


THÔNG TIN CHUNG CỦA BÁO CÁO

- Link YouTube video của báo cáo (tối đa 5 phút): https://youtu.be/K8I6yYSI_DY
- Link slides (dạng .pdf đặt trên Github):
<https://github.com/pinaneK/CS2205.CH181/blob/main/Slide%20-%20CS2205.CH181%20-%20Ng%C3%B4%20D%E1%BB%A9c%20H%C3%A0ng%20S%C6%A1n%20-%20230202030.pdf>
- *Mỗi thành viên của nhóm điền thông tin vào một dòng theo mẫu bên dưới*
- *Sau đó điền vào Đề cương nghiên cứu (tối đa 5 trang), rồi chọn Turn in*

<ul style="list-style-type: none">• Họ và Tên: Ngô Đức Hoàng Sơn• MSSV: 230202030 	<ul style="list-style-type: none">• Lớp: CS2205.CH181• Tự đánh giá (điểm tổng kết môn): 8.5/10• Số buổi vắng: 2• Số câu hỏi QT cá nhân: 3• Link Github: https://github.com/pinaneK/CS2205.CH181
--	---

ĐỀ CƯƠNG NGHIÊN CỨU

TÊN ĐỀ TÀI (IN HOA)

MỘT CÁCH TIẾP CẬN SỬ DỤNG HỌC SÂU TRONG PHÁT HIỆN MÃ ĐỘC POWERSHELL

TÊN ĐỀ TÀI TIẾNG ANH (IN HOA)

A DEEP LEARNING APPROACH TO DETECTING POWERSHELL MALWARE

TÓM TẮT *(Tối đa 400 từ)*

Mã độc từ lâu đã trở thành một mối đe dọa lớn, thường được sử dụng như một công cụ nhằm xâm nhập và tấn công hệ thống nạn nhân. Một trong những xu hướng phát triển mã độc đó chính là tận dụng các công cụ có sẵn trên hệ thống, và trong đó phổ biến nhất là PowerShell. Loại mã độc này lợi dụng việc PowerShell được coi là một chương trình hợp pháp nhằm lẩn trốn phát hiện, cùng với đó là thực thi trực tiếp trên bộ nhớ (memory), từ đây có thể loại bỏ dấu vết xâm nhập trên máy nạn nhân. Từ đây, các công cụ truyền thống không có khả năng phòng thủ trước các cuộc tấn công từ mã độc PowerShell. Vì vậy, trong nghiên cứu này chúng tôi đề xuất một phương pháp phát hiện mã độc PowerShell sử dụng mô hình ngôn ngữ nhằm khắc phục các điểm yếu của các phương pháp truyền thống. Cụ thể chúng tôi thực hiện xây dựng một quy trình: Gỡ rối và làm sạch các đoạn mã PowerShell nhằm tránh các kẻ tấn công trốn tránh với mô hình phát hiện; tiền huấn luyện các đoạn mã PowerShell với mô hình DeBERTa; cuối cùng thực hiện việc tinh chỉnh mô hình nhằm phát hiện các mã độc PowerShell.

GIỚI THIỆU (Tối đa 1 trang A4)

Một trong những xu hướng mới của mã độc là chuyển từ việc cài đặt mã độc trực tiếp vào hệ thống của nạn nhân sang việc tạo ra các chương trình không thực thi từ ổ cứng hoặc các ứng dụng có tính hợp lệ, nhưng lại được sử dụng để thực hiện các cuộc tấn công [1], tiêu biểu như tấn công mã độc PowerShell. Việc phát hiện các cuộc tấn công này trở nên khó khăn vì thường không để lại dấu vết hoặc tập tin có hại trên máy tính của nạn nhân, và các phương pháp truyền thống không thể phát hiện chúng.

Theo khảo sát [2], các phương pháp được sử dụng nhằm phát hiện mã độc PowerShell như dựa trên quy tắc và học sâu đang được tiếp cận. Với các phương pháp dựa trên quy tắc [3], [4] có khả năng phát hiện nhanh chóng nhưng mang tính bị động, không thể nào phát hiện các lỗ hổng chưa từng biết. Vì vậy, các phương pháp phát hiện sử dụng học sâu đang là một giải pháp đầy tiềm năng vì không dựa vào các quy tắc cố định. Tuy đã đạt cho thấy được sự hiệu quả [5], [6], các nghiên cứu này chỉ thực hiện việc nhúng ở mức độ token, chưa cân nhắc ngữ cảnh của toàn bộ đoạn mã, gây ra sự khó khăn khi sử dụng các véc tơ nhúng đầu ra này nhằm phân loại.

Ngoài ra, các mô hình ngôn ngữ cũng cho thấy sự phát triển mạnh mẽ trong những năm gần đây, đặc biệt là các mô hình transformer-based [7]. Chúng đã được ứng dụng trong việc phát hiện mã độc và đạt một số thành công nhất định [8], [9]. Trong số đó DeBERTa [10] mang lại sự cải tiến khá lớn so với các mô hình trước đó. Tuy vậy, mô hình này lại chưa được sử dụng rộng rãi, đặc biệt là trong việc phát hiện mã độc.

Nhận thấy được vấn đề còn tồn đọng trong các nghiên cứu phát hiện mã độc phi mã, chúng tôi đề xuất một phương pháp phát hiện các cuộc tấn công có độ phức tạp cao như mã độc PowerShell sử dụng học sâu với mô hình xử lý ngôn ngữ tự nhiên. Phương pháp đề xuất có thể phát hiện các mã độc PowerShell kể cả khi chúng đã bị làm rối mã thông qua quá trình gỡ rối và làm sạch. Cùng với đó là việc ứng dụng mô hình ngôn ngữ DeBERTa nhằm học và nhúng toàn bộ đoạn mã thay vì ở mức token.

Input: Một tập tin mã nguồn PowerShell.

Output: Phát hiện được rằng mã nguồn PowerShell là mã độc hay không.

MỤC TIÊU

(Viết trong vòng 3 mục tiêu, lưu ý về tính khả thi và có thể đánh giá được)

- **Mục tiêu 1:** Xây dựng được bộ dữ liệu không nhãn và có nhãn về mã nguồn PowerShell.
- **Mục tiêu 2:** Xây dựng mô hình phân loại dựa trên mô hình ngôn ngữ DeBERTa nhằm phát hiện mã độc PowerShell một cách hiệu quả.
- **Mục tiêu 3:** Cung cấp một mô hình được tiền huấn luyện nhằm dễ dàng tinh chỉnh cho các tác vụ khác.

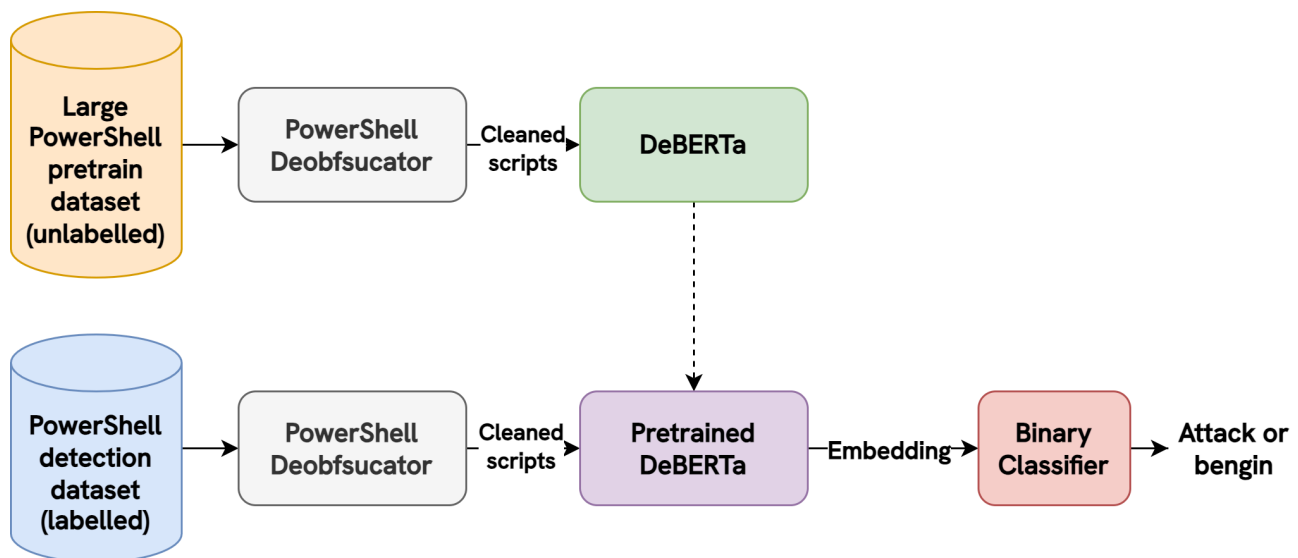
NỘI DUNG VÀ PHƯƠNG PHÁP

(Viết nội dung và phương pháp thực hiện để đạt được các mục tiêu đã nêu)

- **Nội dung 1:** Tìm hiểu về mã độc phi mã và các phương pháp phát hiện mã độc PowerShell.
 - Nghiên cứu và khảo sát về đặc điểm, hình thức và phương pháp tấn công của mã độc PowerShell.
 - Nghiên cứu và đánh giá các phương pháp phát hiện mã độc PowerShell hiện có.
 - Nghiên cứu về mô hình ngôn ngữ, mô hình DeBERTa và khả năng ứng dụng của mô hình ngôn ngữ trong phát hiện mã độc PowerShell.
- **Nội dung 2:** Tìm hiểu và xây dựng bộ dữ liệu về mã nguồn PowerShell.
 - Nghiên cứu và khảo sát về các nguồn dữ liệu cung cấp các mẫu mã nguồn PowerShell, cả về mã độc và lành tính.
 - Nghiên cứu và tìm hiểu về các xây dựng bộ dữ liệu (có nhãn và không có nhãn).
- **Nội dung 3:** Tìm hiểu về các phương pháp gỡ rối mã nguồn PowerShell.
 - Nghiên cứu và khảo sát các phương pháp gỡ rối mã nguồn PowerShell.
 - Nghiên cứu về các phương pháp gỡ rối mã nhằm giải quyết các phương pháp gỡ rối mã trên. Đánh giá và chọn ra phương pháp tối ưu nhất.
- **Nội dung 4:** Nghiên cứu và xây dựng mô hình phân loại dựa trên mô hình ngôn

ngữ DeBERTa nhằm phát hiện mã độc PowerShell.

- Nghiên cứu và xây dựng mô hình phát hiện mã độc PowerShell.
- Mô hình tổng quan được đề xuất như *hình 1*. Bộ dữ liệu không nhãn và có nhãn sẽ đều được gỡ rồi nhằm chuẩn hóa. Bộ dữ liệu không nhãn sẽ được tiền huấn luyện, sau đó sẽ được tinh chỉnh với bộ dữ liệu có nhãn nhằm phát hiện mã độc PowerShell



Hình 1: Mô hình tổng quan phương pháp đề xuất

KẾT QUẢ MONG ĐỢI

(Viết kết quả phù hợp với mục tiêu đặt ra, trên cơ sở nội dung nghiên cứu ở trên)

- Xây dựng được bộ dữ liệu không nhãn phục vụ tiền huấn luyện và có nhãn của mã nguồn PowerShell.
- Phương pháp gỡ rồi có tỉ lệ chính xác từ 90% - 95%.
- Mô hình phát hiện mã độc có tỉ lệ phát hiện trên 95%, và cải thiện hiệu suất so với các công trình khác.
- Tiền huấn luyện thành công mô hình DeBERTa.

TÀI LIỆU THAM KHẢO (Định dạng DBLP)

[1]. Jannatul Ferdous, Md. Rafiqul Islam, Arash Mahboubi, Md Zahidul Islam: A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms. IEEE Access 11: 121118-121141 (2023).

- [2]. Side Liu, Guojun Peng, Haitao Zeng, Jianming Fu: A survey on the evolution of fileless attacks and detection techniques. *Comput. Secur.* 137: 103653 (2024).
- [3]. Victor M. Alvarez: Yara: The pattern matching swiss knife. GitHub: VirusTotal/yara. URL: <https://github.com/VirusTotal/yara>. (2014).
- [4]. SigmaHQ: Sigma: The shareable detection format for security professionals. GitHub: SigmaHQ/sigma. URL: <https://github.com/SigmaHQ/sigma> (2019).
- [5]. Yong Fang, Xiangyu Zhou, Cheng Huang: Effective method for detecting malicious PowerShell scripts based on hybrid features☆. *Neurocomputing* 448: 30-39 (2021).
- [6]. Mamoru Mimura, Yui Tajiri: Static detection of malicious PowerShell based on word embeddings. *Internet Things* 15: 100404 (2021).
- [7]. Diksha Khurana, Aditya Koli, Kiran Khatter, Sukhdev Singh: Natural language processing: state of the art, current trends and challenges. *Multim. Tools Appl.* 82(3): 3713-3744 (2023).
- [8]. Zhifeng Xu, Xianjin Fang, Gaoming Yang: Malbert: A novel pre-training method for malware detection. *Comput. Secur.* 111: 102458 (2021).
- [9]. Ferhat Demirkiran, Aykut Çayır, Ugur Ünal, Hasan Dag: An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. *Comput. Secur.* 121: 102846 (2022).
- [10]. Pengcheng He, Jianfeng Gao, Weizhu Chen: DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *ICLR* 2023.