BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

**Môn học: Cơ chế hoạt động của mã độc**

**Tên chủ đề: Similarity hash based scoring of portable executable files for efficient malware detection in IoT**

*Mã nhóm: G13 - Mã đề tài: S22*

**Lớp**: **NT230.N21.ANTT**

1. **THÔNG TIN THÀNH VIÊN NHÓM:**

*(Sinh viên liệt kê tất cả các thành viên trong nhóm)*

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Lê Trần Thùy Trang | 20520323 | 20520323@gm.uit.edu.vn |
| 2 | Bùi Đức Hoàng | 20520514 | 20520514@gm.uit.edu.vn |
| 3 | Phạm Phúc Đức | 20520162 | 20520162@gm.uit.edu.vn |

1. **TÓM TẮT NỘI DUNG THỰC HIỆN:[[1]](#footnote-2)**
   1. **Chủ đề nghiên cứu trong lĩnh vực Mã độc:** *(chọn nội dung tương ứng bên dưới)*

þ Phát hiện mã độc

¨ Đột biến mã độc

¨ Khác: …………………………………

* 1. **Liên kết lưu trữ mã nguồn của nhóm:**

Mã nguồn của đề tài đồ án được lưu tại:

* + <https://drive.google.com/drive/folders/1Kbf787n906WvlaLAOHRtIjSwnwSWae2x?usp=drive_link>

*(Lưu ý: GV phụ trách phải có quyền truy cập nội dung trong Link)*

* 1. **Tên bài báo tham khảo chính:**

|  |
| --- |
| M. S. Islam, K. M. Iftekharuddin and A. I. Khan, "Similarity hash based scoring of portable executable files for efficient malware detection in IoT," IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4515-4526, Dec. 2018. |

* 1. **Dịch tên Tiếng Việt cho bài báo:**

|  |
| --- |
| Đánh giá dựa trên băm tương tự để tính điểm cho các tệp thực thi di động nhằm phát hiện mã độc hiệu quả trong IoT |

* 1. **Tóm tắt nội dung chính:**

|  |
| --- |
| Bài báo khai thác băm tương tự (Similarity hashing) áp dụng để xử lý mẫu trong phân tích và phát hiện phần mềm độc hại. Tính tương tự của tệp được sử dụng để phân cụm phần mềm độc hại thành các cụm sao cho có thể thiết kế chữ ký chung. Bài báo này nghiên cứu bốn loại hàm băm hiện đang được sử dụng trong phân tích phần mềm độc hại cho các tệp portable executable (PE). Mặc dù mỗi kỹ thuật băm tạo ra kết quả đều thể hiện khả năng phát hiện mã độc nhưng khi được áp dụng độc lập, chúng có tỷ lệ phát hiện sai cao. Bài báo nghiên cứu một vấn đề trọng tâm về cách kết hợp các kỹ thuật băm khác nhau để cung cấp điểm độc hại định lượng và để đạt được tỷ lệ phát hiện tốt hơn. Nhóm nghiên cứu thiết kế và phát triển một phương pháp mới để chấm điểm phần mềm độc hại dựa trên kết quả băm. Phương pháp đề xuất được đánh giá thông qua một số thử nghiệm. Đánh giá thể hiện rõ ràng sự cải thiện đáng kể (> 90%) về tỷ lệ phát hiện phần mềm độc hại thực sự. |

* 1. **Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ  Mô tả được tạo tự động  Hình 1: Kiến trúc của hệ thống  Bài báo sử dụng 6 kỹ thuật cho 6 giai đoạn chính trong phương pháp đề xuất, bao gồm:   * **Kỹ thuật 01: Nghiên cứu những tệp đơn lẻ.**   + Chọn một tệp sạch ngẫu nhiên (arp.exe) từ hệ thống Windows. Tệp gốc sẽ được phân tích và tính các giá trị băm quan tâm khác nhau. Sau đó, tệp được chỉnh sửa bằng Radare và các giá trị băm được tính toán lại. Giá trị băm từ hai tệp được so sánh.  Bảng 1: Bảng so sánh hàm băm của arp.exe trước và sau khi sửa đổi     * **Kỹ thuật 02: Thu thập dữ liệu**   + Các bộ dữ liệu trong Bảng 2 và Bảng 3 được thu thập như sau:   * + Thu thập tập dữ liệu về các tệp PE độc hại từ nhiều nguồn như từ kho lưu trữ phần mềm độc hại trực tuyến, honeypots của nhóm tác giả và kho lưu trữ phần mềm độc hại của Nettitude Ltd, Vương quốc Anh.   + Các tệp sạch từ nhiều loại hệ thống Windows khác nhau (ví dụ: Windows XP , Win 7, Win 8 và Win 10).   + Mỗi tệp được lưu dưới dạng tổng MD5 của nó để đảm bảo rằng không có tệp trùng lặp trong tập dữ liệu. Như thể hiện trong Bảng 2, các tệp độc hại được chia thành 3 tập dữ liệu phụ, I, IIm và IIIm, và các tệp Sạch được chia thành 2 tập dữ liệu phụ, IIc và IIIc.   Bảng 2: Chức năng của từng dataset  Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng  Mô tả được tạo tự động  Bảng 3: Số lượng file của từng dataset     * **Kỹ thuật 03: Tạo cơ sở dữ liệu về các Hash Signature**   + Cơ sở dữ liệu của các mã băm (Hdb) về các tệp độc hại sẽ được sử dụng làm chữ ký khởi đầu được tính toán từ các mẫu mã độc. Chúng được lưu trong tập dữ liệu thông qua thuật toán 1.  + Mục đích của bước này là để làm dữ liệu đầu vào cho các bước so sánh hash và các đầu ra liên quan ở các bước sau.  Thuật toán 1: Tạo cơ sở dữ liệu hash     * **Kỹ thuật 04: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI)**   + Tính Detection Rate của mỗi hàm băm:   * + Việc so sánh cần thiết cho các mã hash được tính cho các tệp tin trong Dataset II cùng với Hdb bằng cách khai báo Hflag\_set, trong đó, 4 mã băm đều có 1 vị trí cụ thể riêng biệt. Từng file trong dataset II, 5 mã hash khác nhau sẽ được tính toán. 4 trong số đó được tính toán đưa vào trong database, mỗi query trả về 1 tệp có cấu trúc dữ liệu như dưới.   XHi ← {md5, {Imp\_H, Pe\_H, Sd\_H, RSd\_H}} (1)  Tổng số lượng để đạt được chỉ số trong Confusion Matrix (ma trận hỗn loạn) được đưa ra cho mỗi kỹ thuật hash được thể hiện ở thuật toán sau:  Thuật toán 2: Tính Detection Rate.    + Giá trị CFI sẽ được làm yếu tố tin cậy tương ứng cho từng kỹ thuật hash. Điều này hỗ trợ cho việc phát hiện tệp độc hại. Giá trị trên sẽ được đưa vào bước tiếp theo để đánh giá toàn bộ mức độ nhiễm mã độc của tệp thông qua kiểm tra.  (2)  Trong đó: TDR =   * **Kỹ thuật 05: Ứng dụng của giả thuyết “kết hợp bằng chứng”**   + Yếu tố tin cậy cho các hash được tính toán từ kết quả của query tương ứng.   * + Với PeHash và Imphash, nếu kết quả khác null, ESF tương ứng sẽ được gán bằng CFI của hàm băm đó, còn với kết quả là null, ESF của hash được đặt là 0.   + Đối với Resource Section Ssdeep hash và file Ssdeep hash, ESF tương ứng được gán bằng CFI nhân với tỉ lệ (%) tương đồng tối đa, tỉ lệ này lấy từ việc so sánh tệp với hash trong database.   + Mục đích của việc này là giá trị ESF đã tính toán của các loại hash khác nhau sẽ được kết hợp sử dụng mô hình kết hợp bằng chứng. Việc này là để lấy được tổng đại số cho giả thiết chung được đưa vào TLBSA (Traffic Light Based Scoring Assessor).   * **Kỹ thuật 06: TLBSA Thresholds**   + Kết quả dưới dạng tỉ lệ từ kĩ thuật băm kết hợp được so sánh để có TLBSA tổng thể đánh giá số điểm gán vào tệp. Nó cung cấp cho ngời dùng một gợi ý dựa trên Bảng 4.  Bảng 4: Định nghĩa màu sắc của TLBSA   |  |  |  | | --- | --- | --- | | Colors | Deduced file intent | System Recommendation | | Red | Definitely | Do not install | | Amber | Medium Suspicion | Highly encouraged to submit it for further analysis | | Green | Low Suspicion | Submit it for further analysis |   + Mục đích của việc này là do hệ thống không hoàn toàn đảm bảo tập tin là an toàn, quyết định cuối cùng dựa trên phương thức phân tích tập tin sẽ thuộc về người dùng hệ thống hoặc hệ thống phân tích. |

* 1. **Môi trường thực nghiệm của bài báo:**

|  |
| --- |
| * Cấu hình máy tính: Dell T1700, CPU – Intel Xeon@ 3.1GHz, RAM 32GB. Hard Disk – 500GB. * Các công cụ hỗ trợ sẵn có dùng cho 6 giai đoạn: * Study specific Static Analysis Tool: tính toán Ssdeep, Resource Section Ssdeep hash, PeHash, and Imphash. * SQLite Studio version 3.0.6 * Ngôn ngữ lập trình để hiện thực phương pháp: Python IDLE version 2.7.9 * Đối tượng nghiên cứu: nhóm tác giả đã thu thập được 104528 tệp độc hại. Tất cả tệp này đã được đánh giá bằng công cụ ClamAv phiên bản 0.99.2. * Tiêu chí đánh giá tính hiệu quả của phương pháp: Dựa trên True Detection Rate và hiệu suất (Recall, precision, accurancy, F-score). |

* 1. **Kết quả thực nghiệm của bài báo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * **Bước 1: Nghiên cứu những tệp đơn nguyên gốc.**   Một số hàm băm bị ảnh hưởng lớn bởi một thay đổi nhỏ trong tệp: MD5, SHA1, SHA256.  Một số khác cho thấy khả năng ảnh hưởng nhỏ hoặc không ảnh hưởng: Ssdeep-File, PeHash, Imphash, Ssdeep-Resources  Bảng 5: So sánh hàm băm của arp.exe trước và sau khi sửa đổi    Þ Hàm băm tương tự có hiệu quả trong việc so sánh các tệp tương tự, có sự khác biệt nhỏ về nội dung.   * **Bước 2: Thu thập dữ liệu**   Bảng 6: Dataset của nhóm tác giả     * + Dataset I: Tạo database của các giá trị hash.   + Dataset II: Sử dụng để tính toán các giá trị số liệu CFI của bốn kỹ thuật băm.   + Dataset III: Sử dụng để tính tỷ lệ phần trăm tổng thể của tệp độc hại nhằm xác thực framework được đề xuất. * **Bước 3: Tạo cơ sở dữ liệu về các Hash Signature.**   Sử dụng dataset I để tạo cơ sở dữ liệu Hash Signature.   * **Bước 4: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI).**   + Tính Detection Rate của mỗi hàm băm:  Chart, bar chart  Description automatically generated  Hình 2: Hiệu suất tính được cho các hàm hash  **Þ** So sánh hàm băm của Resource Section được giới thiệu mang lại giá trị chính xác tốt thứ hai trong 4 hàm băm, trong đó PeHash hoạt động tốt nhất trong 4 hàm băm. Imphash cho khả năng phát hiện dương tính giả cao nhất nhưng cũng cung cấp phát hiện âm tính giả thấp nhất. Các mức khác nhau trong tỷ lệ phát hiện cung cấp một đối số để kết hợp chúng nhằm đạt được phương pháp phát hiện hiệu quả hơn.  + Giá trị CFI sẽ được làm yếu tố tin cậy tương ứng cho từng kỹ thuật hash  Bảng 7: Hiệu suất tính được cho các hàm hash và CFI  Chart, bar chart  Description automatically generated   * **Kỹ thuật 05: Ứng dụng của giả thuyết “kết hợp bằng chứng”**   Các kết quả đạt được cho phương pháp đề xuất được so sánh với kết quả đạt được cho từng hàm băm riêng lẻ:  Chart  Description automatically generated with medium confidence  Hình 3: So sánh giữa hàm băm kết hợp và hàm băm riêng lẻ  **Þ** Kết quả của phương pháp băm Kết hợp đạt được tỷ lệ phát hiện sai tổng thể là 6,8% và tỷ lệ phát hiện đúng là 93,2%. Đây là những giá trị hiệu suất tốt nhất so với kết quả đạt được của các thuật toán băm riêng lẻ  Các đường cong tính điểm tệp của từng phương pháp được áp dụng:  Chart, histogram  Description automatically generated  Hình 4: Phân bố điểm độc hại của 2 phương pháp kết hợp  **Þ** Ta thấy hầu hết điểm số của các tệp độc hại đều cao hơn tệp sạch. Cả hai đường cong cho thấy 83% tệp độc hại có điểm độc hại trên 50% trong khi 78% tệp sạch có điểm độc hại dưới 50%.   * **Bước 6: TLBSA Thresholds**   Vì mục đích của nghiên cứu này là tạo ra một phương pháp phát hiện phần mềm độc hại tối ưu, nên họ tiếp tục nghiên cứu sự đánh đổi dương tính thật và dương tính giả hai phương pháp trong Hình 5. Hình 6 được sử dụng để xác định tỷ lệ phần trăm ngưỡng TLBSA.  Chart  Description automatically generated with medium confidence  Hình 5: So sánh true positive rate và false positive rate của các phương pháp kết hợp.  **Þ** Tuy nhiên, khi xem xét đánh đổi giữa phát hiện dương tính thật và dương tính giả, các phương pháp được đề xuất cho thấy kỹ thuật này dễ bị dương tính giả rất cao là 60%, do đó cần phải đánh giá mô hình để đạt được sự đánh đổi tốt hơn. Do đó, nhóm tác giả đã giới thiệu công cụ đánh giá TLBSA ở giai đoạn này.  Chart, bar chart  Description automatically generated  Hình 6: Hiệu suất của các hàm hash riêng lẻ so với phương pháp kết hợp  Với các ngưỡng đạt được, họ đã đánh giá mức độ hoạt động của các phương pháp đối với các thuật toán băm riêng lẻ:  ATP vượt trội hơn tất cả các kỹ thuật băm riêng lẻ. Tuy nhiên, vì tỷ lệ phần trăm này tạo ra tỷ lệ dương tính thật rất thấp là 70% đối với phương pháp Fuzzy logic và 62% đối với phương pháp Common Factor Model.  **Þ** Cần phải phân tích GTP: Nó tạo ra sự gia tăng rất cần thiết trong tỷ lệ dương tính thật là 92% cho cả hai kỹ thuật được đề xuất.  Bảng 8: Phân tích hiệu suất sau khi áp dụng TLBSA   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Phân tích so sánh hiệu quả của phương pháp đề xuất sau khi áp dụng TLBSA**. | | | | | | |  | | Prec (%) | Recall (%) | Acc (%) | F-score | | **Fuzzy Logic Method** | FLM\_GTP (>= 25%) | 99.2 | 92.2 | 91.6 | 95.5 | | FLM\_ATP (>= 75%) | 99.9 | 70.5 | 71.2 | 82.7 | | **Common Factore Method** | CFM\_GTP (>= 25%) | 99.2 | 92.1 | 91.6 | 95.5 | | CFM\_CTP (>= 75%) | 100 | 69.8 | 70.4 | 82.1 |   Þ Việc sử dụng TLBSA làm tăng hiệu quả phát hiện của hệ thống như bảng trên. Tỷ lệ phần trăm ngưỡng cho phép đánh đổi tối ưu và cho phép hệ thống cung cấp cho người dùng thông tin giúp bảo vệ hệ thống của họ với độ chính xác ít nhất 92% đã đạt được trong nghiên cứu này.  Sau đó, nhóm tác giả trình bày tỷ lệ phát hiện của từng loại malware đạt được:  Chart, bar chart  Description automatically generated  Hình 7: Tỉ lệ phát hiện dựa theo loại tấn công  Þ Trong số 8 loại được thu thập, phương pháp được thiết kế cung cấp khả năng phát hiện phần mềm độc hại hiệu quả cho 6 loại.  Nhận xét:  Khả năng: Bài báo nghiên cứu về cách kết hợp các kỹ thuật băm khác nhau để cung cấp điểm độc hại định lượng và để đạt được tỷ lệ phát hiện tốt hơn. Sự cải thiện đáng kể (> 90%) về tỷ lệ phát hiện phần mềm độc hại thực sự.  Ưu điểm: Ít tiêu tốn tài nguyên, tối ưu hóa dung lượng lưu trữ cần thiết cho cơ sở dữ liệu băm, phù hợp với môi trường IoT.  Nhược điểm: Dataset chưa thật sự phù hợp với thực tế, do trong thực tế số lượng file độc hại ít hơn rất nhiều so với file sạch, dẫn đến hiệu suất đánh giá chưa thực sự đáng tin cậy. |

* 1. **Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * Cài đặt các công cụ dùng để tính toán 4 hàm hash đã đề cập trong bài báo: PeHash, ImpHash, SsdeepHash. * Với Resource-Ssdeep Hash, chúng em thực hiện lấy dữ liệu nằm trong Resource Directory của file PE sau đó gom lại và băm. * Các công việc mà nhóm thực hiện theo phương pháp đề xuất:   + **Bước 1: Nghiên cứu các tệp đơn lẻ.**   Sử dụng tệp ban đầu là calc.exe và tính toán các giá trị băm quan tâm,  Sử dụng công cụ HxD để thay đổi một số payload của calc.exe và tính lại các giá trị băm.  So sánh 2 kết quả hash của tệp calc.exe trước và sau khi chỉnh sửa, ta được bảng như sau:  Bảng 9: Bảng so sánh hàm băm của calc.exe trước và sau khi sửa đổi  Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ  Mô tả được tạo tự động  Nhận thấy các giá trị hash MD5, SHA1 và SHA256 cho tỉ lệ giống bé hơn 15%. Ngược lại các hàm còn lại ngoài Ssdeep-File đạt độ tương đồng là 99% thì còn lại tuyệt đối là 100%.  Vì vậy bài báo chỉ sử dụng 4 loại hàm băm lần lượt là Ssdeep cho toàn bộ file, cho resource section, Imphash và PeHash   * + **Bước 2: Thu thập dữ liệu.**   Dataset được thu thập ở link: <https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/>  Bảng thống kê số lượng file của 3 Dataset.  Bảng 10: Dataset của thực nghiệm   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Dataset** | **I** | **II** | **III** | **Total Files** | | **Malicious Files** | 1,624 | 865 | 879 | 3,368 | | **Clean Files** | 0 | 135 | 121 | 256 |  * + **Bước 3: Tạo cơ sở dữ liệu về các Hash Signature**   Bước này sử dụng Dataset I và dữ liệu sẽ gồm có 6 cột như sau:  A screenshot of a computer  Description automatically generated  Hình 8: Theo dõi bằng phần mềm HeidiSQL trên window   * + **Bước 4: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI).**   Dựa vào 2 thuật toán cung cấp ở bước 4, công việc sẽ chia làm 2 bước: tính Detection Rates của các hàm hash riêng lẻ và tính CFI theo công thức đã cung cấp. Cuối cùng sẽ được bảng sau:  Bảng 11: Hiệu suất phát hiện của từng hàm băm riêng lẻ và CFI  **Ảnh có chứa văn bản, ảnh chụp màn hình, số, hình vuông  Mô tả được tạo tự động**   * + **Bước 5: Ứng dụng của giả thuyết “kết hợp bằng chứng”**   Nhóm tiến hành tính ESF và sử dụng giả thuyết “kết hợp bằng chứng” để tính điểm theo phần trăm của các file. Sau khi tính toán nhóm được biểu đồ sau:  A picture containing text, screenshot, font, plot  Description automatically generated  Hình 9: File Score của Common Factor Method  A picture containing text, screenshot, font, plot  Description automatically generated  Hình 10: File Score của Fuzzy logic method  Ở cả 2 đường cong có khoảng 97.5% tệp sạch có điểm độc hại bé hơn 50% và có 67% tệp độc hại được đánh giá có điểm độc hại lớn hơn 50%.   * + **Bước 6: TLBSA thresholds.**   Áp dụng TLBSA thresholds dựa trên điểm của các file, nhóm được bảng sau:  Bảng 12: So sánh hiệu suất sau khi áp dụng TLBSA   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | | **Prec**  **(%)** | **Recall**  **(%)** | **Acc**  **(%)** | **F-Score**  **(%)** | | **Fuzzy Logic Method** | **(FLM\_GTP (≥35%)** | 98.7 | 69.6 | 72.5 | 81.7 | | **(FLM\_ATP (≥65%)** | 99.5 | 45.2 | 51.6 | 62.1 | | **Common Factor Method** | **(CFM\_ GTP (≥35%)** | 97.8 | 70.3 | 72.5 | 81.8 | | **(CFM\_ATP (≥65%)** | 99.7 | 64.9 | 69 | 78.7 |   Như vậy hiệu suất đạt được tốt nhất sau khi sử dụng TLBSA thresholds là FLM\_GTP (≥35%) và CFM\_GTP (≥35%) với Accurancy là 72.5% và các chỉ số hiệu suất còn lại có sự chênh lệch không đáng kể. Ngoài ra, False positive đã giảm còn 6.61% đối với FLM\_GTP (≥35%) và 11.57% bé hơn so với khi chưa áp dụng TLBSA thresholds.  **Link demo và code:**  <https://drive.google.com/drive/folders/1Kbf787n906WvlaLAOHRtIjSwnwSWae2x?usp=drive_link> |

* 1. **Các khó khăn, thách thức hiện tại khi thực hiện:**

|  |
| --- |
| * Do bị giới hạn tài nguyên nên dataset được đưa vào demo bị ít hơn rất nhiều so với bài báo đã đề cập. |

1. **TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:**

|  |
| --- |
| 100% |

1. **NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Công việc** | **Phân công nhiệm vụ** |
| 1 | * Tìm hiểu và cài đặt PeHash. * Tổng hợp kết quả của nhóm tác giả. * Thực nghiệm và viết báo cáo Bước 1, 4, 5. * Làm powerpoint. | Lê Trần Thùy Trang |
| 2 | * Tìm hiểu và cài đặt File Ssdeep hash, Resource Ssdeep File hash. * Tìm hiểu Bước 4 đến Bước 6 của phương pháp bài báo. * Thực nghiệm và viết báo cáo Bước 3 và Bước 6. * Tìm hiểu Static feature evaluation methods. | Phạm Phúc Đức |
| 3 | * Tìm hiểu và cài đặt Imphash. * Tìm hiểu môi trường IoT. * Tìm hiểu Bước 1 đến Bước 3 của phương pháp bài báo. * Thực nghiệm và viết báo cáo bước 2: Phân tích, tổng hợp database cho phần thực nghiệm. | Bùi Đức Hoàng |

BÁO CÁO TỔNG KẾT CHI TIẾT

**Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.**

*Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)*

* 1. **Phương pháp thực hiện**

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ

Mô tả được tạo tự động

Hình 11: Kiến trúc của hệ thống

**Bước 1**: Nghiên cứu các tệp đơn lẻ.

Nghiên cứu này được thực hiện trên một tệp sạch được chọn ngẫu nhiên (arp.exe) từ hệ thống Windows. Tệp gốc sẽ được phân tích và tính các giá trị băm quan tâm khác nhau. Sau đó, tệp được chỉnh sửa bằng Radare và các giá trị băm được tính toán lại. Giá trị băm từ hai tệp được so sánh.

**Bước 2**: Thu thập dữ liệu

* Các bộ dữ liệu trong Bảng 13 và Bảng 14 được thu thập như sau:
  + Nghiên cứu này thu thập tập dữ liệu về các tệp PE độc hại từ nhiều nguồn khác nhau, chẳng hạn như phần mềm độc hại từ kho lưu trữ phần mềm độc hại trực tuyến, honeypots của nhóm tác giả và kho lưu trữ phần mềm độc hại của Nettitude Ltd, Vương quốc Anh.
  + Các tệp sạch từ nhiều loại hệ thống Windows khác nhau (ví dụ: Windows XP , Win 7, Win 8 và Win 10).
  + Mỗi tệp được lưu dưới dạng tổng MD5 của nó để đảm bảo rằng không có tệp trùng lặp trong tập dữ liệu. Như thể hiện trong Bảng 13, các tệp độc hại được chia thành 3 tập dữ liệu phụ I, IIm và IIIm, và các tệp Sạch được chia thành 2 tập dữ liệu phụ, IIc và IIIc.

Bảng 13: Chức năng của từng dataset

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Bảng 14: Số lượng file của từng dataset

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

**Bước 3**: Tạo cơ sở dữ liệu về các Hash Signature

Cơ sở dữ liệu của các mã băm (Hdb) về các tệp độc hại sẽ được sử dụng làm chữ ký khởi đầu được tính toán từ các mẫu mã độc. Chúng được lưu trong tập dữ liệu thông qua thuật toán 3:

Thuật toán 3: Tạo cơ sở dữ liệu hash

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

**Bước 4**: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI)

Dataset II có chứa cả tệp độc hại và tệp an toàn được sử dụng trong quá trình này và bước 4 sẽ gồm có 2 bước nhỏ:

1. Xác định hiệu suất cho từng mã băm dựa trên việc phát hiện mã độc.

Việc so sánh cần thiết cho các mã hash được tính cho các tệp tin trong Dataset II cùng với Hdb bằng cách khai báo Hflag\_set, trong đó, 4 mã băm đều có 1 vị trí cụ thể riêng biệt. Từng file trong dataset II, 5 mã hash khác nhau sẽ được tính toán. 4 trong số đó được tính toán đưa vào trong database, mỗi query trả về 1 tệp có cấu trúc dữ liệu như dưới:

XHi ← {md5, {Imp\_H, Pe\_H, Sd\_H, RSd\_H}} (3)

Trong quá trình đối chiếu của PeHash và Imhash, chỉ có hash khớp với hash đã tính toán trước đó được lấy ra từ database. Vị trí trong HFlag\_set cho mã băm thuộc loại i sẽ không được thiết lập tới nếu Xhi là tập rỗng (null).

Đối với hash Ssdeep resource section và của tệp, tất cả mã băm đều được lấy từ database. Kết quả so sánh độ tương đồng được thực hiện cho các hash của tệp và hash tương ứng được lấy từ database. Nếu mức độ tương đồng (%) lớn hơn 0%, vị trí Hflag\_set tương ứng với loại kỹ thuật băm i sẽ được thiết lập. Tổng số lượng để đạt được chỉ số trong Confusion Matrix (ma trận hỗn loạn) được đưa ra cho mỗi kỹ thuật hash được thể hiện ở thuật toán 4.

Thuật toán 4: Tính toán Detection Rates

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

1. Tính toán CFI cho tất cả các hash cụ thể

Tỷ lệ phát hiện lấy từ bước (a) được sử dụng để tính toán CFI cho từng hash và đây là yếu tố tin cậy (belief factor) cho từng kỹ thuật hash. Để giảm thiểu lỗi sai trong yếu tố tin cậy, tỉ lệ phát hiện thực (True detection rate) được sử dụng để tính toán cho yếu tố trên. Tỉ lệ này sẽ được chuẩn hoá về giá trị trong khoảng [0,1]. Phương pháp Simple additive weighting (phương pháp đánh trọng số) cũng được đưa vào tỉ lệ phát hiện, công thức tính toán CFI như sau:

(5)

Trong đó: TDR =

Giá trị CFI sẽ được làm yếu tố tin cậy tương ứng cho từng kĩ thuật hash. Điều này hỗ trợ cho việc phát hiện tệp độc hại. Giá trị trên sẽ được đưa vào bước tiếp theo để đánh giá toàn bộ mức độ nhiễm mã độc của tệp thông qua kiểm tra.

**Bước 5**: Ứng dụng của giả thuyết “kết hợp bằng chứng”

Như đã nói ở trên, giá trị CFI được sử dụng làm input cho phương pháp kết hợp. Giai đoạn đối chiếu MD5 là 1 bước dự phòng để tránh việc sao chép các mẫu độc hại trong quá trình thử nghiệm. Giai đoạn đối sánh hash sử dụng các tệp đã được tính toán hash và so sánh chúng với Hdb. Query (3) cũng được sử dụng trong giai đoạn này. Yếu tố tin cậy cho các hash được tính toán từ kết quả của query tương ứng.

Với PeHash và Imphash, nếu kết quả khác null, ESF tương ứng sẽ được gán bằng CFI của hàm băm đó, còn với kết quả là null, ESF của hash được đặt là 0.

Đối với Resource Section Ssdeep hash và file Ssdeep hash, ESF tương ứng được gán bằng CFI nhân với tỉ lệ (%) tương đồng tối đa, tỉ lệ này lấy từ việc so sánh tệp với hash trong database. Giá trị ESF đã tính toán của các loại hash khác nhau sẽ được kết hợp sử dụng mô hình kết hợp bằng chứng. Việc này là để lấy được tổng đại số cho giả thiết chung được đưa vào TLBSA (Traffic Light Based Scoring Assessor).

**Bước 6**: TLBSA thresholds

Kết quả dưới dạng tỉ lệ phần trăm từ kỹ thuật băm kết hợp được so sánh với TLBSA tổng thể đánh giá số điểm gán vào tệp. Nó cung cấp cho ngời dùng một gợi ý dựa trên Table V. Do hệ thống không hoàn toàn đảm bảo tập tin là an toàn, quyết định cuối cùng dựa trên phương thức phân tích tập tin sẽ thuộc về người dùng hệ thống hoặc hệ thống phân tích.

Bảng 15: Bảng định nghĩa màu cho TLBSA

|  |  |  |
| --- | --- | --- |
| **Colors** | **Deduced file intent** | **System Recommendation** |
| Red | Definitely | Do not install |
| Amber | Medium Suspicion | Highly encouraged to submit it for further analysis |
| Green | Low Suspicion | Submit it for further analysis |

*<Trình bày kiến trúc, thành phần đã thực hiện (nội dung mà nhóm đã thực hiện)>*

Nhóm đã thực hiện lại tất cả 6 bước của kiến trúc nhóm tác giả đề xuất. Cụ thể:

**Bước 1**: Nghiên cứu các tệp đơn lẻ.

Sử dụng tệp nghiên cứu là calc.exe và tiến hành thanh đổi 1 số byte bằng công cụ HxD như sau:

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động

Hình 12: Thay đổi byte của tệp calc.exe

Cụ thể ở tệp này, ta tiến hành thêm vào 1 chuỗi “20520323” và tệp và lưu lại.

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 13: Tính các hàm hash và so sánh sự tương đồng

**Bước 2**: Thu thập dữ liệu

Ở đây, nhóm chúng em thực hiện thu thập tài nguyên trên máy ảo Ubuntu 22.10, 8GB RAM và 250GB ROM, dataset được thu thập ở link: <https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/>

Sau khi tải file về, ta tiến hành giải nén bằng 7z với option x và -p<PASS>

A picture containing text, screenshot, font

Description automatically generated

Hình 14: Tiến hành tải file

Sau khi giải nén xong, ta thu thập được một thư mục khoảng 127GB, lúc này, ta sẽ tiến hành tải ClamAV để scan files.

A computer screen with white text

Description automatically generated with low confidence

Hình 15: Tải ClamAV

Kiểm tra bằng lệnh sau.

A screenshot of a computer code

Description automatically generated with low confidence

Hình 16: Kiểm tra version của ClamAV

Tiếp đó, do dữ liệu quá lớn nên nhóm chúng em chỉ trích 1 phần nhỏ theo tỉ lệ, dưới đây là lệnh tách 1000 files từ dataset đó.

A picture containing text, screenshot, font

Description automatically generated

Hình 17: Tách 1000 files từ dataset

Sau khi đã tách được 1000 files ra, ta tiến hành phân loại bằng một script nhỏ với clamscan, file nào bị nhiễm sẽ bị cách ly ra một thư mục khác, file nào OK sẽ được giữ lại.

A picture containing text, screenshot, font

Description automatically generated

Hình 18: Tách file độc hại và file sạch

Kết quả sau khi scan xong.

A picture containing text, screenshot, font

Description automatically generated

Hình 19: Kết quả sau khi scan

Ta thực hiện thống kê các loại Malware có trong Dataset với lệnh sau. Đầu tiên cat kết quả thu được từ clamscan vào file result.txt, tiếp đó sử dụng awk để tìm số lần xuất hiện của chuỗi. Làm tương tự với Dataset 2 và Dataset 3.

A picture containing text, screenshot, font

Description automatically generated

Hình 20: Sử dụng awk để phân tách loại mã độc

Bảng thống kê số lượng file của 3 Dataset.

Bảng 16: Dataset của thực nghiệm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **I** | **II** | **III** | **Total Files** |
| **Malicious Files** | 1,624 | 865 | 879 | 3,368 |
| **Clean Files** | 0 | 135 | 121 | 256 |

**Bước 3**: Tạo cơ sở dữ liệu về các Hash Signature

Các hàm sử dụng để lấy mã hash từ các file độc hại từ dataset 1:

def calculate\_md5(file\_path):

with open(file\_path, 'rb') as f:

data = f.read()

md5\_hash = hashlib.md5(data).hexdigest()

return md5\_hash

def calculate\_pehash(file\_path):

pehash\_result = pehash.totalhash\_hex(file\_path)

return pehash\_result

def calculate\_imphash(file\_path):

pe = pefile.PE(file\_path)

# Tính toán ImpHash

imphash = pe.get\_imphash()

return imphash

def calculate\_ssdeep\_hash(file\_path):

with open(file\_path, 'rb') as file:

file\_data = file.read()

hash\_value = ssdeep.hash(file\_data)

return hash\_value

def calculate\_resource\_ssdeep\_hash(file\_path):

pe = pefile.PE(file\_path)

combined\_data = b"" # Dữ liệu kết hợp của tất cả các tài nguyên

if hasattr(pe, 'DIRECTORY\_ENTRY\_RESOURCE'):

resource\_section = pe.DIRECTORY\_ENTRY\_RESOURCE

for resource\_type in resource\_section.entries:

if hasattr(resource\_type, 'directory'):

for resource\_id in resource\_type.directory.entries:

if hasattr(resource\_id, 'directory'):

for resource\_lang in resource\_id.directory.entries:

if hasattr(resource\_lang, 'data'):

data\_rva = resource\_lang.data.struct.OffsetToData

size = resource\_lang.data.struct.Size

data = pe.get\_memory\_mapped\_image()[data\_rva:data\_rva+size]

combined\_data += data # Gom tất cả các tài nguyên lại

pe.close()

ssdeep\_hash = ssdeep.hash(combined\_data) # Tính toán SSDeep hash của dữ liệu kết hợp

return ssdeep\_hash

Các hàm hầu hết chỉ gọi lại các phương thức đã có sẵn trong các module, tuy nhiên với resource, chúng ta cần phải duyệt từng dữ liệu được chia ra trong Resource Section, sau đó kết nối chúng lại với nhau và hash với ssdeep.

Cụ thể, khi phân tích một tệp tin PE để truy cập vào dữ liệu trong phần tài nguyên, chúng ta cần đi qua nhiều mục lục (directory entries) khác nhau. Cấu trúc phân cấp của phần tài nguyên trong một tệp tin PE là:

* Directory Entry của Section .rsrc: Đây là mục lục gốc chứa thông tin về phần tài nguyên → truy cập vào đây để tiếp tục đi sâu vào phần tài nguyên.
* Directory Entry của Resource Type: Mục lục này chứa thông tin về các loại tài nguyên khác nhau, ví dụ như ảnh, âm thanh, biểu tượng, v.v. → tiếp tục truy cập đi sâu vào từng mục trong đó
* Directory Entry của Resource ID: Mục lục này chứa thông tin về các ID tài nguyên trong mỗi loại tài nguyên → duyệt qua từng loại tài nguyên để lấy được dữ liệu.

Tất cả giá trị từ hàm hash sẽ được đưa vào hàm dưới để đẩy lên MySQL bằng thư viện pymysql:

def insert\_hash\_to\_database(file\_path, md5\_value, sd\_h\_value, pe\_h\_value, imp\_h\_value, rsd\_h\_value):

conn = pymysql.connect(

host='10.0.139.42',

user='nhom13',

password='123456',

database='dataset'

)

cursor = conn.cursor()

query = "INSERT INTO dataset1 (file\_path, sd\_h, md5, pe\_h, imp\_h, rsd\_h) VALUES (%s, %s, %s, %s, %s, %s) ON DUPLICATE KEY UPDATE sd\_h = %s, md5 = %s, pe\_h = %s, imp\_h = %s, rsd\_h = %s"

values = (file\_path, sd\_h\_value, md5\_value, pe\_h\_value, imp\_h\_value, rsd\_h\_value, sd\_h\_value, md5\_value, pe\_h\_value, imp\_h\_value, rsd\_h\_value)

cursor.execute(query, values)

conn.commit()

cursor.close()

conn.close()

Bởi vì nó được thiết kế cho sử dụng trong hệ thống IoT, nên lượng tài nguyên nó chiếm khi chạy cũng sẽ không cao:

A screenshot of a computer

Description automatically generated

Hình 21: Lượng tài nguyên cần cho việc tính toán dataset 1

Theo đó, sau khi kết nối được tới MySQL với thông tin host, user, password, database, hàm sẽ thực hiện đẩy dữ liệu vào bằng cách query tới đó.

Dữ liệu sẽ gồm có 6 cột như sau:

A screenshot of a computer

Description automatically generated

Hình 22: Theo dõi bằng phần mềm HeidiSQL trên window

**Bước 4**: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI)

Dataset II có chứa cả tệp độc hại và tệp an toàn được sử dụng trong quá trình này và bước 4 sẽ gồm có 2 bước nhỏ:

1. Xác định hiệu suất cho từng mã băm dựa trên việc phát hiện mã độc.

Ở bước này ta sử dụng Dataset2. Ban đầu, thiết lập Confusion Matrix dưới dạng 1 list:

Ảnh có chứa ảnh chụp màn hình, văn bản, Phông chữ, số

Mô tả được tạo tự động

Hình 23: Thiết lập Confusion matrix

Tạo 1 hàm updateDetectionRate thực hiện 2 chức năng chính là tính toán các giá trị hash của file đang xét và tính toán lại Matrix Confusion đã tạo trước đó của từng loại hash.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 24: Hàm updateDetectionRate

Như bài báo đã giới thiệu, hai hàm hash PeHash và Imphash có cách tính giống nhau.

Nếu tồn hash giá trị hash trong database đúng bằng giá trị hash của file ta đang xét đến:

* Nếu đây là file độc hại (có nhãn là độc hại) thì giá trị TP + 1.
* Nếu đây là file sạch (có nhãn sạch) thì FP + 1.

Nếu không có giá trị hash nào trong database đúng giá trị hash của file ta đang xét đến:

* Nếu đây là file độc hại (có nhãn là độc hại) thì giá trị FN + 1.
* Nếu đây là file sạch (có nhãn sạch) thì TN + 1

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 25: Hàm set\_Hflag\_Im\_Pe

Đối với File Ssdeep Hash và Resource section Ssdeep Hash thay vì tìm kiếm chính xác giá trị hash cần tìm ở trong database, ta chỉ cần quan tâm phần trăm tương tự lớn nhất giữa giá trị hash của file đang xét và các giá trị hash trong database lớn hơn 0.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 26: Hàm set\_Hflag\_Sd\_RSd

Sau bước này ta đã cập nhật xong Confusion Matrix, ta dựa vào giá trị tính được này để tính TDR, CFI, Recall, PPV, ACC, F1.

Ảnh có chứa Phông chữ, văn bản, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 27: Tính các giá trị TDR, Recall, PPV, ACC, F1

1. Tính toán CFI cho tất cả các hash cụ thể

Như đã mô tả ở trên CFI được tính bằng công thức:

Trong đó: TDR =

Dựa vào công thức được cung cấp ta thiết lập code:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 28: Tính CFI

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

Hình 29: Tính hiệu suất và CFI của các hàm băm

**Bước 5**: Ứng dụng của giả thuyết “kết hợp bằng chứng”

Sử dụng Dataset3 cho bước này, đầu tiên ta định nghĩa 1 list ESF = [0, 0, 0, 0] đại diện lần lượt cho ESF của Imphash, Pehash, File Ssdeep hash và Resource Ssdeep Hash.

Tiến hành tính toán các giá trị hash cho file đang xét sau đó sử dụng các hàm cal\_ESF để cập nhật lại giá trị ESF cho các hàm hash tương ứng.

Cuối cùng sử dụng hàm **getCombineMethodFuzzyLogic** và **getCombineMethodCertaintyFactorModel** để tính giá trị ESF theo 2 Evidence Combinational Methods đã đề cập trước đó.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Hình 30: Hàm CombinationApproach

Đối với Imphash và Pehash, nếu tồn tại giá trị hash của file trong database thì giá trị ESF của 2 loại hash này sẽ là CFI tương ứng của hàm hash đã tính ở bước 4. Ngược lại giá trị ESF sẽ bằng 0.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 31: Hàm cal\_ESF\_Imp\_Pe

Đối với File Ssdeep hash và Resource Ssdeep hash thì ESF = CFI \* max\_similar. max\_similar của giá trị hash của file sẽ tìm được bằng cách so sánh tương tự lần lượt giá trị của file với từng giá trị trong database, mỗi bước so sánh ta sẽ đưa ra mức độ tương đồng tính theo % của giá trị hash của file và giá trị hash lấy từ database, ta sẽ lấy con số lớn nhất đó chính là max\_similar.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 32: Hàm calculate\_max\_similar

Cuối cùng ta tiến hành khởi tạo 2 hàm FuzzyLogic và TheCertaintyFactor tương ứng với 2 phương pháp kết hợp đã nêu ở trên cho 2 giá trị. Ta mở rộng công thức cho 4 giá trị tương ứng với ESF tính được của từng hàm hash.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 33: Hàm FuzzyLogic và TheCertaintyFactor

Ta nhận thấy TDR của phương pháp kết hợp đạt được là:

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, thuật in máy

Mô tả được tạo tự động

Hình 34: Hiệu suất của phương pháp kết hợp

**Bước 6**: TLBSA thresholds

Sử dụng các tệp dữ liệu về điểm đã phân loại ở trên, chúng ta sẽ tiến hành tính toán Precision, Recall, Accurancy và F-Score dựa trên threshold đưa vào:

def calculate\_tp\_fp\_tn\_fn(clean\_scores, malicious\_scores, threshold):

tp = sum(score >= threshold for score in malicious\_scores)

fp = sum(score >= threshold for score in clean\_scores)

tn = sum(score < threshold for score in clean\_scores)

fn = sum(score < threshold for score in malicious\_scores)

try:

precision = 100\*tp / (tp + fp)

except ZeroDivisionError:

precision = "?"

try:

recall = 100\*tp / (tp + fn)

except ZeroDivisionError:

recall = "?"

try:

accuracy = 100\*(tp + tn) / (tp + fp + tn + fn)

except ZeroDivisionError:

accuracy = "?"

try:

f1\_score = 2 \* (precision \* recall) / (precision + recall)

except Exception:

f1\_score = "?"

Threshold ở đây sẽ là 2 chỉ số ATP và GTP, hai chỉ số này sẽ được chọn dựa trên dữ liệu file score ở trên và ngưỡng hợp lý cho 2 chỉ số trên sẽ là: GTP ≥ 35% và ATP ≥ 65%.

Tổng hợp kết quả như sau:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 35: Thực thi chương trình tính toán dựa trên TLBSA

* 1. **Chi tiết cài đặt, hiện thực**

*<cách cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu, v.v>*

**Cấu hình máy tính:**

* Cấu hình máy sử dụng ở Bước 1, 4, 5:

Sử dụng máy ảo Linux kali 5.18.0 - 11th Gen Intel@ 2.40GHz, 2GB RAM. Hard Disk – 80.1GB.

* Cấu hình máy sử dụng ở Bước 2:

Sử dụng máy ảo Ubuntu 22.10, 8GB RAM và 250GB ROM.

* Cấu hình máy sử dụng ở Bước 3, 6:

Lenovo IdeapadGaming 3, Intel(R) Iris(R) Xe Graphics 3.11Ghz, Ram 2GB, Hard Disk 80GB, Kali Linux

**Các công cụ sử dụng:**

Python 3.11.2

MySQL 8.0.33

**Chuẩn bị dữ liệu:**

Dataset được thu thập ở link: <https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/>

Bảng 17: Thông tin dataset sử dụng

|  |  |
| --- | --- |
| **Mô tả** | **Số lượng** |
| Số lượng file độc hại | 114,737 |
| Số lượng file sạch | 86,812 |
| Tổng cộng | 201,549 |

**Cài đặt:**

* **Cài đặt Ssdeep:**

Bước 1: Cài đặt ssdeep bằng lệnh dưới:

sudo apt install ssdeep

A screenshot of a computer

Description automatically generated

Hình 36: Cài đặt ssdeep

Bước 2: Tải về package hỗ trợ cho ssdeep trên máy ảo Ubuntu:

git clone https://github.com/ssdeep-project/ssdeep.git

A screenshot of a computer screen

Description automatically generated with medium confidence

Hình 37: Cài đặt và sử dụng trên flareVM

Để hiểu rõ hơn về các câu lệnh ssdeep có thể hỗ trợ, chúng ta sử dụng lệnh man ./ssdeep.1.

Lệnh ssdeep có dạng như sau:

ssdeep [-m <file>] [-k <file>] [-vdprgsblcxa] [-t val] [FILES]

ssdeep [-V|h]

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Hình 38: Các dạng lệnh ssdeep

Tính hash của tất cả file trong thư mục bằng cả md5 và ssdeep:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Hình 39: Tính hash

Bước đầu có thể thấy kết quả tạo ra có dạng như trong bài báo đã đề cập ở trên, ví dụ như:

ssdeep,1.1--blocksize:hash:hash,filename

12288:xde7wL0OHXHSdfd+6g8L3U+zWO3Epy3nQqmpnk+:xo7+0kydfd+hME+J0csk+,"/home/pengu/Desktop/1066860.jpg"

Bước 3: Tìm các (cặp) tập tin cho kết quả hash giống nhau, chúng ta sử dụng lệnh

ssdeep -m results.txt \* -s

Kết quả cho thấy có 1 cặp file cho ra kết quả tương đồng tới 61%:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Hình 40: Kết quả so sánh tương đồng

Để sử dụng ssdeep như một module trong python, ta tiến hành cài đặt:

|  |
| --- |
| sudo apt-get -y install libfuzzy-dev |

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Hình 41: Tải thư viện cần thiết

|  |
| --- |
| sudo pip install ssdeep --use-pep517 |

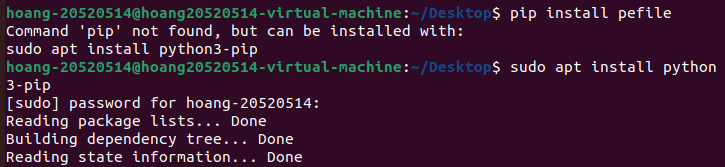
Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Hình 42: Tải ssdeep thông qua pip

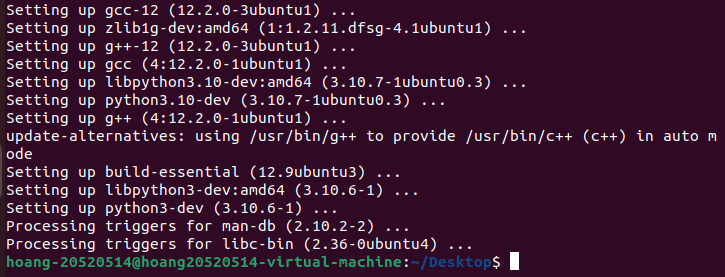
* **Cài đặt Imhash:**
* Việc cài đặt Imphash khá là dễ, do theo tài liệu tham khảo thì Imphash đã được cài đặt sẵn trong Module “pefile.py”
* Đầu tiên, ta tiến hành cài đặt pip bằng cách cài đặt python3-pip

|  |
| --- |
| sudo apt install python3-pip |



Hình 43: Cài đặt pip

* Hình ảnh cài đặt thành công.



Hình 44: Cài đặt pip thành công

* Sau khi đã cài đặt thành công, ta tiến hành cài đặt module pefile.

|  |
| --- |
| pip install pefile |

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 45: Cài đặt module pefile

* Sau khi đã cài đặt thành công, ta tiến hành chạy Imphash bằng các thao tác sau.

|  |
| --- |
| python3  import pefile  pefile.PE(“File name.exe”).get\_imphash() |

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình 46: Sử dụng imphash

* **Cài đặt PeHash:**

Đầu tiên, cài đặt các công cụ cần thiết:

* Cần cài đặt git và pip bằng lệnh sau:

|  |
| --- |
| sudo apt-get update && sudo apt-get install git -y && sudo apt-get install pip -y |

* Do ở đây máy đã cài đặt git nên chỉ cần cài đặt thêm git:

A picture containing text, screenshot, font

Description automatically generated

Hình 47: Cài đặt git

* Tải mã nguồn của PeHash bằng cách sử dụng Git, bằng cách chạy lệnh sau:

|  |
| --- |
| git clone https://github.com/knowmalware/pehash.git |

A picture containing text, screenshot, font

Description automatically generated

Hình 48: Tải mã nguồn PeHash

* Liệt kê các file của mã nguồn này:



Hình 49: Liệt kê các file trong mã nguồn

Cài đặt các yêu cầu của PeHash bằng cách sử dụng lệnh pip:

|  |
| --- |
| sudo pip install -r requirements.txt |

A picture containing text, screenshot, font

Description automatically generated

Hình 50: Tải các yêu cầu của PeHash

Sau khi cài đặt thành công, bạn có thể sử dụng PeHash để tính toán giá trị băm của file PE bằng cách chạy lệnh sau:

|  |
| --- |
| python3 pehash.py <path\_to\_PE\_file> |

A picture containing text, screenshot, font

Description automatically generated

Hình 51: Tính PeHash của file calc.exe

Có thể cài đặt PeHash như một module trong Python bằng cách chạy lệnh sau:

|  |
| --- |
| sudo python3 setup.py install |

A picture containing text, screenshot, font

Description automatically generated

Hình 52: Cài đặt module PeHash

Khi sử dụng ta chỉ cần import module này như sau vào file:

import pehash

**Link demo và code:**

<https://drive.google.com/drive/folders/1Kbf787n906WvlaLAOHRtIjSwnwSWae2x?usp=drive_link>

* 1. **Kết quả thực nghiệm**

*<mô tả hình ảnh về thực nghiệm, bảng biểu số liệu thống kê từ thực nghiệm, nhận xét về kết quả thu được.>*

**Bước 1: Nghiên cứu các tệp đơn lẻ.**

So sánh 2 kết quả hash của tệp calc.exe trước và sau khi chỉnh sửa, ta được bảng như sau:

Bảng 18: Bảng so sánh hàm băm của calc.exe trước và sau khi sửa đổi

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

Nhận thấy các giá trị hash MD5, SHA1 và SHA256 cho tỉ lệ giống bé hơn 15%. Ngược lại các hàm còn lại ngoài Ssdeep-File đạt độ tương đồng là 99% thì còn lại tuyệt đối là 100%.

Vì vậy bài báo chỉ sử dụng 4 loại hàm băm lần lượt là Ssdeep cho toàn bộ file, cho resource section, Imphash và PeHash.

**Bước 2: Thu thập dữ liệu.**

Bảng thống kê số lượng file của 3 Dataset.

Bảng 19: Dataset của thực nghiệm

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Bản phân loại mã độc có trong dataset.

Bảng 20: Phân bố mã độc ở Dataset 1

|  |  |  |  |
| --- | --- | --- | --- |
| **DATASET 1** | | | |
| **Malware Type** | **Percentage** | **Malware Type** | **Percentage** |
| Trojan | 44,1% (716) | Virus | 0,62% (10) |
| Adware | 48,1% (781) | Spyware | 0,5% (8) |
| Worm | 1,6% (26) | Exploit | 0% (0) |
| Downloader | 3,2% (52) | Etc | 0,8% (13) |
| Dropper | 0,8% (13) |  |  |

Bảng 21: Phân bố mã độc ở Dataset 2

|  |  |  |  |
| --- | --- | --- | --- |
| **DATASET 2** | | | |
| **Malware Type** | **Percentage** | **Malware Type** | **Percentage** |
| Trojan | 45,32% (392) | Virus | 1% (9) |
| Adware | 46% (398) | Spyware | 0,7% (6) |
| Worm | 2% (17) | Exploit | 0,12% (1) |
| Downloader | 3,24% (28) | Etc | 0,7% (6) |
| Dropper | 0,8% (7) |  |  |

Bảng 22: Phân bố mã độc ở Dataset 3

|  |  |  |  |
| --- | --- | --- | --- |
| **DATASET 3** | | | |
| **Malware Type** | Percentage | **Malware Type** | Percentage |
| Trojan | 43,8% (385) | Virus | 1,3% (11) |
| Adware | 46,8% (411) | Spyware | 0,2% (2) |
| Worm | 2,6% (23) | Exploit | 0% (0) |
| Downloader | 3,3% (29) | Etc | 0,7% (6) |
| Dropper | 1,3% (11) |  |  |

Do hạn chế về tài nguyên nên nhóm chỉ thu thập số lượng file độc hại ít hơn rất nhiều so với bài nghiên cứu (~1/32) vì vậy kết quả thực nghiệm có sự chênh lệch nhất định đối với hiệu suất được mô tả trong bài báo.

**Bước 3: Tạo cơ sở dữ liệu về các Hash Signature.**

Bởi vì nó được thiết kế cho sử dụng trong hệ thống IoT, nên lượng tài nguyên nó chiếm khi chạy cũng sẽ không cao:

A screenshot of a computer

Description automatically generated

Hình 53: Lượng tài nguyên cần cho việc tính toán dataset 1

Dữ liệu sẽ gồm có 6 cột như sau:

A screenshot of a computer

Description automatically generated

Hình 54: Theo dõi bằng phần mềm HeidiSQL trên window

Nhận thấy đã thêm vào database tổng cộng 1,624 dòng. Database này sẽ dùng làm Hash Signature cho bước 4, 5. Nhận thấy số lượng Hashes Signatures tương đối nhỏ so với bài báo, điều này ảnh hưởng trực tiếp đến hiệu suất phát hiện của thực nghiệm.

**Bước 4: Áp dụng công thức đánh giá chỉ số dựa trên sự tương đồng mã hash (CFI).**

Sao khi hoàn thành tính toán hiệu suất và CFI cho từng kỹ thuật băm ta được bảng sau:

Bảng 23: Hiệu suất phát hiện của từng hàm băm riêng lẻ và CFI

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, hình vuông

Mô tả được tạo tự động**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ

Mô tả được tạo tự động

Hình 55: Sơ đồ biểu diễn hiệu suất của các hàm băm

Do sự chênh lệch nhiều so với số lượng mẫu ở dataset của nhóm tác giả nên số liệu nhóm thu thập được thấp hơn rất nhiều so với bài báo đưa ra.

**Bước 5: Ứng dụng của giả thuyết “kết hợp bằng chứng”.**

Sau khi tính toán ta được biểu đồ thể hiện độ phân bố của File Score dựa trên phần trăm:

A picture containing text, screenshot, font, plot

Description automatically generated

Hình 56: File Score của Common Factor Method

A picture containing text, screenshot, font, plot

Description automatically generated

Hình 57: File Score của Fuzzy logic method

Từ biểu đồ ta có thể tính được ở cả 2 đường cong có khoảng 97.5% tệp sạch có điểm độc hại bé hơn 50% và có 67% tệp độc hại được đánh giá có điểm độc hại lớn hơn 50%. So sánh True Detection Rates của phương pháp kết hợp với các phương pháp khác. Ta nhận thấy TDR của phương pháp kết hợp đạt được tính là:

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, thuật in máy

Mô tả được tạo tự động

Hình 58: Hiệu suất của phương pháp kết hợp

So sánh hiệu suất với các hàm băm riêng lẻ ta nhận được biểu đồ sau:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tạo tự động

Hình 59: So sánh Combined Hashing Approach với các hàm hash riêng lẻ.

Nhóm nhận thấy tỉ lệ True Decection Rates đạt được của phương pháp kết hợp cao hơn các phương pháp riêng lẻ. Cụ thể ở phương pháp kết hợp đạt được True Detection Rates là 78.4%. Tuy nhiên false positive cũng cao là 35.5% vì vậy bước sử dụng TLBSA threshold được sử dụng nhằm đánh đổi giữa true positive và false positive.

**Bước 6: TLBSA thresholds.**

Bảng 24: So sánh hiệu suất sau khi áp dụng TLBSA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Prec**  **(%)** | **Recall**  **(%)** | **Acc**  **(%)** | **F-Score**  **(%)** |
| **Fuzzy Logic Method** | **(FLM\_GTP (≥35%)** | 98.7 | 69.6 | 72.5 | 81.7 |
| **(FLM\_ATP (≥65%)** | 99.5 | 45.2 | 51.6 | 62.1 |
| **Common Factor Method** | **(CFM\_ GTP (≥35%)** | 97.8 | 70.3 | 72.5 | 81.8 |
| **(CFM\_ATP (≥65%)** | 99.7 | 64.9 | 69 | 78.7 |

Như vậy hiệu suất đạt được tốt nhất sau khi sử dụng TLBSA thresholds là FLM\_GTP (≥35%) và CFM\_GTP (≥35%) với Accurancy là 72.5% và các chỉ số hiệu suất còn lại có sự chênh lệch không đáng kể. Ngoài ra, False positive đã giảm còn 6.61% đối với FLM\_GTP (≥35%) và 11.57% bé hơn so với khi chưa áp dụng TLBSA thresholds.

**So sánh giữa kết quả thực nghiệm và kết quả cung cấp của bài báo:**

Do hạn chế về tài nguyên, chúng em chỉ sử dụng kích thước dataset nhỏ nên dẫn đến các hiệu suất trong quá trình thực nghiệm giảm đi rất nhiều so với kết quả được cung cấp.

Mặc dù có sự giảm đáng kể về mặt hiệu suất nhưng nhìn chung thực nghiệm vẫn chứng minh được rằng việc áp dụng phương pháp kết hợp có thể đem lại True Detection Rate cao hơn các phương pháp riêng lẻ.

* 1. **Hướng phát triển**

Ngoài các hàm băm tương tự đã được mô tả trên, ta có thể sử dụng một số hàm băm tương tự khác như mvHash-B, sdHash,… để kết hợp và so sánh kết hợp các hàm băm khác nhau nhằm đánh giá các hàm băm nào kết hợp đạt hiệu quả tốt nhất.

Bài báo mang lại tính ứng dụng cao trong việc phát hiện các tệp độc hại trong môi trường khan hiếm tài nguyên như IoT. Phương pháp đơn giản hóa việc xác định các tập tin độc hại bằng cách cung cấp một giá trị định lượng cho biết mức độ độc hại của tệp. Nó cũng tối ưu hóa dung lượng lưu trữ cần thiết cho cơ sở dữ liệu băm. Vì vậy phương pháp được đề xuất cung cấp cách xây dựng một hệ thống phát hiện phần mềm độc hại tích hợp, hiệu quả cho thiết bị IoT.

---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

# **YÊU CẦU CHUNG**

* Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
* Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
* Sinh viên báo cáo kết quả thực hiện và nộp bài.

**Báo cáo:**

* File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
* Đặt tên theo định dạng: [Mã lớp]-Project\_Final\_NhomX\_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

*Ví dụ: [NT521.N11.ANTT]-Project\_Final\_Nhom03\_CK01.*

* Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
* Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**:

* Hoàn thành tốt yêu cầu được giao.
* Có nội dung mở rộng, ứng dụng.

*Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**

1. Ghi nội dung tương ứng theo mô tả [↑](#footnote-ref-2)