thăm dò (exploration rate). để cân bằng giữa khám phá và khai thác. Trong mỗi lượt chọn hành động, agent tạo một số ngẫu nhiên $r \in [0,1]$:

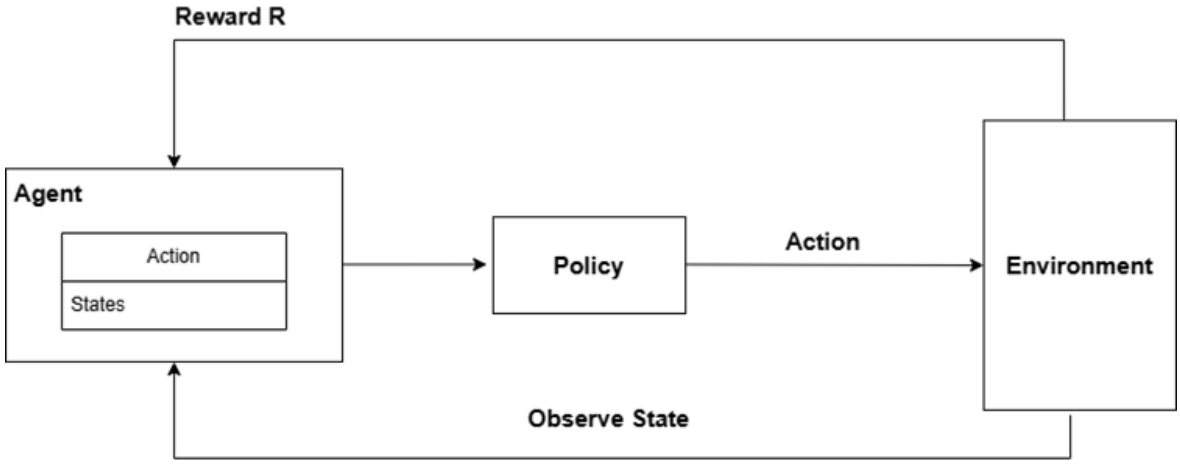
- Nếu $r < \varepsilon$: chọn ngẫu nhiên một hành động.
- Nếu $r \geq \varepsilon$: chọn hành động có Q-value cao nhất theo Q-table hoặc mạng nơ-ron.

Hệ số ε được giảm dần theo từng tập huấn luyện (episode) để tăng dần tính khai thác.

3.2 Phương pháp học

Trong thử nghiệm này, chúng tôi sử dụng hai thuật toán.

3.2.1 Mô hình sử dụng thuật toán Q-Learning



Hình 3.1: Mô hình triển khai sử dụng Q-Learning

Với Q-Learning, không gian trạng thái và hành động được ánh xạ vào một bảng Q có kích thước cố định. Q-table được khởi tạo với giá trị 1 và cập nhật sau mỗi hành động bằng công thức:

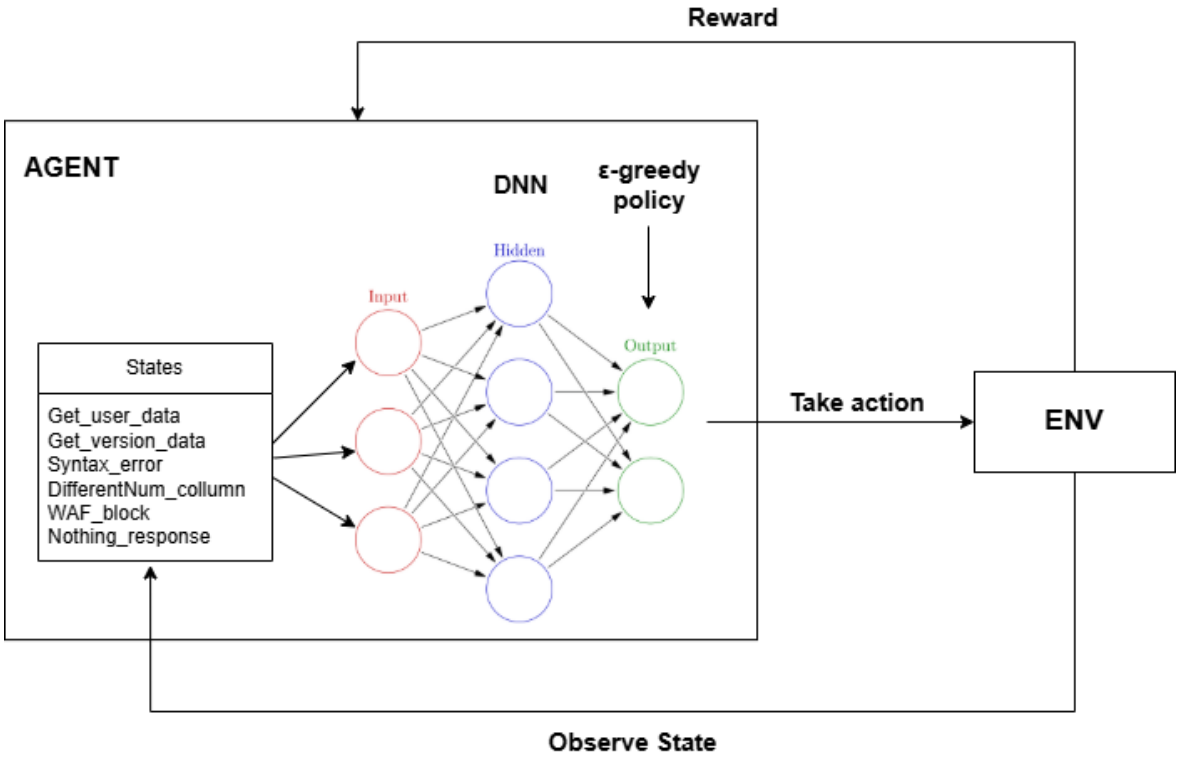
$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$$

Trong đó:

+ s là trạng thái hiện tại
+ a là hành động được thực hiện
+ r là phần thưởng nhận được
+ s' là trạng thái tiếp theo
+ α là tốc độ học (learning rate)
+ γ là hệ số chiết khấu (discount factor)

Trong quá trình huấn luyện, agent được chạy trong nhiều episode. Mỗi episode tương ứng với một phiên tấn công giả lập, bắt đầu từ trạng thái ban đầu và kết thúc khi đạt được mục tiêu hoặc bị WAF chặn liên tiếp. Q-table được cập nhật liên tục và lưu lại sau mỗi tập huấn luyện.

3.2.2 Mô hình sử dụng thuật toán Deep Q-Learning



Hình 3.2: Mô hình triển khai sử dụng Deep Q-Learning

Mạng DQN được thiết kế với ba lớp tuyến tính:

- Lớp vào có kích thước bằng số states có thể xảy ra
- Hai lớp ẩn gồm 128 neuron sử dụng hàm kích hoạt ReLU.
- Lớp ra có kích thước bằng số hành động, đại diện cho giá trị Q cho mỗi hành động.

```
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, output_dim)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.fc3(x)
```

Hình 3.3: Mô hình mạng noron trong thuật toán

3.2.2.1 Quy trình huấn luyện

Quá trình huấn luyện lặp lại trong num_episodes tập, mỗi tập đại diện cho một chuỗi hành động từ trạng thái khởi đầu đến khi đạt trạng thái kết thúc (thành công hoặc thất bại). Tại mỗi bước:

1. Tác tử chọn hành động a theo ε-greedy policy
2. Gửi hành động đến trang web, nhận phản hồi r và trạng thái kế tiếp s'.
3. Tính giá trị Q mục tiêu theo công thức:

$$Q_{target} = r + \gamma max_{a'} Q(s', a')$$

4. Tính loss và cập nhật trọng số mạng DQN bằng hàm mất mát MSE giữa Q hiện tại và Q mục tiêu.

$$loss = \left(Q(s, a) - Q_{target}\right)^2$$

5. Cập nhật trạng thái hiện tại với phản hồi mới.

3.2.2.2 Quy trình kiểm thử

Sau khi huấn luyện xong, tiến hành nạp mô hình đã huấn luyện, sau đó thử nghiệm trên một số lượng lớn (mặc định 10,000) tình huống khác nhau. Với mỗi lần thử:

- Tác tử thực hiện một tập hành động với chính sách ε=0.05 (tập trung khai thác).
- Kết quả được ghi ra file .txt, bao gồm thời gian trung bình trên mỗi bước và phần thưởng đạt được.

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Mục tiêu thực nghiệm

Mục tiêu của phần này là đánh giá và so sánh hiệu quả của mô hình Q-Learning và Deep Q-Learning trong việc khai thác lỗ hổng SQL Injection trong môi trường mô phỏng. Quá trình thực nghiệm thu thập các chỉ số về tỷ lệ thành công, số truy vấn trung bình, thời gian thực thi và phân phối hành động.

4.2 Thiết lập thực nghiệm

Trong quá trình thực nghiệm, mô hình Q-Learning và Deep Q-Learning được triển khai nhằm kiểm tra hiệu quả trong việc phát hiện và khai thác các lỗ hổng dạng SQL Injection. Mô hình học tăng cường được huấn luyện qua nhiều tập huấn luyện, với mỗi bước hành động tương ứng với một payload được gửi đi để kiểm tra phản hồi từ hệ thống.

Dữ liệu đầu ra của mô hình bao gồm các chỉ số thống kê được lưu trong hai tệp: metrics.csv và time_output.txt, cùng mô hình đã huấn luyện được lưu lại dưới dạng dqn_agent.pth và ql_agent.pth.

Môi trường thực nghiệm:

- Trang web Zixem SQLi chứa các lỗ hổng SQL Injection.
- Agent sử dụng mô hình Q-Learning và Deep Q-Learning.
- Action Space bao gồm nhiều payload phổ biến cho SQL Injection.
- Các tập dữ liệu kết quả được thu thập sau 10000 tập huấn luyện.

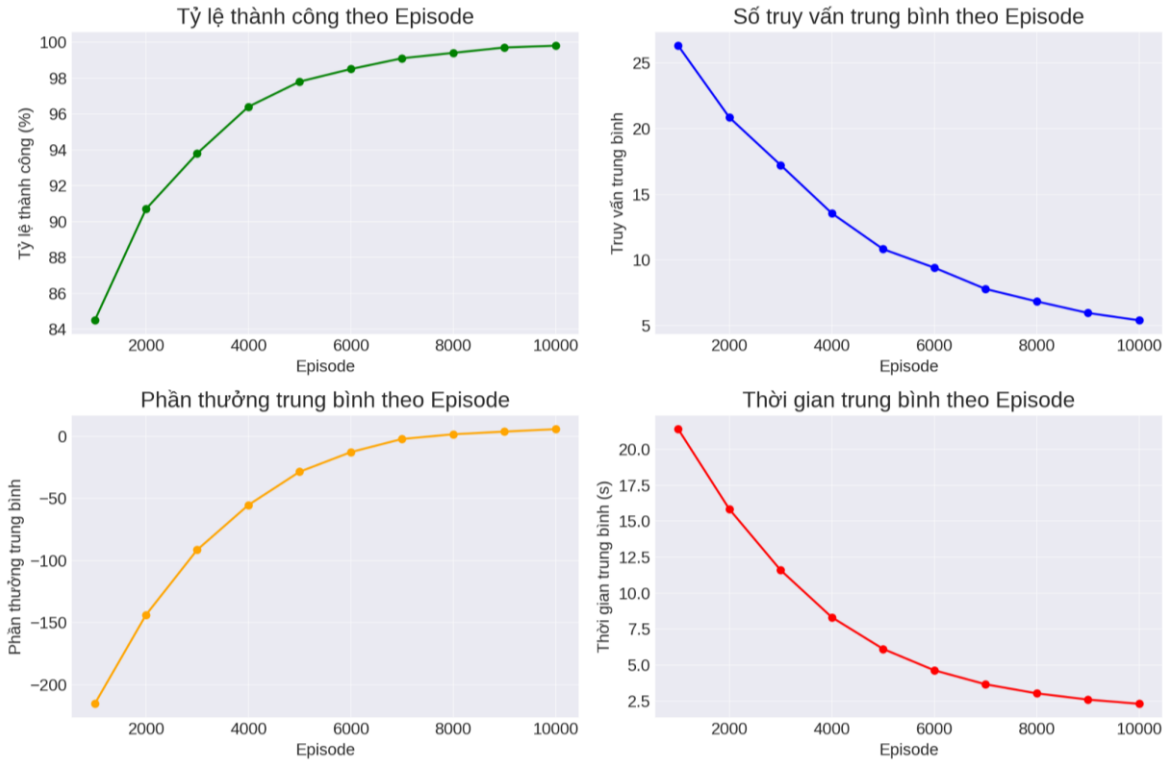
4.3 Thực nghiệm với mô hình sử dụng thuật toán Q-Learning

4.3.1 Kết quả huấn luyện

Bảng 4.1 dưới đây tóm tắt một số chỉ số chính sau mỗi mốc 1000 tập huấn luyện theo dữ liệu từ metrics.csv. Ngoài ra, hình 4.1 minh họa kết quả huấn luyện từ các thông số trong bảng 4.1:

Episode	Success Rate	Average Queries	Average Reward	Average Time	Failures Max Step
1000	84.51	26.32	-215.42	21.3948	47
2000	90.69	20.85	-143.76	15.8261	28
3000	93.80	17.23	-91.34	11.5927	19
4000	96.39	13.56	-55.23	8.3154	12
5000	97.83	10.83	-28.44	6.1178	6
6000	98.48	9.42	-12.63	4.6312	3
7000	99.12	7.80	-2.01	3.6589	1
8000	99.39	6.85	1.73	3.0293	0
9000	99.72	5.97	3.91	2.5926	0
10000	99.77	5.41	5.87	2.3035	0

Bảng 4.1: Tóm tắt kết quả huấn luyện mô hình Q-Learning



Hình 4.1: Biểu đồ kết quả huấn luyện mô hình Q-Learning

Dựa trên biểu đồ minh họa kết quả huấn luyện của mô hình Q-Learning trong Hình 4.1, ta nhận thấy sự tiến triển của mô hình qua các giai đoạn huấn luyện:

- Tỷ lệ thành công theo số tập huấn luyện: Mô hình Q-Learning đạt được tỷ lệ thành công tương đối ổn định sau khoảng 4000 tập huấn luyện, và dần tiến tới gần mức tối đa (~ 99.8%) ở giai đoạn cuối. Tuy nhiên, số lần agent thất bại do thực hiện hết số bước mà không đạt được mục tiêu vẫn còn xuất hiện rải rác đến khoảng tập 6000, sau đó giảm hẳn về 0.
- Số truy vấn trung bình: Chỉ số này cho thấy sự cải thiện ổn định từ hơn 26 truy vấn mỗi tập ở giai đoạn đầu, giảm dần xuống khoảng 5.4 ở cuối quá trình huấn luyện. Điều này cho thấy việc agent dần học được cách tối ưu hóa các bước đi để đạt mục tiêu nhanh hơn.
- Phần thưởng trung bình: Giá trị phần thưởng ban đầu rất thấp (~ -215), cho thấy nhiều hành động sai hoặc dẫn đến trạng thái thất bại. Tuy nhiên, theo thời gian, phần thưởng được cải thiện rõ rệt và chuyển dần sang vùng dương (~ 5.8), thể hiện khả năng học chính sách hiệu quả của Q-Learning.
- Thời gian trung bình mỗi tập huấn luyện: Giảm dần từ ~21.4 giây xuống còn khoảng 2.3 giây, cho thấy sự tiến bộ rõ rệt trong việc ra quyết định nhanh hơn và tránh được các đường đi dư thừa khi agent đã học được chiến lược tốt hơn.

4.3.2 Thời gian huấn luyện

Dữ liệu từ file time_output.txt ghi nhận thời gian hoàn thành của từng tập huấn luyện. Thời gian huấn luyện dao động khá rộng, từ khoảng 1.5 giây đến hơn 110 giây:

- Thời gian huấn luyện ngắn nhất: 1.5234 giây
- Thời gian huấn luyện dài nhất: 110.7491 giây
- Thời gian trung bình: ~27.35 giây

Sự biến động lớn về thời gian giữa các episode cho thấy đặc trưng của thuật toán Q-Learning, trong đó tác nhân cần khám phá nhiều hành động để cập nhật Q-Table một cách hiệu quả. Một số episode có thể mất nhiều thời gian hơn do yêu cầu thử nghiệm và học hỏi trong những trạng thái ít gặp hoặc phức tạp hơn. Dù vậy, phần lớn các tập vẫn được xử lý trong khoảng từ 10 đến 40 giây, cho thấy mô hình có mức độ ổn định tương đối trong môi trường huấn luyện. Điều này chứng tỏ rằng Q-Learning có khả năng khai thác và thích nghi hiệu quả.

4.3.3 Hiệu suất tổng thể

Sau khi hoàn thành huấn luyện, mô hình Q-Learning đạt được:

- Tổng thời gian huấn luyện: 59,800.32 giây (khoảng 16.6 giờ)
- Tỷ lệ thành công tổng thể: 96.70%
- Số truy vấn trung bình: 12.23 truy vấn/tập huấn luyện

Các chỉ số trên phản ánh rằng mô hình Q-Learning học khá nhanh và thể hiện được năng lực học hỏi đáng kể trong việc tìm ra chiến lược khai thác hiệu quả. Dù số lượng truy vấn cao hơn và thời gian huấn luyện còn dài, mô hình này vẫn đạt được tỷ lệ thành công tương đối cao sau 10,000 tập huấn luyện.

4.3.4 Đánh giá tổng thể

Mô hình Q-Learning cho thấy khả năng ứng dụng của các thuật toán học tăng cường cơ bản trong khai thác lỗ hổng SQL Injection:

- Tiến trình học diễn ra ổn định với tỷ lệ thành công tăng dần theo thời gian.
- Tối ưu hóa truy vấn dần theo từng giai đoạn.
- Chiến lược hành động được học rõ ràng thể hiện qua việc giảm dần số truy vấn và tăng dần phần thưởng trung bình.

Thông qua kết quả huấn luyện, có thể nhận định rằng Q-Learning là một lựa chọn khả thi trong các hệ thống kiểm thử bảo mật tự động, đặc biệt trong các môi trường có tài nguyên hạn chế hoặc không yêu cầu mô hình quá phức tạp. Q-Learning giúp mô hình học được các chiến lược khai thác hợp lý, góp phần tự động hóa quá trình kiểm thử lỗ hổng một cách có hệ thống và hiệu quả.

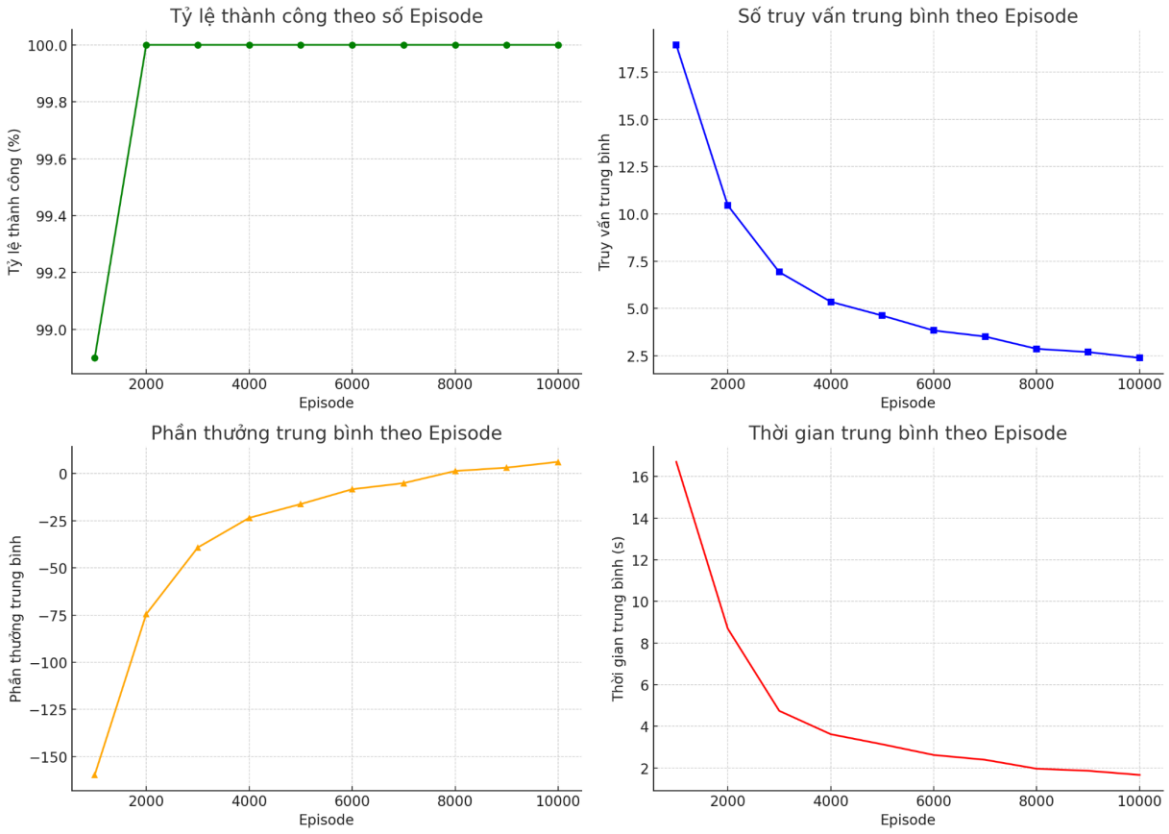
4.4 Thực nghiệm với mô hình sử dụng thực toán Deep Q-Learning

4.4.1 Kết quả huấn luyện

Bảng 4.2 dưới đây tóm tắt một số chỉ số chính sau mỗi mốc 1000 tập huấn luyện theo dữ liệu từ metrics.csv. Ngoài ra, hình 4.2 minh họa kết quả huấn luyện từ các thông số trong bảng 4.2:

Episode	Success Rate	Average Queries	Average Reward	Average Time	Failures Max Step
1000	98.9	18.94	-159.71	16.7072	11
2000	100	10.45	-74.49	8.6944	0
3000	100	6.92	-39.22	4.7431	0
4000	100	5.35	-23.47	3.6204	0
5000	100	4.62	-16.16	3.1363	0
6000	100	3.83	-8.3	2.6227	0
7000	100	3.51	-5.06	2.3912	0
8000	100	2.86	1.37	1.9621	0
9000	100	2.69	3.14	1.8632	0
10000	100	2.38	6.23	1.665	0

Bảng 4.2: Tóm tắt kết quả huấn luyện mô hình DQL



Hình 4.2: Biểu đồ kết quả huấn luyện mô hình DQL

Dựa trên biểu đồ minh họa kết quả huấn luyện của mô hình Deep Q-Learning trong Hình 4.2, ta nhận thấy sự cải thiện rõ rệt qua các giai đoạn huấn luyện:

- Tỷ lệ thành công theo số tập huấn luyện: Mô hình đạt được mức 100% rất sớm (từ ~2000 episode trở đi), cho thấy khả năng học rất nhanh các chiến lược hiệu quả để khai thác SQLi. Bên cạnh đó, chỉ có 11 lần agent thất bại do thực hiện hết số bước mà không đạt được mục tiêu ở giai đoạn đầu (1000 tập đầu tiên). Tuy nhiên, từ tập huấn luyện 2000 trở đi, chỉ số này đã về 0 và duy trì như vậy trong suốt quá trình huấn luyện.
- Số truy vấn trung bình: Giảm dần đều, từ gần 19 xuống chỉ còn khoảng 2.5, thể hiện sự tối ưu hóa hành động rõ rệt của agent.
- Phần thưởng trung bình: Tăng dần từ giá trị âm nặng (~ -160) lên trên 0 (~ 6), nghĩa là càng về sau agent có xu hướng giảm các hành động dẫn tới lỗi hoặc thất bại, phản ánh khả năng học chính sách hiệu quả.
- Thời gian trung bình mỗi tập huấn luyện: Giảm mạnh từ ~17 giây xuống còn ~1.6 giây, cải thiện rõ ràng hiệu suất xử lý.

4.4.2 Thời gian huấn luyện

Dữ liệu từ time_output.txt ghi nhận thời gian hoàn thành của từng tập huấn luyện. Trong đó, thời gian dao động khá rộng từ dưới 1 giây đến hơn 85 giây:

- Thời gian huấn luyện ngắn nhất: 0.6416 giây
- Thời gian huấn luyện dài nhất: 85.8282 giây
- Thời gian trung bình: ~ 20.43 giây

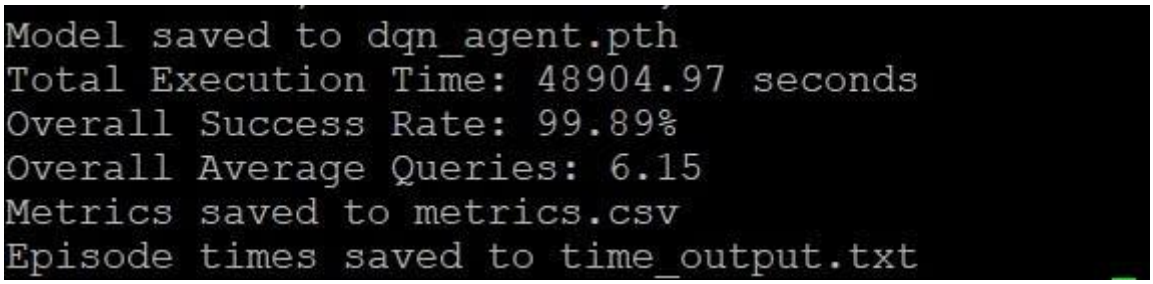
Điều này phản ánh sự khác biệt độ khó giữa các thử nghiệm, có thể do độ phức tạp của độ sâu của chuỗi hành động cần thực hiện. Tuy nhiên, độ lệch không quá lớn và phần lớn các episode đều hoàn thành trong vòng 5–20 giây, cho thấy mô hình vẫn giữ được tính ổn định trong suốt quá trình huấn luyện.

4.4.3 Hiệu suất tổng thể

Sau khi hoàn thành huấn luyện, mô hình DQL đạt được:

- Tổng thời gian huấn luyện: 48,904.97 giây (khoảng 13.6 giờ)

- Tỷ lệ thành công tổng thể: 99.89%
- Số truy vấn trung bình: 6.15 truy vấn/tập huấn luyện



Hình 4.3: Kết quả sau quá trình huấn luyện

Các chỉ số trên cho thấy mô hình DQL có hiệu quả rất cao trong việc học chiến lược khai thác lỗ hổng SQLi. Tỷ lệ thành công gần như tuyệt đối và số lượng truy vấn cần thiết tương đối ít, phản ánh khả năng tối ưu hành vi của agent.

4.4.4 Đánh giá tổng thể

Mô hình DQL thể hiện khả năng học tăng cường rất hiệu quả trong việc khai thác lỗ hổng SQL Injection:

- Học nhanh, đạt hiệu quả cao chỉ sau khoảng 2000 tập.
- Khả năng tối ưu truy vấn và giảm thiểu lỗi đáng kể.
- Phân phối hành động hợp lý từ đó học được chiến lược hiệu quả nhất.

Thông qua các kết quả trên, có thể thấy Deep Q-Learning là một phương pháp hiệu quả để tự động hóa quá trình tấn công khai thác lỗ hổng SQLi. Với tỉ lệ thành công gần như tuyệt đối và số truy vấn được tối ưu hóa, Deep Q-Learning cho thấy tiềm năng cao trong việc xây dựng các hệ thống kiểm thử bảo mật tự động.

4.5 So sánh và đánh giá

Sau khi tiến hành thực nghiệm trên cả hai mô hình Q-Learning và Deep Q-Learning, nhóm nhận thấy có khá nhiều điểm khác biệt rõ rệt giữa hai phương pháp này, cả về hiệu suất huấn luyện, khả năng khai thác lỗ hổng lẫn thời gian xử lý. Do đó, nhóm tiến hành so sánh và đánh giá tổng thể hai mô hình dựa trên các khía cạnh sau: hiệu quả học tập, tỷ lệ thành công, số truy vấn trung bình, phần thưởng trung bình và thời gian huấn luyện.

Hiệu quả học tập: Mô hình Deep Q-Learning cho kết quả rất khả quan khi chỉ sau khoảng 2000 tập huấn luyện thì agent đã đạt được tỷ lệ thành công 100%. Trong khi đó, mô hình Q-Learning phải mất hơn 8000 tập mới chạm ngưỡng gần tương đương. Điều này cho thấy DQL học nhanh hơn và ổn định hơn trong việc tìm ra chiến lược tấn công hiệu quả.

Tỷ lệ thành công: Cả hai mô hình đều đạt tỷ lệ thành công cao, tuy nhiên DQL có tỷ lệ cao hơn với 99.89% so với 96.70% của Q-Learning. Đặc biệt, số lần thất bại do đạt đến số bước tối đa trong DQL gần như bằng 0 sau 2000 tập, cho thấy agent học được chiến lược hiệu quả và ổn định hơn so với Q-Learning.

Số truy vấn trung bình: Số lượng truy vấn trung bình là một chỉ số quan trọng để đánh giá khả năng tối ưu hóa hành động của agent. Mô hình DQL giảm số truy vấn từ gần 19 xuống còn 2.38 truy vấn/tập huấn luyện, trong khi Q-Learning chỉ giảm từ 26.32 xuống còn 5.41 truy vấn/tập huấn luyện. Như vậy, DQL không chỉ học nhanh mà còn ra quyết định chính xác với ít bước hơn.

Phần thưởng trung bình: Phần thưởng trung bình phản ánh mức độ hiệu quả của hành động. Mô hình DQL bắt đầu với phần thưởng âm lớn (~ -159) và tăng lên 6.23, trong khi Q-Learning từ ~ -215 lên 5.87. Mặc dù kết quả cuối cùng tương đối gần nhau, nhưng quá trình tăng của DQL diễn ra sớm và mượt hơn, thể hiện khả năng tránh lỗi và ra quyết định tốt hơn trong giai đoạn sớm.

Thời gian huấn luyện: Tổng thời gian huấn luyện của DQL là khoảng 48,905 giây (~13.6 giờ), nhanh hơn Q-Learning với 59,800 giây (~16.6 giờ). Điều này thể hiện rằng dù mô hình DQL phức tạp hơn, khả năng khai thác hiệu quả và giảm thiểu thất bại đã giúp rút ngắn thời gian huấn luyện tổng thể.

Từ quá trình thực nghiệm và các số liệu thu được, có thể thấy rằng Deep Q-Learning là phương pháp cho hiệu quả tốt hơn rõ rệt so với Q-Learning trong bài toán khai thác lỗ hổng SQL Injection. DQL học nhanh hơn, tỷ lệ thành công cao hơn, số truy vấn ít hơn và thời gian huấn luyện cũng ngắn hơn. Dù vậy, Q-Learning vẫn có ưu điểm là đơn giản, dễ triển khai và phù hợp với các môi trường không yêu cầu cao về tài nguyên. Tùy vào mục đích sử dụng và điều kiện hệ thống, mỗi thuật toán đều có thể phát huy thế mạnh riêng của mình.

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong bối cảnh các cuộc tấn công mạng ngày càng tinh vi và khó phòng ngừa, đặc biệt là tấn công SQL Injection, việc tự động hóa kiểm thử bảo mật đang trở thành một hướng đi cần thiết trong lĩnh vực an toàn thông tin. Đồ án này đã xây dựng và đánh giá một hệ thống kiểm thử xâm nhập tự động dựa trên học tăng cường, với hai phương pháp chính là Q-Learning và Deep Q-Learning. Hệ thống có khả năng huấn luyện agent tự động sinh và gửi các payload SQLi nhằm khai thác lỗ hổng, từ đó đánh giá khả năng phòng thủ của hệ thống mục tiêu.

Các kết quả thực nghiệm cho thấy:

- Mô hình Q-Learning có thể học được chiến lược tấn công hiệu quả sau quá trình huấn luyện dài, đạt tỷ lệ thành công ~99.77% sau 10.000 tập, tuy nhiên mất nhiều thời gian và số truy vấn trung bình cao hơn so với DQL.
- Mô hình Deep Q-Learning thể hiện hiệu suất vượt trội, đạt tỷ lệ thành công tuyệt đối 100% chỉ sau 2.000 tập huấn luyện, số truy vấn giảm đáng kể (~2.38 truy vấn/tập huấn luyện) và thời gian xử lý trung bình thấp hơn rõ rệt.

Điều này chứng minh rằng việc áp dụng Deep Q-Learning vào quá trình kiểm thử xâm nhập mang lại hiệu quả rõ rệt trong việc tự động hóa khai thác SQLi, đồng thời tăng khả năng thích nghi, khái quát hóa và tối ưu hóa hành động so với phương pháp Q-Learning truyền thống.

5.2 Hạn chế

Mặc dù hệ thống đã đạt được những kết quả khả quan nhưng mô hình vẫn tồn tại một số hạn chế cần được cải thiện:

- Môi trường thử nghiệm còn đơn giản, chưa bao gồm nhiều dạng lỗ hổng SQLi nâng cao hoặc sự hiện diện của các hệ thống phòng thủ phức tạp như WAF.
- Không gian hành động được xây dựng thủ công, có thể dẫn đến việc bỏ sót các payload hiệu quả hoặc bị giới hạn trong khả năng khai thác.

- Quá trình huấn luyện sử dụng reward function cố định, chưa tích hợp khả năng thích nghi linh hoạt theo từng phản hồi cụ thể hoặc mức độ nghiêm trọng của lỗ hổng.

5.3 Hướng phát triển

Trong tương lai, hệ thống có thể được cải tiến và mở rộng theo các hướng sau:

- Mở rộng môi trường thực nghiệm: Tích hợp nhiều loại hệ thống web thực tế, bao gồm các hệ thống có cơ chế phòng thủ tiên tiến như WAF, honeypot và kết hợp mô phỏng hành vi người dùng nhằm tăng độ chân thực và thử thách cho agent.
- Tự động sinh payload: Kết hợp mô hình học ngôn ngữ như T5 để tự động sinh các payload tấn công mới, giúp mở rộng không gian hành động và tránh bị lặp lại mẫu.
- Tối ưu hàm phần thưởng: Thiết kế hàm phần thưởng động, linh hoạt theo từng trạng thái và phản hồi hệ thống, từ đó giúp agent học được chiến lược tốt hơn trong môi trường đa dạng.
- Chuyển sang các thuật toán nâng cao hơn: Thử nghiệm các thuật toán học tăng cường hiện đại như Double DQN, Dueling DQN, Actor-Critic hoặc Proximal Policy Optimization (PPO) nhằm tăng tốc hội tụ và cải thiện khả năng khái quát.
- Triển khai trên quy mô lớn: Xây dựng hệ thống kiểm thử bảo mật toàn diện có thể tích hợp nhiều kiểu tấn công (XSS, LFI, CSRF...) và hỗ trợ kiểm thử đồng thời nhiều mục tiêu, phục vụ thực tiễn trong quy trình DevSecOps.
- Ứng dụng vào hệ thống thực tế: Triển khai mô hình RL đã huấn luyện vào kiểm thử bảo mật bán tự động trên các hệ thống thật hoặc môi trường staging, giúp đánh giá hiệu quả tấn công trong điều kiện thực tế và liên tục thích nghi với phản hồi từ hệ thống mục tiêu.

TÀI LIỆU THAM KHẢO

Jabr, I., Salman, Y., Shqair, M., & Hawash, A. (2024). Penetration Testing and Attack Automation Simulation: Deep Reinforcement Learning Approach. *An-Najah University Journal for Research-A (Natural Sciences)*, *39*(1), None-None.

Erdődi, L., Sommervoll, Å. Å., & Zennaro, F. M. (2021). Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. *Journal of Information Security and Applications*, *61*, 102903.

Hasan, M., Balbahaith, Z., & Tarique, M. (2019, November). Detection of SQL injection attacks: a machine learning approach. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)* (pp. 1-6). IEEE.

Nguyen, H. P. T., Hasegawa, K., Fukushima, K., & Beuran, R. (2025). PenGym: Realistic training environment for reinforcement learning pentesting agents. *Computers & Security*, *148*, 104140.

Leung, D., Tsai, O., Hashemi, K., Tayebi, B., & Tayebi, M. A. (2024, October). XploitSQL: Advancing Adversarial SQL Injection Attack Generation with Language Models and Reinforcement Learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (pp. 4653-4660).

Marinelli, R., Erdődi, L., Sommervoll, Å. Å., & Zennaro, F. M. (2024, October). Extending Q-Learning Agents in SQLi Environments. In *2024 Cyber Awareness and Research Symposium (CARS)* (pp. 1-6). IEEE.

Sommervoll, Å. Å., Erdődi, L., & Zennaro, F. M. (2024). Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents. *International Journal of Information Security*, *23*(1), 225-246.

Boekweg, K. I. (2024). *Developing a SQL Injection Exploitation Tool with Natural Language Generation* (Master's thesis, Brigham Young University).

Al Wahaibi, S., Foley, M., & Maffeis, S. (2023). {SQIRL}:{Grey-Box} Detection of {SQL} Injection Vulnerabilities Using Reinforcement Learning. In 32nd USENIX Security Symposium (USENIX Security 23) (pp. 6097-6114).