# THÔNG TIN CHUNG CỦA NHÓM

- Link YouTube video của báo cáo (tối đa 5 phút): *https://youtu.be/jyXXSeXHu6Q*
- Link slides (dạng .pdf đặt trên Github của nhóm): *https://github.com/letrongnhanlp/CS2205.CH190/blob/main/Nh%C3%A2n%20L%C3%AA%20Tr%E1%BB%8Dng%20-%20CS2205.FEB2025.DeCuong.FinalReport.Template.Slide.pdf*

- Họ và Tên: Lê Trọng Nhân
- MSSV: 240101060

- Lớp: CS2205.CH190
- Tự đánh giá (điểm tổng kết môn): 8/10
- Số buổi vắng: 0
- Số câu hỏi QT cá nhân: 5
- Link Github: https://github.com/letrongnhanlp/CS2205.CH190

# ĐỀ CƯƠNG NGHIÊN CỨU

**TÊN ĐỀ TÀI (IN HOA)**

PHƯƠNG PHÁP TẠO BIẾN THỂ MÃ ĐỘC WINDOWS DỰA TRÊN HỌC TĂNG CƯỜNG CÓ KIỂM CHỨNG CHỨC NĂNG BẰNG PHÂN TÍCH ĐỘNG

**TÊN ĐỀ TÀI TIẾNG ANH (IN HOA)**

A METHOD OF WINDOWS MALWARE MUTATION USING REINFORCEMENT LEARNING WITH DYNAMIC ANALYSIS-BASED FUNCTIONALITY VALIDATION

**TÓM TẮT** *(Tối đa 400 từ)*

Trong những năm gần đây, các nhà nghiên cứu về an toàn thông tin đã áp dụng các mô hình học máy/học sâu để tìm những lỗ hổng hoặc tấn công xâm nhập. Hiện tại, có những giải pháp ứng dụng các mô hình máy học để phát hiện mã độc đem đến kết quả cao trong việc phân loại mã độc. Tuy nhiên, các mô hình này dễ bị đánh lừa bởi phép biến đổi khôn khéo cho đối tượng đầu vào, dễ bị vượt mặt bởi những mẫu đối kháng được điều chỉnh một cách tinh vi. Để đánh giá toàn diện của các giải pháp phát hiện mã độc, cần xem xét khả năng chống chịu của giải pháp trước các biến thể mã độc. Việc tạo biến thể từ mã độc gốc cũng là hướng tiếp cận thiết thực, giúp mô phỏng tình huống thực tế.

Trong bài toán tạo biến thể mã độc cũng có những yêu cầu nhất định, đặc biệt là việc bảo toàn được khả năng thực thi cũng như chức năng độc hại. Do khi thực hiện những phép biến đổi có thể dẫn đến những rủi ro khiến tệp thực thi bị hỏng và thậm chí mất đi chức năng ban đầu. Khi đó, các mẫu biến thể được tạo ra sẽ trở nên vô nghĩa và không thể sử dụng được. Do đó, vấn đề bảo toàn tính toàn vẹn về chức năng là một điều rất quan trọng trong bài toán tạo những biến đổi trên những tập tin mã độc để tạo ra các biến thể đa dạng.

**GIỚI THIỆU** *(Tối đa 1 trang A4)*

Đã có nhiều cách tiếp cận và phương pháp đã được áp dụng để tạo biến thể mã

độc. Goodfellow và cộng sự [1] đề xuất tạo mẫu đối kháng dựa trên gradient sử dụng phương pháp Fast Gradient Sign Method (FGSM). Trong một nghiên cứu khác, Yuste và cộng sự [2] đề xuất tạo mẫu đối kháng dựa trên Genetic Algorithm (Thuật toán di truyền) bằng cách chèn nhiễu vào những khoảng trống (code cavity) giữa header với section và giữa các section với nhau. Lucas và cộng sự [3] cũng đề xuất tạo mẫu đối kháng dựa trên sử dụng các kỹ thuật rối mã (Code Obfuscation) khác nhau.

Bên cạnh đó, một phương pháp tiềm năng cho phép sửa đổi mã độc để trốn tránh phát hiện của những mô hình học máy là việc cách sử dụng phương pháp Học tăng cường (Reinforcement Learning). Trong đó, Anderson và cộng sự [4] là những tác giả đầu tiên áp dụng RL để tạo biến thể mã độc với mười hành động khác nhau được thiết kế để bảo toàn ngữ nghĩa và chức năng, cấu trúc của chương trình. Dựa trên ý tưởng tương tự, Fang và cộng sự [5], đã thực hiện một cách tiếp cận tương tự bằng nhưng với kích thước đầu vào và không gian hành động đều được giảm xuống nhỏ hơn. AIMED-RL [6] cũng đề xuất phương pháp RL tạo mẫu đối kháng dựa trên một loạt những hành động biến đổi tệp mã độc, tuy có đề xuất điểm đánh giá liên quan đến bảo toàn chức năng nhưng điểm này dựa trên so sánh các byte thô nên chưa thực sự hiệu quả.

Nhìn chung, các nghiên cứu tạo biến thể mã độc Windows, với nhiều hướng tiếp cận khác nhau, hầu hết đều chưa có cơ chế kiểm chứng chức năng của biến thể một cách đầy đủ.

Từ động lực giải quyết vấn đề thiếu cơ chế kiểm chứng chức năng của biến thể mã độc, cùng tiềm năng của Học tăng cường (RL), một giải pháp đầy hứa hẹn là tích hợp thêm các bước kiểm chứng chức năng của biến thể làm điều kiện để tối ưu hóa việc lựa chọn các phép biến đổi trong mô hình RL, nhằm tạo ra các mẫu biến thể không chỉ có khả năng qua mặt mà còn thực sự là các mẫu có ý nghĩa và thực thi được.

Do vậy, trong đề tài này, chúng tôi hướng tới đề xuất một cơ chế kiểm chứng dựa trên phân tích động, theo dõi và so sánh các chức năng được thực thi của chương trình mã độc ban đầu và biến thể mã độc đã biến đổi, dựa trên trích xuất các tập API và so

sánh các tập API với nhau. Cơ chế kiểm chứng này có thể được tích hợp vào mô hình Học tăng cường như một mô-đun hỗ trợ quá trình huấn luyện để tối ưu hóa việc lựa chọn các phép biến đổi mã độc hiệu quả.

**Input:** Một hoặc nhiều mẫu mã độc Windows (PE file)

**Output:** Một hoặc nhiều mẫu mã độc đã được chỉnh sửa để qua mặt trình phát hiện mã độc học máy nhưng vẫn bảo toàn chức năng độc hại.

**MỤC TIÊU** *(Viết trong vòng 3 mục tiêu)*

1. Nghiên cứu và ứng dụng học tăng cường (RL) vào bài toán tạo biến thể mã độc Windows, tập trung vào việc bảo toàn chức năng gốc sau biến đổi.
2. Xây dựng hệ thống tạo biến thể tự động tích hợp cơ chế kiểm chứng chức năng dựa trên phân tích động (API call analysis).
3. Đánh giá hiệu quả hệ thống thông qua khả năng qua mặt mô hình phát hiện mã độc và tỷ lệ bảo toàn chức năng so với các phương pháp khác.

**NỘI DUNG VÀ PHƯƠNG PHÁP**

1. Nghiên cứu cơ sở lý thuyết
    a. Nội dung:
        - Tìm hiểu định dạng PE file, đặc điểm mã độc Windows.
        - Phân tích các kỹ thuật biến đổi mã độc (code obfuscation, RL).
        - Nghiên cứu học tăng cường và ứng dụng trong tạo biến thể bảo toàn chức năng mã độc Window.
    b. Phương pháp:
        - Đọc các công trình về phân tích và phát hiện mã độc Windows ở cả hướng động và tĩnh, nắm được các phương pháp và các thông tin thường được sử dụng để phát hiện mã độc.
        - Đọc các công trình về áp dụng học tăng cường, đặc biệt là trong việc phát hiện mã độc (Windows hoặc các loại khác)
2. Xây dựng hệ thống tạo biến thể dựa trên học tăng cường có kiểm chứng
    a. Nội dung:

- Thiết kế không gian hành động (action space) phù hợp.
- Phát triển mô-đun kiểm chứng chức năng bằng phân tích API call (so sánh vector API qua cosine similarity).

b. Phương pháp:

- Sử dụng môi trường RL với hàm reward dựa trên:
  i. Khả năng qua mặt trình phát hiện (MalConv).
  ii. Kiểm tra chức năng (so sánh API call gốc và biến thể).
- Triển khai các phép biến đổi mã độc (dùng thư viện LIEF, PEfile) và tích hợp sandbox (Cuckoo) để kiểm tra chức năng của mã độc.

3. Thực nghiệm và đánh giá kết quả

a. Nội dung:

- Đánh giá bộ kiểm tra chức năng.
- Đo lường hiệu quả hệ thống trên tập dữ liệu mã độc thực tế
- So sánh với phương pháp tạo biến thể khác

b. Phương pháp:

- Thu thập dữ liệu đầu vào cho toàn bộ mô hình là mẫu chương trình thực thi malware (dưới dạng file PE).
- Đánh giá khả năng kiểm chứng chức năng của mô-đun đã phát triển (độ chính xác).
- Thực nghiệm so sánh hiệu quả của mô hình RL và một số phương pháp tạo biến thể mã độc khác như việc chọn ngẫu nhiên action.
- Tiêu chí đánh giá cho học tăng cường:
  i. Tỷ lệ thành công khi qua mặt trình phát hiện (Evasion Rate).
  ii. Tỷ lệ bảo toàn chức năng (Functional Integrity).
  iii. Thời gian sinh biến thể (Throughput).

**KẾT QUẢ MONG ĐỢI**

1. Độ chính xác của mô-đun kiểm chứng đạt kết quả cao (lớn hơn 90%)
2. Hoàn thành huấn luyện agent RL có khả năng biến đổi mã độc vượt mặt tốt hơn

hoặc bằng các phương pháp hiện tại nhưng khả năng tạo bảo toàn chức năng ban đầu tốt hơn.

**TÀI LIỆU THAM KHẢO** *(Định dạng DBLP)*

[1] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, Charles Nicholas:
Malware Detection by Eating a Whole EXE. arXiv:1710.09435

[2] J. Yuste, E. G. Pardo and J. Tapiador:
"Optimization of code caves in malware binaries to evade machine learning detectors," Computers & Security, vol. 116, no. Elsevier, p. 102643, 2022.

[3] K. Lucas, M. Sharif, L. Bauer, M. K. Reiter and S. Shintre:
"Malware makeover: Breaking ml-based static analysis by modifying executable bytes," Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, pp. 744-758, 2021.

[4] Hyrum S. Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth: Learning to evade static pe machine learning malware models via rl. ArXiv, 2018.

[5] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, and H. Huang:
Evading anti-malware engines with deep reinforcement learning. IEEE Access, 7:48867–48879, 2019.

[6] Labaca-Castro, Raphael and Franz, Sebastian and Rodosek, Gabi Dreo:
AIMED-RL: Exploring Adversarial Malware Examples with Reinforcement Learning. Joint European Conference on Machine Learning and Knowledge Discovery in Databases

[7] Hyrum S. Anderson, Phil Roth. EMBER:
An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv:1804.04637v2