ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO CUỐI KỲ

AN TOÀN MẠNG

Đề tài:

A novel few-shot malware classification approach for unknown family recognition with multi-prototype modeling

---

Phân loại mã độc
Với kỹ thuật few-shot và mô hình đa nguyên mẫu

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn	:	Nghi Hoàng Khoa	
Lớp	:	NT140.O11.ANTT	
Sinh viên thực hiện	:	Nguyễn Thị Hồng Lam	20521518
		Nguyễn Triệu Thiên Bảo	20520653
		Trần Lê Minh Ngọc	21521195
		Huỳnh Minh Khuê	21522240

TP.HCM năm 2023

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO CUỐI KỲ

AN TOÀN MẠNG

Đề tài:

A novel few-shot malware classification approach for unknown family recognition with multi-prototype modeling

---

Phân loại mã độc
Với kỹ thuật few-shot và mô hình đa nguyên mẫu

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn	:	Nghi Hoàng Khoa	
Lớp	:	NT140.O11.ANTT	
Sinh viên thực hiện	:	Nguyễn Thị Hồng Lam	20521518
		Nguyễn Triệu Thiên Bảo	20520653
		Trần Lê Minh Ngọc	21521195
		Huỳnh Minh Khuê	21522240

TP.HCM năm 2023

Lời cảm ơn

Chúng em xin chân thành cảm ơn Khoa Mạng máy tính và Truyền thông, Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em học tập và thực hiện đề tài báo cáo cuối kì này này.

Chúng em xin bày tỏ lòng biết ơn sâu sắc tới thầy Nghi Hoàng Khoa đã tận tình hướng dẫn chỉ bảo chúng em trong quá trình thực hiện đề tài.

Mặc dù đã cố gắng hoàn thành bài báo cáo trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý thầy cô và các bạn.

Chúng em xin chân thành cảm ơn quý thầy cô và các bạn!

Sinh viên

Nguyễn Thị Hồng Lam

Nguyễn Triệu Thiên Bảo

Trần Lê Minh Ngọc

Huỳnh Minh Khuê

Mục lục

I – MỞ ĐẦU

1. Malware là gì?

Malware, viết tắt của "*malicious software*" hay "*phần mềm độc hại*", là một loại phần mềm được tạo ra với mục đích xâm phạm, tấn công hoặc gây hại cho hệ thống máy tính và dữ liệu người sử dụng mà không được sự cho phép của họ. Tính chất độc hại của malware thường xuất phát từ khả năng ẩn nấp, tự động sao chép và lan truyền trong mạng, làm cho chúng trở thành mối đe dọa không ngừng đối với an toàn thông tin. Người tạo ra malware thường sử dụng các chiến lược rafinement và phức tạp, thậm chí thay đổi định dạng của chúng để tránh phát hiện từ phần mềm diệt virus.

Malware là một trong những đe dọa ngày càng phổ biến và nguy hiểm trong thế giới kỹ thuật số ngày nay. Với khả năng gây tổn thương nặng nề đến dữ liệu quan trọng, sự riêng tư, và thậm chí là hoạt động hệ thống, malware đang đặt ra thách thức lớn cho cộng đồng mạng và các chuyên gia bảo mật. Để đối phó hiệu quả với sự đa dạng và nguy hiểm của malware, cần phải có những biện pháp bảo mật mạnh mẽ và sự nhận thức cao về an ninh thông tin từ cả cộng đồng người sử dụng và các chuyên gia kỹ thuật.

Các dạng malware có thể rất đa dạng, bao gồm virus, worm, trojan, ransomware và spyware, mỗi loại mang đến mối đe dọa và tác động khác nhau. Virus và worm thường tự sao chép và lây nhiễm qua các hệ thống, trong khi trojan ẩn mình dưới vỏ ngoài vô hại để xâm nhập vào hệ thống. Ransomware tấn công bằng cách mã hóa dữ liệu và yêu cầu một khoản tiền chuộc để giải mã, trong khi spyware giám sát và thu thập thông tin cá nhân mà không được sự đồng ý của người sử dụng.

2. Mô hình đa nguyên mẫu là gì?

"*Mô hình đa nguyên mẫu*" (multi-agent model) là một hệ thống hoặc mô phỏng được thiết kế để nghiên cứu và mô tả các tương tác giữa nhiều đối tượng tự động độc lập được gọi là "nguyên mẫu" hoặc "agents". Mỗi nguyên mẫu trong mô hình này đại diện cho một thực thể có khả năng tự quyết định và thực hiện hành động dựa trên thông tin và môi trường xung quanh.

Các nguyên mẫu trong mô hình đa nguyên mẫu có thể tương tác với nhau thông qua truyền thông, hợp tác, hoặc thậm chí có thể cạnh tranh để đạt được mục tiêu cụ thể. Mô hình này thường được sử dụng để nghiên cứu và mô phỏng các hệ thống phức tạp, nơi có sự tương tác đa dạng giữa các thực thể tự động có khả năng thay đổi hành vi của mình dựa trên thông tin từ môi trường và các nguyên mẫu khác. Ứng dụng của mô hình đa nguyên mẫu rất đa dạng, từ nghiên cứu trong lĩnh vực trí tuệ nhân tạo và robot học đến

quản lý tài nguyên, giao thông vận tải, và thậm chí là trong lĩnh vực kinh tế học để mô phỏng thị trường và quyết định chiến lược.

3. Ngữ cảnh ứng dụng

Malware ngày càng trở nên đa dạng và phổ biến, đặt ra thách thức lớn cho giới kỹ thuật số trong việc bảo vệ hệ thống khỏi sự xâm hại và hư tổn bởi mối đe dọa này. Mặc dù machine learning được coi là một công cụ mạnh mẽ để phòng tránh malware, nhưng những mô hình chúng có một số nhược điểm chung khi áp dụng với các nhóm mã độc mới phát sinh. Chúng không thể nhận ra các nhóm chưa phát hiện không có trong tập dữ liệu huấn luyện, điều này là do bị ràng buộc bởi các tham số cố định trong các mô-đun phân loại của chúng sau khi đào tạo. Vấn đề này thực chất có hai phần:

Thứ nhất, số lượng mẫu khan hiếm có thể dẫn đến tình trạng overfitting nghiêm trọng cho các mô hình máy học này.

Thứ hai, các mô hình học sâu thường có thể dao động mạnh và không thể đạt đến trạng thái ổn định khi lượng mẫu được cung cấp quá khan hiếm.

Một trong những vấn đề quan trọng là yêu cầu một lượng lớn dữ liệu để huấn luyện mô hình, nhưng việc thu thập thông tin về các loại malware mới thường gặp nhiều khó khăn. Nhằm giải quyết thách thức nêu trên, bài báo này đề xuất một mô hình tiên tiến mang tên "*Simple*" được xây dựng dựa trên phương pháp FewShot Learning và Meta Learning Framework. Few-shot Learning và Meta Learning Framework là gì và chúng được áp dụng như thế nào vào "*Simple*" sẽ được nêu rõ trong các phần sau. Mô hình "*Simple*" đặc biệt linh hoạt trong việc thích ứng với các chủng loại malware mới do nó chỉ cần sử dụng một lượng dataset ít ỏi để có thể phân loại các họ malware.

Với sự trợ giúp của Meta Learning, "*Simple*" được đào tạo với các họ malware được xác định trước và có thể duy trì khả năng phân loại các họ malware mới chưa từng gặp. Ngoài ra, những kiến thức đã học qua Meta Learning có thể ngăn chặn tình trạng bị overfitting do khan hiếm mẫu. "*Simple*" giới thiệu mô hình đa nguyên mẫu để tạo ra nhiều nguyên mẫu của mỗi họ malware nhằm nâng cao khả năng khái quát hóa, dựa trên trình tự gọi API từ phân tích động.

So sánh với mô hình "*IMP*" cơ bản, "*Simple*" dễ triển khai hơn và không yêu cầu thêm thông số. "*Simple*" cũng đạt được độ chính xác tốt hơn tất cả các mô hình cơ sở trong tất cả các cài đặt thử nghiệm và hiển thị hiệu suất phân loại mã độc tiên tiến nhất trong vài lần thử nghiệm. Bằng cách này, "*Simple*" hứa hẹn mang lại một giải pháp linh hoạt và hiệu quả trong việc bảo vệ hệ thống chống lại sự đa dạng ngày càng tăng của malware.

4. Nhiệm vụ của mô hình

Nhiệm vụ chính của mô hình là:

Giải quyết các mẫu mã độc bị khan hiếm và vấn đề nhận dạng động nhóm mã độc một cách mới lạ với sự trợ giúp của Meta Learning. Độ chính xác phân loại đạt được hơn 90% trên các nhóm chưa được phát hiện khi cung cấp một vài mẫu và chỉ cần chuyển tiếp một lần trong quá trình suy luận mà không cần tinh chỉnh tốn thời gian.

Đề xuất một mô hình hoàn toàn mới có tên "*Simple*" để tạo ra nhiều nguyên mẫu bằng cách phân cụm có giám sát. Đầu tiên sử dụng các nguyên mẫu được tạo linh hoạt để xử lý việc phân phối đa phương thức phức tạp trong bộ dữ liệu phần mềm độc hại. Các nguyên mẫu được suy luận một cách thông minh từ dữ liệu đã cho để giảm thiểu overffiting và giảm mức tiêu thụ tài nguyên.

Tiến hành các thử nghiệm rộng rãi để chứng minh "*Simple*" đạt được hiệu suất phân loại mã độc tốt nhất. "*Simple*" vượt trội hơn tất cả các mô hình cơ sở bao gồm cả phương pháp phân loại mã độc động truyền thống và mô hình phân loại Few-shot hiện có, được thừa hưởng từ cả meta-learning và đa nguyên mẫu.

II – KỸ THUẬT ỨNG DỤNG

1. Phân loại mã độc

Các vấn đề về phân loại mã độc có thể được chia thành việc phát hiện mã độc để quyết định mẫu có phải là mã độc hay không và phân loại nhãn mã độc của mẫu.

Phân tích tĩnh và động là hai loại phân tích mã độc và có những ưu điểm và nhược điểm tương ứng. Phân tích tĩnh thường sử dụng chuỗi byte của tệp nhị phân, trình tự tập hợp và mã opcode mà không thực thi các tệp đáng ngờ và xây dựng bộ phân loại học máy dựa trên những tính năng này. Ngược lại, phân tích động thường thực thi nhị phân các tệp trong môi trường ảo hoặc sandbox để thu thập thời gian chạy các tính năng, chẳng hạn như trình tự gọi API hoặc biểu đồ API. Nó có thể phản ánh ý định thực sự của những người được thực thi và tự giải nén mã độc được đóng gói một cách tự nhiên khi thực thi.

2. Few-shot Learning

Few-shot learning là một phương pháp máy học mạnh mẽ, đặc biệt được thiết kế để giải quyết vấn đề của việc huấn luyện mô hình với một lượng nhỏ các mẫu dataset. Một lý do khiến Few-shot Learning rất quan trọng là vì nó làm cho việc phát triển các mô hình học máy trong môi trường thực tế trở nên khả thi. Trong nhiều tình huống thực tế, việc có được một tập dữ liệu lớn mà chúng ta có thể sử dụng để đào tạo mô hình học máy có thể là một thách thức rất lớn. Học trên tập dữ liệu huấn luyện nhỏ hơn có thể giảm đáng kể chi phí và công sức cần thiết để huấn luyện các mô hình học máy. Và Few-shot

Learning có thể thực hiện được điều này vì kỹ thuật này cho phép các mô hình học chỉ từ một lượng nhỏ dữ liệu.

Điều đặc biệt ấn tượng với Few-shot learning là khả năng tự động thích ứng với các tình huống mới và không cần phải được huấn luyện lại từ đầu khi có sự thay đổi trong dữ liệu đầu vào. Thay vì dựa vào lượng lớn dữ liệu, Few-shot learning tận dụng các tri thức đã học được từ các tác vụ trước đó để áp dụng vào những tình huống mới, giảm đáng kể yêu cầu về dữ liệu huấn luyện.

Việc áp dụng Few-shot Learning như thế nào sẽ được miêu tả cụ thể trong phần **III.1.3.Áp dụng Few-shot**.

3. Meta Learning Framework

Meta learning, còn được gọi là "*learning to learn*" nghĩa là "*học cách học*", là một tập hợp con của machine learning trong khoa học máy tính. Nó được sử dụng để cải thiện kết quả và hiệu suất của thuật toán học bằng cách thay đổi một số khía cạnh của thuật toán học dựa trên kết quả thử nghiệm. Các thuật toán Meta Learning sử dụng siêu dữ liệu của các thuật toán học làm đầu vào. Sau đó đưa ra dự đoán và cung cấp thông tin về hiệu suất của các thuật toán học tập này dưới dạng đầu ra.

Trong bài báo này, Meta Learning Framework cho phép "*Simple*" tìm hiểu một số kiến thức cấp cao về malware và xây dựng các bộ phân loại theo nhiệm vụ cụ thể cho các họ malware khác nhau từ một vài mẫu. Nó sử dụng các chuỗi lệnh gọi API làm chuỗi đầu vào, đây là một phương pháp phân tích động hiệu quả được áp dụng trong các nghiên cứu khoa học. Sau đó, các họ malware được thể hiện bằng nhiều nguyên mẫu từ việc phân cụm thay vì một nguyên mẫu duy nhất được áp dụng trong các mô hình Few-shot cơ bản khác. Ngoài ra, số lượng cụm không cần phải xác định trước và dung lượng mô hình có thể được suy ra một cách thích ứng từ dữ liệu tác vụ để cân bằng cả phân phối dữ liệu đơn giản và phức tạp, được gọi là nguyên mẫu hỗn hợp vô hạn (*IMP* - Infinite Mixture Prototypes).

4. IMP

IMP - Infinite Mixture Prototypes, được trích xuất từ bài báo "*Supplement to Infinite Mixture Prototypes for Few-Shot Learning*", là một thuật toán phân cụm hoặc phân loại dữ liệu không có nhãn. Thuật toán này tập trung vào việc tạo và cập nhật các nguyên mẫu không giới hạn để biểu diễn các nhóm dữ liệu. IMP không yêu cầu số lượng mẫu lớn và linh hoạt trong việc xử lý dữ liệu đa dạng và phức tạp.

Mục đích sử dụng để sử dụng IMP là: phân cụm dữ liệu không có nhãn, phân loại dữ liệu, xử lý dữ liệu không đồng nhất và phát triển mô hình tự động.

Áp dụng thuật toán IMP trong Few-shot Learning:

Biểu diễn dữ liệu: IMP có thể tạo ra các nguyên mẫu để biểu diễn sự đa dạng của dữ liệu trong các nhóm hoặc lớp dữ liệu. Trong few-shot learning, việc biểu diễn này có thể giúp mô hình hiểu được cách các điểm dữ liệu phân tán trong không gian đa chiều.

Phân loại: Nếu có ít mẫu huấn luyện, IMP có thể được sử dụng để phân loại các điểm dữ liệu mới vào các nhóm tương ứng dựa trên các nguyên mẫu đã được tạo ra trước đó. Điều này có thể giúp mô hình phân loại dữ liệu mới dựa trên sự tương đồng với các nguyên mẫu đã biết.

Phát triển mô hình linh hoạt: IMP có khả năng linh hoạt trong việc tạo ra số lượng nguyên mẫu không giới hạn, giúp mô hình có thể thích ứng với sự đa dạng của dữ liệu mẫu huấn luyện ít.

III – PHÂN TÍCH HỆ THỐNG

1. Tập dữ liệu

1.1. Giới thiệu tập dữ liệu

1.1.1. VirusShare_00177

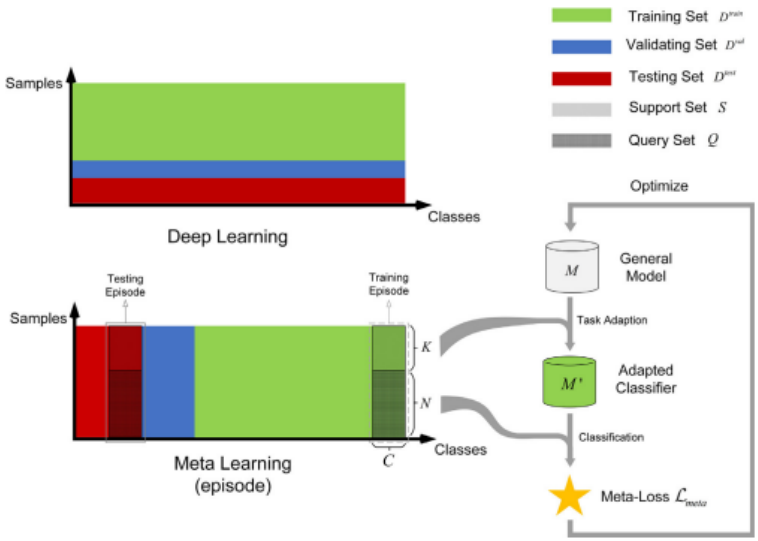
Bộ dữ liệu này là một phần lớn của phần mềm độc hại Windows PE được tải từ trang web VirusShare với số thứ tự 177, bao gồm 65,535 mẫu và được công bố để tải xuống vào ngày 4 tháng 10 năm 2015. Vì bộ dữ liệu này hoàn toàn không có nhãn, chúng ta mượn các quy trình gán nhãn từ một công cụ khác, trong đó chúng ta sử dụng VirusTotal để quét và tạo báo cáo cho mỗi mẫu trong bộ dữ liệu. VirusTotal sẽ báo cáo các giá trị hash như MD5 và quét mỗi mẫu với nhiều động cơ chống virus (AV) để tạo ra kết quả quét trong một tệp JSON.

1.1.2. APIMDS

APIMDS là một bộ dữ liệu công khai về việc gọi API của phần mềm độc hại, được đề xuất lần đầu năm 2015 và đã trở thành một bộ dữ liệu thử nghiệm tiêu chuẩn để thực hiện các thí nghiệm phân loại malware. Nó bao gồm 23,146 chuỗi gọi API của malware được ghi lại bởi thư viện hook, nhưng một số nhãn gia đình của chuỗi bị thiếu. Do đó, chúng ta đặt lại nhãn cho bộ dữ liệu này theo cùng cách như VirusShare_00177.

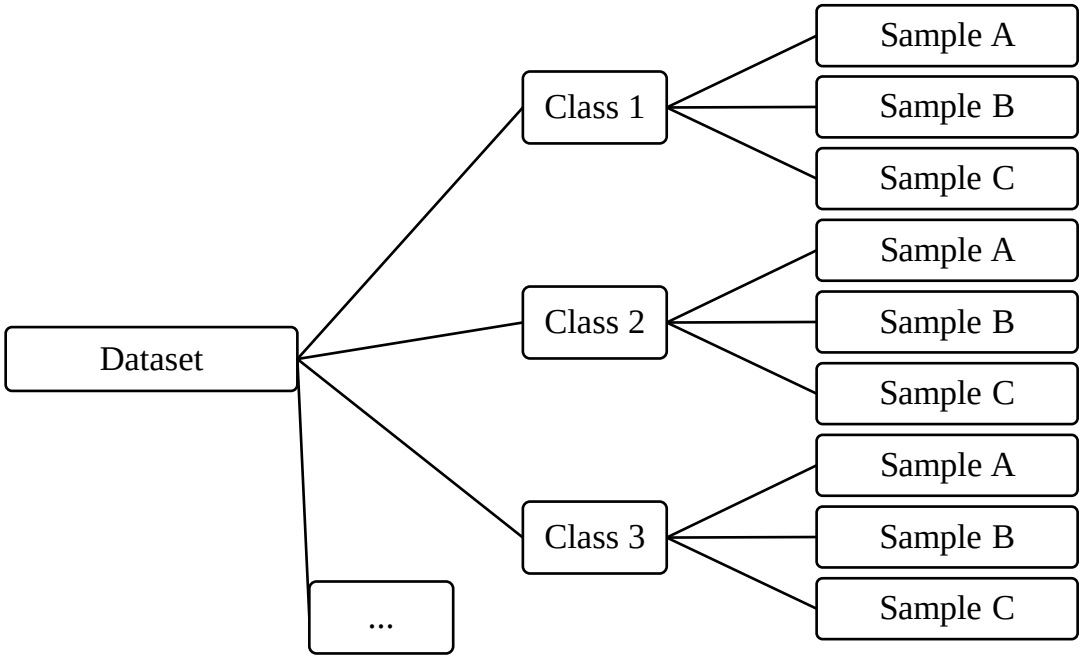
1.2. Phân chia dữ liệu

Trong mô hình "*Simple*", chúng ta sẽ chia tập dữ liệu theo chiều dọc thay vì chiều ngang như các mô hình truyền thống. Mỗi tập dữ liệu sẽ được chia thành ba phần gồm dữ liệu training, dữ liệu testing và dữ liệu validate.



Hình 1: Phân chia dữ liệu

Để có thể dễ hình dung, chúng ta sẽ xem ví dụ minh họa sau đây (class 1, class2, class 3 là các họ mã độc và sample A, sample B, sample C là các mẫu mã độc):



Khi phân chia theo chiều ngang như các mô hình truyền thống, mỗi sample trong mỗi class sẽ được lấy ra để chia thành các tập dữ liệu. Ví dụ, sample A của class 1, sample A của class 2, sample A của class 3 sẽ là tập dữ liệu training; sample B của class 1, sample B của class 2, sample B của class 3 là tập dữ liệu testing; sample C của class 1, sample

C của class 2, sample C của class 3 là tập dữ liệu training. Như vậy để có thể huấn luyện mô hình, mô hình phải quét qua hết tất cả các class để có thể lấy ra các sample của mỗi class. Và điều này hoàn toàn đi ngược lại với ý tưởng ban đầu rằng mô hình không được biết các class malware khác ngoài dữ liệu training.

Vì thế, mô hình "*Simple*" sẽ phân chia tập dữ liệu theo chiều dọc, thay vì lấy mỗi sample trong mỗi class để chia thành các tập dữ liệu, chúng ta sẽ lấy hoàn toàn một class thành một tập dữ liệu. Ví dụ, class 1 sẽ là tập dữ liệu training, class 2 là tập dữ liệu testing và class 3 là tập dữ liệu validate. Như vậy, khi mô hình được huấn luyện (training), nó chỉ được phép biết được mỗi class 1 mà không được phép biết các class 2 và class 3.

1.3. Áp dụng Few-shot

Mục tiêu của việc phân loại malware trong Few-shot là sử dụng một số ít mẫu malware được gán nhãn (được định nghĩa là tập hỗ trợ S) để dự đoán các họ của các mẫu malware chưa được gán nhãn (được định nghĩa là tập truy vấn Q). Khi tập hỗ trợ S có tổng cộng C họ và mỗi họ có K mẫu, nhiệm vụ phân loại Few-shot này được gọi là một nhiệm vụ C-way K-shot. Thông thường, K nhỏ (ví dụ: K = 1, 5, 10...) để huấn luyện một bộ phân loại với hàng nghìn tham số, được biết đến là "few-shot". Lưu ý rằng chỉ có tập truy vấn có nhãn các họ malware trong giai đoạn huấn luyện, không có trong giai đoạn kiểm thử. Khi được đưa ra một nhiệm vụ phân loại mới để phân loại các gia đình malware mới, một bộ phân loại mới phải được xây dựng bằng cách sử dụng một tập hỗ trợ mới để phân loại một tập truy vấn mới.

2. Thuật toán Meta Learning

Để vượt qua vấn đề thiếu mẫu khi kiểm thử, mô hình của chúng ta phải học được một số kiến thức cấp cao có thể chuyển giao từ tập huấn luyện, điều này được thực hiện thông qua Meta Learning (học để học).Cụ thể, C malware family (không gian nhãn) được chọn ngẫu nhiên từ tập huấn luyện D^{train} và sau đó K×C và N×C mẫu được lấy mẫu từ các family đã chọn để tạo thành tập hỗ trợ S và tập truy vấn Q. Tập hỗ trợ có nhãn S được sử dụng để mô hình thích ứng với một nhiệm vụ mục tiêu (thực sự là tạo ra các prototypes trong "*Simple*"). Sau đó, mô hình đã được thích ứng cố gắng dự đoán các nhãn của tập truy vấn Q (thực sự là so sánh với các prototypes được tạo ra trong "*Simple*"), dựa trên tập hỗ trợ S và giá trị mất mát có thể được tính từ dự đoán. Do C malware family được chọn có thể đa dạng trong mỗi tập, mô hình buộc phải học được kiến thức có thể chuyển giao trong quá trình đào tạo, điều này có lợi cho việc nhận diện gia đình malware mới. Chi tiết thuật toán được mô tả trong "Thuật toán 1".

Algorithm 1: Episode training strategy

Input: Training set D^{train}, loss function $\mathcal{L}$, maximum training epoch T, task parameter C, K and N

Output: Trained model M

Initialize the model M with parameter ϕ;

for $i \leftarrow 1$ **to** T **do**

  Randomly sample C families from D^{train} to form task label space L;

  Randomly sample K and N samples from each family in L to formthe support set $S = \{(x_i^S, y_i^S)\}_{i=0}^{K \times C}$ and the query set$Q = \{(x_i^Q, y_i^Q)\}_{i=0}^{N \times C}$ respectively ;

  Feed the support set S to the model M to adapt the new task and outputtailored task classifier: $M' \leftarrow M(\phi, S)$;

  Use the task classifier M' to classify query set Q and produce theclassification loss (meta loss): $\mathcal{L}_{meta} \leftarrow \sum_{i=1}^{N \times C} \mathcal{L}(M'(x_i^Q | \phi, S), y_i^Q)$;

  Derive gradients of ϕ from meta loss $\mathcal{L}_{meta}$ and optimizeϕ using SGD optimizer;

end

return M

Hình 2: Thuật toán 1

3. Hiệu suất ứng dụng

3.1. Mô hình cơ sở

Giới thiệu sơ lược về các mô hình được sử dụng để so sánh với "*Simple*":

MANNWARE: Đây là mô hình phân loại mã độc few-shot đầu tiên được đề xuất bởi Tran và đồng nghiệp, sử dụng mạng thần ký bổ trí bộ nhớ (memory-augmented neural network) làm bộ phân loại.

InductionNet: Mô hình này sử dụng thuật toán định tuyến động để tạo ra các nguyên mẫu (prototypes) của các lớp và đo độ tương tự giữa một mẫu được truy vấn và một nguyên mẫu thông qua một lớp mạng neural tensor.

HybridAttentionNet: Mô hình này sử dụng một mô-đun chú ý theo mẫu để bắt lấy mối quan hệ giữa các mẫu lớp và một mô-đun chú ý đặc điểm để đánh trọng số cho các chiều của các vectơ đặc trưng.

ATAML: Đây là một biến thể của mô hình MAML (Model-Agnostic Meta-Learning) nhanh chóng chia riêng thông số trích xuất đặc trưng không phụ thuộc vào nhiệm vụ và thông số chú ý đối tượng cụ thể cho từng nhiệm vụ.

PerLayer ATAML: Đây là phiên bản cải tiến của ATAML, bổ sung tỷ lệ học từng tầng (layer-wise learning rate) trong từng vòng thích nghi để kiểm soát chính xác độ dài bước.

FEAT: Sử dụng một hàm từ tập hợp đến tập hợp để bắt lấy thông tin cụ thể về nhiệm vụ trong các mẫu một cách không thứ tự để thích nghi với nhiệm vụ mục tiêu.

IMP: Mô hình này mô hình hoá phân phối đa dạng trong tập dữ liệu bằng cách sử dụng gom cụm DP-means và nhiều nguyên mẫu cùng tồn tại để đại diện cho một phân phối lớp phức tạp.

3.2. Thước đo đánh giá

Tuân theo quy ước trong phân loại malware, chúng ta sử dụng bốn loại chỉ số để đánh giá hiệu suất của mô hình: accuracy (độ chính xác), precision (độ chính xác cao), recall (độ nhớ) và F1-score. Các chỉ số được quy ước như sau:

$$Accuracy = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

$$Precision = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i}$$

$$Recall = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i}$$

$$F_1 - Score = \frac{1}{C}\sum_{i=1}^{C}\frac{2 * Precision(i) * Recall(i)}{Precision(i) + Recall(i)}$$

Table 2 – Hyper-parameters in typical experiments.

Type	Name	Value
Common Parameters	TF-IDF Items l	2000
	Maximum Sequence Length t	200
	N-gram Window Size	3
Few-Shot Model Parameters	GloVe Dimension	300
	GloVe Learning Rate	0.05
	GloVe Training Epoch	20
	BiLSTM Hidden Dimension	128
	BiLSTM Layer	1
	Dropout Ratio	0.5
	Temporal Convolution Kernel Size	3
	Temporal Convolution Padding Size	1
SIMPLE Parameters	Initial Variance σ	1
	Maximum Clustering Iteration T_c	1

Hình 3: Các siêu tham số trong thí nghiệm

3.3. So sánh với mô hình cơ sở

Để chứng minh rằng "*Simple*" đạt được hiệu suất phân loại mã độc few-shot hàng đầu, chúng ta so sánh nó với các mô hình cơ sở trong phần trên. Vì sự đơn giản, chúng ta chỉ đo độ chính xác (accuracy) khi thực hiện các thử nghiệm so sánh để quyết định hiệu suất của mỗi mô hình.

Table 3 – Testing accuracy of all models on VirusShare_00177 and APIMDS datasets in few-shot malware classification. 95% confidence interval are attached after the mean accuracy and the best accuracy of each column is in bold.

Model	VirusShare_00177				APIMDS			
	5-way		10-way		5-way		10-way	
	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
Traditional Malware Classification Methods								
API Sequence Histogram (Ndibanje et al., 2019)	86.59 ± 0.31	89.90 ± 0.26	81.51 ± 0.26	85.96 ± 0.23	75.04 ± 0.37	80.78 ± 0.36	68.91 ± 0.29	75.46 ± 0.28
API Sequence Alignment (Cho et al., 2014)	79.43 ± 0.77	81.10 ± 0.77	72.37 ± 0.55	74.11 ± 0.62	59.40 ± 0.86	62.00 ± 0.83	48.15 ± 0.63	50.14 ± 0.63
API Markov Chain (Amer and Zelinka, 2020)	64.12 ± 0.45	66.61 ± 0.44	61.87 ± 0.32	63.82 ± 0.30	55.12 ± 0.41	58.33 ± 0.42	46.32 ± 0.30	49.37 ± 0.30
Few-Shot Models								
MANNWARE (Tran et al., 2020)	–	-	–	–	75.66	78.85	–	–
InductionNet (Geng et al., 2020)	79.43 ± 0.28	80.33 ± 0.28	75.11 ± 0.18	73.23 ± 0.19	73.04 ± 0.34	74.18 ± 0.32	66.09 ± 0.21	67.08 ± 0.21
HybridAttentionNet (Gao et al., 2019)	89.73 ± 0.21	87.88 ± 0.23	81.45 ± 0.18	80.35 ± 0.18	77.53 ± 0.32	79.90 ± 0.33	64.11 ± 0.25	68.37 ± 0.23
ATAML (Abbasi et al., 2019)	85.98 ± 0.24	88.13 ± 0.20	79.89 ± 0.19	83.28 ± 0.17	77.84 ± 0.30	80.74 ± 0.29	63.09 ± 0.23	68.90 ± 0.25
PerLayer ATAML (Antoniou et al., 2019)	86.01 ± 0.19	87.92 ± 0.21	80.21 ± 0.17	84.67 ± 0.14	76.35 ± 0.30	78.63 ± 0.31	63.22 ± 0.22	69.91 ± 0.21
FEAT (Ye et al., 2020)	91.12 ± 0.16	91.56 ± 0.17	87.65 ± 0.13	90.07 ± 0.12	84.89 ± 0.26	86.31 ± 0.24	78.27 ± 0.20	80.44 ± 0.20
IMP (Allen et al., 2019)	91.33 ± 0.17	93.50 ± 0.14	88.21 ± 0.13	90.08 ± 0.13	84.77 ± 0.26	87.68 ± 0.23	78.48 ± 0.19	82.30 ± 0.18
SIMPLE(ours)	**92.35 ± 0.20**	**94.15 ± 0.17**	**89.15 ± 0.12**	**91.20 ± 0.13**	**86.13 ± 0.24**	**89.22 ± 0.21**	**80.04 ± 0.18**	**84.21 ± 0.16**

Hình 4: Bảng so sánh độ chính xác với các mô hình cơ sở

Mặc dù một số mô hình cơ sở mạnh mẽ như IMP có độ chính xác rất cao, "*Simple*" vẫn vượt trội hơn tất cả các mô hình cơ sở trên cả hai tập dữ liệu và đạt được độ chính xác 90%. Một số mô hình thích nghi nhanh trong các mô hình cơ sở, như ATAML và PerLayer ATAML, kém hơn mô hình "*Simple*" đề xuất hơn 7% trên tập dữ liệu VirusShare_00177, và khoảng cách này còn lớn hơn trên tập dữ liệu APIMDS. Hơn nữa, các mô hình thích nghi nhanh cần tính toán đạo hàm bậc cao trong quá trình tối ưu hóa, trong đó một số phép tính phức tạp như mô hình tuần tự có thể không được hỗ trợ trên các nền tảng hiện có. Đối với các mô hình dựa trên đo đoạn (metric-based) như "*Simple*",

loại vấn đề này không gây phiền vì các trình tối ưu thông thường là đủ cho quá trình đào tạo.

Hầu hết các mô hình đều có độ chính xác thấp hơn trên tập dữ liệu APIMDS so với tập dữ liệu VirusShare_00177 vì APIMDS khó hơn và gần gũi hơn với môi trường sản xuất so với VirusShare_00177. Đối với các phương pháp phân loại mã độc truyền thống, sự khác biệt về hiệu suất giữa "*Simple*" và chúng trở nên rõ rệt hơn trên tập dữ liệu APIMDS hơn so với VirusShare_00177, với sự vượt trội lên đến hơn 10%. Điều này chứng tỏ rằng "*Simple*" có tính linh hoạt và hiệu quả hơn khi xử lý chuỗi API.

3.4. Tác động của Meta-Learning

Để nghiên cứu tác động của Meta-learning, chúng ta sẽ so sánh "*Simple*" với hai mô hình khác được huấn luyện mà không sử dụng Meta-learning trong các thử nghiệm phân loại malware dựa trên lượng dữ liệu ít ỏi.

Chúng ta sẽ so sánh "*Simple*" và hai mô hình không sử dụng Meta-learning là fine-tunning và SIMPLE#. Quá trình fine-tuning sử dụng một bộ phân loại tuyến tính để phân loại các chuỗi đã nhúng vào malware family dự đoán sau khi các tham số được cập nhật qua vài vòng lặp, dựa trên mất mát chuyển tiếp của bộ dữ liệu hỗ trợ sở hữu một vài mẫu. SIMPLE# biểu thị mô hình SIMPLE mà không sử dụng Meta-learning.

Kết quả so sánh được hiển thị trong bảng dưới đây:

Table 4 – Experimental results of investigation of meta learning effect in few-shot malware classification. SIMPLE# stands for the SIMPLE model without applying meta learning. Best metric value among all the models under the same settings is in bold.

Metric	Model	VirusShare_00177				APIMDS			
		5-way		10-way		5-way		10-way	
		5-shot	10-shot	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
Accuracy	Fine-tuning	83.35	85.01	79.28	80.97	43.32	42.63	36.93	35.62
	SIMPLE#	87.30	90.44	83.03	86.16	74.99	77.66	67.73	68.23
	SIMPLE	**92.35**	**94.15**	**89.15**	**91.20**	**86.13**	**89.22**	**80.04**	**84.21**
Precision	Fine-tuning	84.30	85.52	79.92	81.29	33.22	31.8	30.54	28.58
	SIMPLE#	89.19	91.90	85.35	88.28	77.13	80.09	70.03	70.34
	SIMPLE	**93.80**	**95.08**	**90.81**	**92.25**	**87.47**	**90.37**	**81.55**	**85.76**
Recall	Fine-tuning	83.35	85.01	79.28	80.97	43.32	42.63	36.93	35.62
	SIMPLE#	87.3	90.44	83.03	86.16	74.99	77.66	67.73	68.23
	SIMPLE	**92.35**	**94.15**	**89.15**	**91.21**	**86.12**	**89.22**	**80.04**	**84.21**
F1-score	Fine-tuning	81.49	83.21	76.92	78.65	33.68	32.69	29.65	28.07
	SIMPLE#	86.56	90.10	82.18	85.59	73.71	76.78	66.34	66.88
	SIMPLE	**92.02**	**93.79**	**88.68**	**90.86**	**85.84**	**89.07**	**79.56**	**83.98**
Testing Loss	Fine-tuning	0.972	0.790	0.850	0.801	4.047	4.136	2.549	2.606
	SIMPLE#	1.579	1.576	2.267	2.266	1.6	1.601	2.293	2.293
	SIMPLE	**0.299**	**0.257**	**0.521**	**0.420**	**0.449**	**0.386**	**0.712**	**0.618**
Testing Time (ms)	Fine-tuning	923.22	1026.57	1021.88	1156.70	899.76	1150.63	1164.08	1438.59
	SIMPLE#	40.90	60.30	88.33	129.77	**41.39**	66.67	81.87	113.55
	SIMPLE	**37.83**	**59.02**	**74.49**	**105.86**	42.40	**58.68**	**75.62**	**106.26**

Hình 5: Bảng so sánh tác động của Meta-learning

Từ bảng trên, chúng ta có thể thấy rằng fine-tuning hoạt động khá tốt trên tập dữ liệu VirusShare_00177, nhưng gặp vấn đề trên tập dữ liệu APIMDS. Điều này xảy ra vì trên tập dữ liệu VirusShare_00177 các chuỗi trong mỗi họ gia đình rất giống nhau. Tuy nhiên, trên tập dữ liệu APIMDS, việc nâng cao hiểu biết ngữ cảnh là quan trọng hơn, và fine-tuning không thể thu được đủ thông tin hữu ích từ tập hỗ trợ, dẫn đến hiệu suất kém. SIMPLE# chưa được huấn luyện có hiệu suất kém hơn so với "*Simple*" nhưng hiệu suất vẫn cao hơn so với fine-tuning. Điều này chứng minh rằng "*Simple*" thực sự có thể học được một số kiến thức chung ở mức cao qua các nhiệm vụ và Meta-learning là khả thi trong phân loại malware.

3.5. Tác động của mô hình đa nguyên mẫu

Để chứng minh thêm rằng "*Simple*" thực sự có thể tự động điều chỉnh khả năng mô hình để phù hợp tốt nhất với phân phối dữ liệu, chúng tôi so sánh nó với hai mô hình cơ sở: NN và UM.

UM giả định rằng có một nguyên mẫu đại diện duy nhất cho mỗi gia đình và việc sử dụng nguyên mẫu gia đình để phân loại các mẫu được truy vấn thay vì điểm dữ liệu thô có thể hiệu quả và mạnh mẽ hơn. NN xử lý tất cả các mẫu trong tập hỗ trợ như là nguyên mẫu và gán nhãn cục bộ của điểm dữ liệu gần nhất cho mẫu kiểm tra. Nó có thể mô hình hóa phân phối dữ liệu phức tạp hơn so với unimodal, nhưng nhạy cảm hơn đối với nhiễu trong dữ liệu và tiêu thụ bộ nhớ nhiều hơn.

Kết quả so sánh được hiển thị trong bảng dưới đây:

Table 5 – Experimental results of three prototype-based models owning different model capacity in few-shot malware classification scenario. Bold items indicate the highest metric value among all the models under the same settings.

Metric	Model	VirusShare_00177				APIMDS			
		5-way		10-way		5-way		10-way	
		5-shot	10-shot	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
Accuracy	UM	89.14	91.01	87.33	89.14	83.10	85.28	76.37	80.31
	NN	91.38	93.67	88.70	90.92	84.19	87.82	78.59	83.14
	SIMPLE	**92.35**	**94.15**	**89.15**	**91.20**	**86.13**	**89.22**	**80.04**	**84.21**
Precision	UM	91.27	92.90	89.80	91.31	85.36	87.16	79.11	82.19
	NN	92.87	94.72	90.22	92.33	85.82	89.18	80.20	83.75
	SIMPLE	**93.80**	**95.08**	**90.81**	**92.52**	**87.47**	**90.37**	**81.55**	**85.76**
Recall	UM	89.14	91.01	87.33	89.14	83.10	85.28	76.37	80.31
	NN	91.38	93.67	88.70	90.92	84.19	87.82	78.59	83.14
	SIMPLE	**92.35**	**94.15**	**89.15**	**91.21**	**86.12**	**89.22**	**80.04**	**84.21**
F1-score	UM	88.61	90.61	86.86	88.75	82.45	84.81	75.68	79.95
	NN	90.95	93.43	88.13	90.54	83.31	87.24	77.56	82.26
	SIMPLE	**92.02**	**93.79**	**88.68**	**90.86**	**85.84**	**89.07**	**79.56**	**83.98**

Hình 6: Bảng so sánh tác động của mô hình đa nguyên mẫu

Có thể thấy "*Simple*" vượt trội hơn so với cả hai mô hình cơ sở này, do đó có thể suy luận rằng quá ít prototypes (UM) không thể nhận diện được phân phối đa dạng và quá

nhiều prototypes (NN) đôi khi sẽ làm lạc hướng các mẫu được truy vấn, đây đều là nguyên nhân của độ chính xác tương đối thấp.

Ngược lại, "*Simple*" có thể thông minh suy luận số lượng prototypes thích hợp từ các mẫu theo cách "thử-nghiệm". Bằng cách kiểm tra trước kết quả phân loại bằng cách sử dụng các prototypes hiện tại, các prototypes không cần thiết sẽ được bỏ qua trước khi được thêm vào tập prototypes, và điều này ngăn chặn việc tạo ra quá nhiều prototypes.

4. Công việc tương lai

Mặc dù "*Simple*" đã gặt hái được những thành quả cao nhưng vẫn có một số thách thức cần giải quyết.

Ví dụ, tác giả quan sát thấy rằng một số mẫu malware từ các họ khác nhau có các chuỗi gọi API giống nhau và chúng không thể được xác định bằng mô-đun nhúng chỉ bằng cách sử dụng chuỗi làm đầu vào. Ngoài ra, trong mô hình phân loại malware Few-shot tổng quát, tập kiểm thử chứa một số họ cơ bản, chưa được kiểm tra trong các thí nghiệm, vì vậy nó sẽ là một thách thức thách thức trong các ứng dụng ngành công nghiệp.

Do đó, tác giả có đề xuất một số công việc tương lai để hỗ trợ phân loại malware Few-shot. Đầu tiên là tăng kích thước của tập dữ liệu. Chỉ có 20 mẫu cho mỗi họ malware trong các bộ dữ liệu và điều này nghiêm trọng hạn chế khả năng tổng quát của mô hình. Đối với Few-shot Learning đặc biệt với Meta Learning, có nhiều mẫu hơn trong một họ có nghĩa là có nhiều kiến thức tiên định được đưa ra bởi các họ cơ bản. Ngoài ra, số lượng họ malware nên được tăng lên để tạo ra nhiều kết hợp nhiệm vụ có thể. Tóm lại, việc tăng cường lượng thông tin của dữ liệu đầu vào là có lợi.

IV – TRIỂN KHAI ỨNG DỤNG

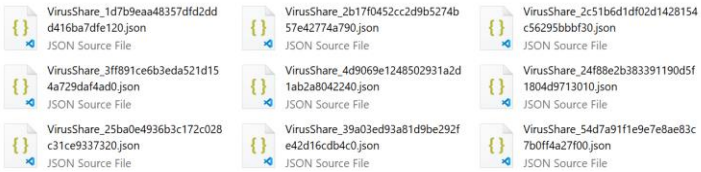
1. Xử lý dữ liệu

1.1. Trích xuất chuỗi API

Để theo dõi và ghi lại trình tự gọi API, bài báo sử dụng Cuckoo sandbox2 để thực thi phần mềm độc hại trong môi trường ảo. Sandbox Cuckoo được thực thi trên host machine và quản lý các guest machine để phân tích mã độc.Mô-đun báo cáo tùy chỉnh được viết bằng Python được sử dụng sau khi phân tích để tạo báo cáo JSON chỉ chứa các chuỗi lệnh gọi API để giảm mức tiêu thụ dung lượng ổ đĩa và cài đặt Cuckoo mặc định chung đang chạy được thông qua trong quá trình phân tích. Để phân tích báo cáo do Cuckoo tạo, cbài báo chỉ để lại tên API trong khi bỏ các tham số và giá trị trả về của mỗi lệnh gọi API. Đối với mã độc khởi chạy một số quy trình, API sẽ được gọi bởi các quy trình khác nhau được nối theo thứ tự tạo của chúng để làm cho chuỗi API trở nên tuyến tính.

Lưu ý rằng chúng tôi loại bỏ các mẫu mã độc có độ dài chuỗi nhỏ hơn 10, vì chuỗi quá ngắn thường cho thấy việc thực thi không thành công.

Dataset được thực thi trong ứng dụng sẽ là các file JSON.



Hình 7: Các file dữ liệu thực thi

1.2. Tiền xử lý API và nhúng từ

Việc tiền xử lý để loại bỏ các đặc trưng dư thừa và chỉ giữ lại những đặc trưng quan trọng nhất sau khi có được chuỗi API là rất quan trọng. Khi phân tích được thực hiện, thông thường một chương trình sẽ liên tục gọi một số API giống nhau khi thực hiện công việc liên quan đến tệp hoặc thực hiện vòng lặp, và những chuỗi gọi liên tục này có thể được coi là dư thừa. Do đó, nhóm tác giả đã thay thế những chuỗi API dài nơi một API giống nhau xuất hiện hơn hai lần liên tiếp bằng một chuỗi con ngắn với hai lời gọi. Họ cũng áp dụng phương pháp N-gram từ các chuỗi API nguyên thủy để tăng cường biểu diễn đặc trưng cục bộ. Các chuỗi N-gram API nguyên thủy thường được nhúng bằng ma trận nhúng và chuỗi vector đầu ra, được biết đến là giai đoạn nhúng.

2. Demo ứng dụng

Demo được nhóm tác giả chia sẻ trên github:

https://github.com/Asichurter/APISeqFewShot.git

Dataset được xử lý sẵn và public ở link:

https://drive.google.com/file/d/1F5lTzn5_ubms288KPIImzb2dyBCS5ZB8/view?usp=sharing

Cấu hình máy tính triển khai:

OS: Windows 11

CPU: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz, 2688 Mhz, 6 Core(s), 12 Logical Processor(s)

GPU: RTX 3050 1.5GHz, 16 cores

Memory: 16GB

Cuda: 12.3

Dataset bao gồm 3 tập D^{train}, D^{val}, D^{test} như mô tả trong bài báo.

Table A.6 – Detailed dataset splitting result for VirusShare_00177.

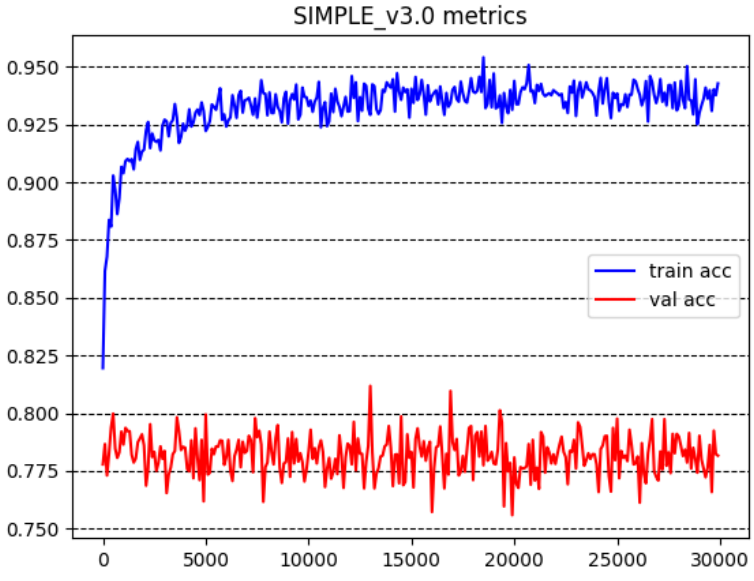
Subset	Owned family name	Count
D^{train}	directdownloader,hiloti,sytro,downloadadmin,softcnapp,buterat, kykymber,zbot,psyme,badur,zapchast,toggle,simbot,qqpass,outbrowse, blacole,soft32downloader,qhost,loring,softonic,webprefix,getnow,startp, lunam,ibryte,egroupdial,darkkomet,pirminay,reconyc,lineage,dealply, darbyen,cpllnk,ipamor,antavmu,kovter,linkular,jeefo,banload,firseria, hidelink,scrinject,vilsel,downloadassistant,installmonetizer,sefnit, instally,autoit,staser,extenbro,urelas,installerex,zeroaccess,airinstaller, jyfi,gator,goredir,icloader,lipler,bundlore,ircbot,nimda,gepys,inor,iframeref, wonka,urausy,bettersurf,adclicer,refresh,domaiq,fsysna,c99shell	73
D^{val}	xtrat,faceliker,somoto,dlhelper,fujacks,includer,1clickdownload,trymedia, patchload,acda,scarsi,windef,shipup,decdec,loadmoney,microfake,msposer, refroso,yoddos,redir	20
D^{test}	llac,fearso,kido,zvuzona,xorer,fakeie,hicrazyk,iframeinject,fosniw, downloadsponsor,vittalia,hijacker,mydoom,fbjack,wabot,4shared,mikey, black,zzinfor,midia	20

Hình 8: Tập dữ liệu

3. Kết quả triển khai

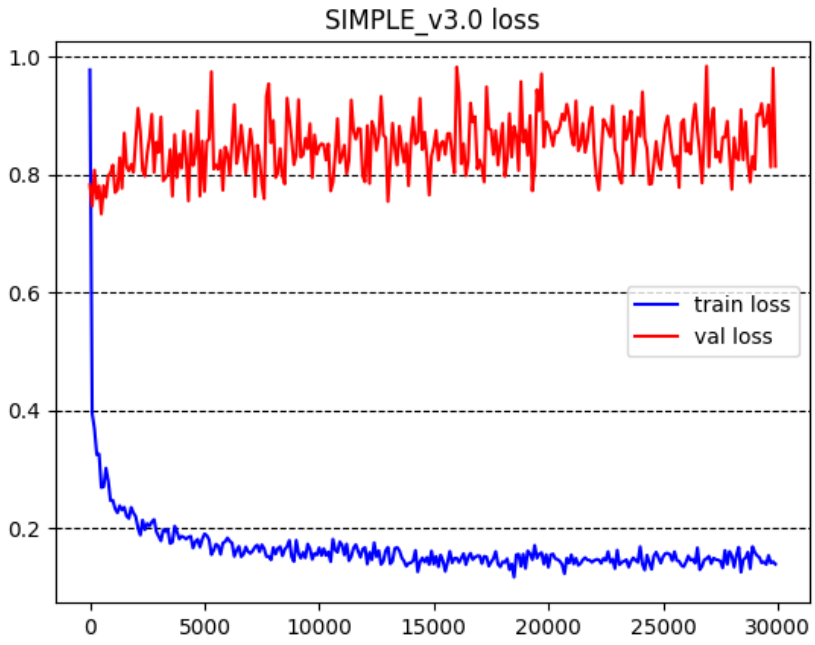
Mô hình chạy trong 30000 epoch và đánh giá theo 2 tiêu chí: accuracy và loss.

Kết quả training cho thấy mô hình SIMPLE cho kết quả rất tốt với tập Dtrain với mức accuracy trong khoảng từ 92,5 – 95%. Tuy nhiên đối với tập Dval, mô hình cho kết quả thấp hơn so với trong bài báo (mức accuracy trung bình trong khoảng 77,5 – 80%).



Hình 9: Kết quả accuracy đo được

Đối với kết quả loss, mô hình cho mức loss khá thấp với tập Dtrain (khoảng 10%). Tuy nhiên, mức loss lại khá cao đối với tập Dval (trung bình trong khoảng từ 80 – 90%).



Hình 10: Kết quả loss đo được

Đây là hiện tượng overfit, khá phổ biến với các mô hình áp dụng Fewshot learning. Nguyên nhân có thể là do các thư viện hỗ trợ của bài báo đã cũ và không còn hỗ trợ một số tính năng, hoặc một số hypermeter được cài đặt mà tác giả không đề cập trong bài (Trong quá trình chạy demo, thư viện Tensorflow có báo lỗi không tìm thấy đủ hypermeter nên đã áp dụng hypermeter mặc định của thư viện).

V – Hướng phát triển

"*Simple*" cho kết quả phân tích rất tốt và là một model có tiềm năng. Tuy nhiên vẫn còn vài vấn đề cần được cải tiến. Nếu các mẫu malware thuộc các họ khác nhau nhưng lại có chuỗi gọi API giống nhau thì phương pháp phân tích API trong bài chưa thể phân biệt được các mẫu này. Thêm vào đó, "*Simple*" chưa được kiểm chứng bằng các bài test được thiết kế trong môi trường lab mà chỉ được kiểm chứng thông qua các ngữ cảnh thực tế trong các ứng dụng thương mại. Một điểm nữa để cải thiện là về dataset, hiện tại mỗi họ malware chỉ có 20 mẫu, nếu có thể tăng số lượng mẫu trong mỗi họ thì kết quả sẽ được cải thiện. Ngoài ra việc tăng số lượng họ malware cũng có thể làm kết quả chính xác

hơn. Cả hai điều này đều có thể thực hiện được bằng cách bổ sung thêm mẫu thử vào dataset.

Khi dataset được mở rộng thì việc có một mô hình khác để xử lí dữ liệu là cần thiết. Bài báo đề xuất mô hình BERT hoặc biến thể của nó để xử lí vấn đề này. BERT là một mô hình dùng để phân tích ngôn ngữ tự nhiên nên có thể trích xuất thông tin trong 1 chuỗi tốt hơn.

Cuối cùng, mô hình FSL nói chung rất dễ bị tình trạng overfit. Để khắc phục, có thể tăng số lượng thông tin cần phân tích trong chuỗi đầu vào. Một cách được bài báo đề xuất là thêm các tham số trong quá trình phân tích API vào phân input.

Tài liệu tham khảo

- Wang, Peng & Zhijie, Tang & Wang, Junfeng. (2021). A Novel Few-Shot Malware Classification Approach for Unknown Family Recognition with Multi-Prototype Modeling. Computers & Security. 106. 102273. 10.1016/j.cose.2021.102273.
- Allen, K., Shelhamer, E., Shin, H. & Tenenbaum, J.. (2019). Infinite Mixture Prototypes for Few-shot Learning. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research. 97:232-241
- https://github.com/Asichurter/APISeqFewShot.git