**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**Đồ án chuyên ngành**

**Đề tài:**

**A study on detecting and unpacking packed Windows executable files**

**Nghiên cứu phương pháp phát hiện và giải nén các tập tin thực thi Windows**

Giảng viên hướng dẫn: **Đỗ Thị Thu Hiền**

Lớp: **NT114.P11.ANTT**

Sinh viên thực hiện: **Trần Lê Minh Ngọc 21521195**

**Lưu Thị Huỳnh Như 21521242**

*TP.Hồ Chí Minh, ngày ... tháng ... năm 2024*

**NHẬN XÉT TỪ GIẢNG VIÊN**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………………………………..……………………………………………………………………

# MỤC LỤC

[MỤC LỤC i](#_Toc187817037)

[LỜI CẢM ƠN iii](#_Toc187817038)

[DANH MỤC HÌNH VẼ iv](#_Toc187817039)

[DANH SÁCH BẢNG viii](#_Toc187817040)

[TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH 1](#_Toc187817041)

[CHƯƠNG 1: TỔNG QUAN 2](#_Toc187817042)

[1.1 Giới thiệu vấn đề 2](#_Toc187817043)

[1.2 Thách thức và phương pháp 3](#_Toc187817044)

[1.2.1 Thách thức 3](#_Toc187817045)

[1.2.2 Phương pháp 4](#_Toc187817046)

[1.3 Mục tiêu, nội dung cụ thể 4](#_Toc187817047)

[1.3.1 Mục tiêu 4](#_Toc187817048)

[1.3.2 Nội dung cụ thể 5](#_Toc187817049)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 7](#_Toc187817050)

[2.1 Tập tin thực thi Windows 7](#_Toc187817051)

[2.1.1 Portable Executable – PE là gì? 7](#_Toc187817052)

[2.1.2 Cấu trúc file PE 8](#_Toc187817053)

[2.2 Trình đóng gói 14](#_Toc187817054)

[2.2.1 Trình đóng gói (Packer) là gì? 14](#_Toc187817055)

[2.2.2 Cách đóng gói tập tin 15](#_Toc187817056)

[2.2.3 Các công cụ đóng gói phổ biến 18](#_Toc187817057)

[2.3 Các công cụ phát hiện và giải nén các tập tin thực thi Windows 30](#_Toc187817058)

[2.3.1 Unipacker 30](#_Toc187817059)

[2.3.2 ClamAV 38](#_Toc187817060)

[2.3.3 Detect It Easy 43](#_Toc187817061)

[2.3.4 PEiD 47](#_Toc187817062)

[CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM 51](#_Toc187817063)

[3.1 Triển khai các công cụ đóng gói 51](#_Toc187817064)

[3.1.1 UPX 51](#_Toc187817065)

[3.1.2 FSG 53](#_Toc187817066)

[3.1.3 MEW 55](#_Toc187817067)

[3.1.4 MPRESS 57](#_Toc187817068)

[3.2 Triển khai các công cụ phát hiện và giải nén các tập thực thi Windows 59](#_Toc187817069)

[3.2.1 Unipacker 59](#_Toc187817070)

[3.2.2 ClamAV 65](#_Toc187817071)

[3.2.3 PEiD 67](#_Toc187817072)

[3.2.4 DIE 69](#_Toc187817073)

[CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM 72](#_Toc187817074)

[4.1 Bộ dữ liệu 72](#_Toc187817075)

[4.2 Cấu hình máy sử dụng cho thực nghiệm 73](#_Toc187817076)

[4.3 Tự động hóa 73](#_Toc187817077)

[CHƯƠNG 5: HƯỚNG PHÁT TRIỂN 84](#_Toc187817078)

[5.1 Kết luận 84](#_Toc187817079)

[5.2 Hướng phát triển 85](#_Toc187817080)

[TÀI LIỆU THAM KHẢO 87](#_Toc187817081)

# LỜI CẢM ƠN

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

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

Em xin chân thành cảm ơn cô!

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

# DANH MỤC HÌNH VẼ

[Hình 1. Cấu trúc một file PE 8](#_Toc187856614)

[Hình 2. Một tập tin thực thi đã bị đóng gói 16](#_Toc187856615)

[Hình 3. Công cụ nén tệp tin UPX 18](#_Toc187856616)

[Hình 4. Công nghệ nén tệp tin ASPack 21](#_Toc187856617)

[Hình 5. Công cụ nén tệp tin MPRESS 22](#_Toc187856618)

[Hình 6. Công cụ nén tệp tin PECompact 24](#_Toc187856619)

[Hình 7. Công cụ nén tập tin FSG 26](#_Toc187856620)

[Hình 8. Công cụ nén tệp tin MEW 27](#_Toc187856621)

[Hình 9. Công cụ nén tệp tin PETITE 29](#_Toc187856622)

[Hình 10. Công cụ Unipacker 31](#_Toc187856623)

[Hình 11. Dấu hiệu nhận biết file bị nén bởi UPX 32](#_Toc187856624)

[Hình 12. Dấu hiệu nhận biết file bị nén bởi FSG 32](#_Toc187856625)

[Hình 13. Dấu hiệu nhận biết file bị nén bởi ASPack 32](#_Toc187856626)

[Hình 14. Dấu hiệu nhận biết file bị nén bởi PECompact 33](#_Toc187856627)

[Hình 15. Dấu hiệu nhận biết file bị nén bởi PETITE 33](#_Toc187856628)

[Hình 16. Dấu hiệu nhận biết file bị nén bởi MEW 33](#_Toc187856629)

[Hình 17. Dấu hiệu nhận biết file bị nén bởi MPress 33](#_Toc187856630)

[Hình 18. Dấu hiệu nhận biết file bị nén bởi YZPack 34](#_Toc187856631)

[Hình 19. Trình giải nén công cụ UPX 36](#_Toc187856632)

[Hình 20. Trình giải nén công cụ PETITE 36](#_Toc187856633)

[Hình 21. Trình giải nén công cụ FSG 36](#_Toc187856634)

[Hình 22. Trình giải nén công cụ ASPack 37](#_Toc187856635)

[Hình 23. Trình giải nén công cụ YZPack 37](#_Toc187856636)

[Hình 24. Trình giải nén công cụ MEW 37](#_Toc187856637)

[Hình 25. Trình giải nén công cụ MPress 38](#_Toc187856638)

[Hình 26. Trình giải nén công cụ PECompact 38](#_Toc187856639)

[Hình 27. Công cụ ClamAV 39](#_Toc187856640)

[Hình 28. Các file chứa đoạn code liên quan đến UPX 40](#_Toc187856641)

[Hình 29. Đoạn code nhận biết UPX 41](#_Toc187856642)

[Hình 30. Đoạn code giải nén upx\_inflate2b 41](#_Toc187856643)

[Hình 31. Đoạn code giải nén upx\_inflate2d 42](#_Toc187856644)

[Hình 32. Các file chứa đoạn code liên quan đến FSG 42](#_Toc187856645)

[Hình 33. Các file chứa đoạn code liên quan đến ASPack 42](#_Toc187856646)

[Hình 34. Các file chứa đoạn code liên quan đến PETITE 43](#_Toc187856647)

[Hình 35. Các file chứa đoạn code liên quan đến MEW 43](#_Toc187856648)

[Hình 36. Công cụ Detect It Easy 43](#_Toc187856649)

[Hình 37. Các file trong DiE liên quan đến UPX 45](#_Toc187856650)

[Hình 38. Các file trong DiE liên quan đến ASPack 45](#_Toc187856651)

[Hình 39. Các file trong DiE liên quan đến Mpress 45](#_Toc187856652)

[Hình 40. DiE nhận biết tệp tin bị nén bởi FSG 46](#_Toc187856653)

[Hình 41. Công cụ PeID 47](#_Toc187856654)

[Hình 42. Signature của trình đóng gói FSG trong PEiD 49](#_Toc187856655)

[Hình 43. Signature của trình đóng gói UPX trong PEiD 49](#_Toc187856656)

[Hình 44. Signature của trình đóng gói UPX đặc biệt trong PEiD 50](#_Toc187856657)

[Hình 45. Thư mục chứa công cụ UPX 51](#_Toc187856658)

[Hình 46. Giao diện chính của UPX 51](#_Toc187856659)

[Hình 47. Dùng UPX để đóng gói tệp tin 52](#_Toc187856660)

[Hình 48. Trước khi đóng gói bằng UPX 52](#_Toc187856661)

[Hình 49. Sau khi đóng gói bằng UPX 52](#_Toc187856662)

[Hình 50. Giải nén tệp tin bị đóng gói 52](#_Toc187856663)

[Hình 51. Thư mục chứa file cài đặt FSG 53](#_Toc187856664)

[Hình 52. Thư mục chứa các tệp tin thực thi FSG cho từng phiên bản 53](#_Toc187856665)

[Hình 53. Thực thi công cụ FSG 54](#_Toc187856666)

[Hình 54. Chọn tệp tin để đóng gói trong FSG 54](#_Toc187856667)

[Hình 55. Tệp tin được đóng gói bởi FSG 54](#_Toc187856668)

[Hình 56. Thư mục chứa công cụ MEW 55](#_Toc187856669)

[Hình 57. Câu lệnh sử dụng giao diện đồ họa của MEW 56](#_Toc187856670)

[Hình 58. Chọn tệp tin muốn đóng gói trong MEW 56](#_Toc187856671)

[Hình 59. Kết quả đóng gói tệp tin bởi MEW 56](#_Toc187856672)

[Hình 60. Sử dụng câu lệnh để khởi chạy MEW 57](#_Toc187856673)

[Hình 61. Quá trình đóng gói hàng loạt trong MEW 57](#_Toc187856674)

[Hình 62. Thư mục chứa công cụ MPress 58](#_Toc187856675)

[Hình 63. Thực hiện đóng gói tệp tin bằng Mpress 58](#_Toc187856676)

[Hình 64. Thư mục chứa công cụ Unipacker 59](#_Toc187856677)

[Hình 65. Giao diện chính của công cụ Unipacker 60](#_Toc187856678)

[Hình 66. Kiểm tra một tệp tin bị đóng gói bằng Unipacker 60](#_Toc187856679)

[Hình 67. Kiểm tra thông tin tệp trong Unipacker 61](#_Toc187856680)

[Hình 68. Các thông số PE thống kê được trong Unipacker 61](#_Toc187856681)

[Hình 69. Thư viện tĩnh trong PE header 62](#_Toc187856682)

[Hình 70. Thư viện động trong PE header 62](#_Toc187856683)

[Hình 71. Trạng thái của các thanh ghi trong file PE 62](#_Toc187856684)

[Hình 72. Giải nén tệp tin trong Unipacker 63](#_Toc187856685)

[Hình 73. Dockerfile của Unipacker 63](#_Toc187856686)

[Hình 74. Truy cập shell để lấy tệp tin giải nén trong Unipacker 64](#_Toc187856687)

[Hình 75. Mở shell trong Unipacker 64](#_Toc187856688)

[Hình 76. Bắt đầu sử dụng Unipacker như bình thường 65](#_Toc187856689)

[Hình 77. Kết quả thu được tệp tin đã giải nén 65](#_Toc187856690)

[Hình 78. Cài đặt ClamAV 65](#_Toc187856691)

[Hình 79. Cập nhật cơ sở dữ liệu cho ClamAV 66](#_Toc187856692)

[Hình 80. Tiến hành quét tệp tin bằng ClamAV 66](#_Toc187856693)

[Hình 81. Kết quả khi quét tệp tin bằng ClamAV 66](#_Toc187856694)

[Hình 82. Tùy chọn để lấy tệp tin đã giải nén bằng ClamAV 66](#_Toc187856695)

[Hình 83. Đường dẫn đến tệp tin đã giải nén bằng ClamAV 67](#_Toc187856696)

[Hình 84. Tệp tin đã được giải nén bằng ClamAV 67](#_Toc187856697)

[Hình 85. Cài đặt biến môi trường cho pipx 67](#_Toc187856698)

[Hình 86. Cài đặt PEiD bằng pipx 67](#_Toc187856699)

[Hình 87. Quét tệp tin bằng PEiD phiên bản câu lệnh 68](#_Toc187856700)

[Hình 88. Quét tệp tin bằng PEiD phiên bản giao diện người dùng 68](#_Toc187856701)

[Hình 89. Ba chế độ quét trong PEiD 69](#_Toc187856702)

[Hình 90. Thư mục chứa tệp tin cấu hình DiE 69](#_Toc187856703)

[Hình 91. Phân tích tệp tin với DiE phiên bản command line 70](#_Toc187856704)

[Hình 92. Phân tích tệp tin bằng DiE phiên bản GUI 70](#_Toc187856705)

[Hình 93. Phân tích tệp tin bằng DiE phiên bản GUI nhẹ 71](#_Toc187856706)

[Hình 94. Tập dataset 72](#_Toc187856707)

[Hình 95. Đoạn code tự động hóa của công cụ DiE 74](#_Toc187856708)

[Hình 96. Đoạn code tự động hóa của công cụ PEiD 74](#_Toc187856709)

[Hình 97. Đoạn code tự động hóa của công cụ Unipacker 75](#_Toc187856710)

[Hình 98. Đoạn code tự động hóa của công cụ ClamAV 76](#_Toc187856711)

[Hình 99. Lưu kết quả tự động hóa 76](#_Toc187856712)

[Hình 100. Đoạn code đọc các tệp tin trong folder đầu vào 77](#_Toc187856713)

[Hình 101. Hàm lấy folder đầu vào 78](#_Toc187856714)

[Hình 102. Kết quả quét lần lượt các tệp tin 78](#_Toc187856715)

[Hình 103. Hàm đếm các file đã phát hiện được 78](#_Toc187856716)

[Hình 104. Hàm tính tính phần trăm phát hiện packer của các công cụ 79](#_Toc187856717)

[Hình 105. Hàm tính phần trăm chính xác của các công cụ 79](#_Toc187856718)

[Hình 106. Hàm tính thời gian chạy trung bình của các công cụ 79](#_Toc187856719)

[Hình 107. Hàm scan file, thực hiện gọi hàm, tính toán rồi in kết quả ra màn hình 80](#_Toc187856720)

[Hình 108. Thời gian trung bình của các công cụ 81](#_Toc187856721)

# DANH SÁCH BẢNG

[Bảng 1. Các packer mà công cụ có thể phát hiện được 73](#_Toc187796361)

[Bảng 2. Thời gian chạy và độ chính xác của từng công cụ 82](#_Toc187796362)

[Bảng 3. So sánh các công cụ 83](#_Toc187796363)

# TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH

Trong đồ án này, nhóm chúng em sẽ tập trung nghiên cứu các công cụ phát hiện và giải nén các tập tin thực thi Windows khi các tập tin này bị đóng gói để ngăn chặn việc phân tích ngược và che giấu hành vi bất thường.

Trong phạm vi của đồ án, chúng em sẽ tập trung nghiên cứu cấu trúc của các tập tin thực thi Windows (Portable Executable - PE), bao gồm các thành phần chính và cách chúng hoạt động. Đồng thời, nhóm sẽ tìm hiểu cơ chế hoạt động của các trình đóng gói phổ biến, từ đó phân tích cách chúng bảo vệ mã nguồn và che giấu dữ liệu. Để thực hiện nghiên cứu, nhóm sẽ sử dụng bốn công cụ chính, bao gồm: Unipacker, ClamAV, PeiD và Detect It Easy (DIE). Cả bốn công cụ trên đều là những công cụ hỗ trợ phát hiện và phân loại các trình đóng gói (packer) phổ biến. Trong đó, một số công cụ còn cả khả năng giải nén và cho phép truy xuất mã nguồn gốc một cách nhanh chóng và hiệu quả.

Cuối cùng, chúng em sẽ đánh giá hiệu quả của từng công cụ trong việc phát hiện và giải nén, sau đó đưa ra sự so sánh và nhận xét về các công cụ này.

# TỔNG QUAN

## Giới thiệu vấn đề

“*Trong 6 tháng đầu năm 2024, tình hình an toàn thông tin trên toàn cầu tiếp tục diễn biến phức tạp và đáng lo ngại. Các cuộc tấn công mạng gia tăng cả về số lượng và mức độ tinh vi, nhắm vào các cơ quan chính phủ, doanh nghiệp và tổ chức cá nhân. Số lượng các cuộc tấn công trên toàn thế giới đã tăng gấp 1,3 lần so với cùng kỳ 2023, riêng khu vực Việt Nam đã tăng gần gấp 2 lần và để lại nhiều ảnh hưởng nghiêm trọng đến cả uy tín, tài sản của nhiều doanh nghiệp, tổ chức.*” – Trung tâm An toàn thông tin VNPT (VNPT Cyber Immunity) ghi nhận.

Hiện nay, vấn nạn an toàn thông tin ngày càng nghiêm trọng khi các loại mã độc và phần mềm độc hại liên tục được phát triển với những kỹ thuật tinh vi nhằm qua mặt các hệ thống bảo mật. Những cuộc tấn công mạng không chỉ gây tổn hại đến dữ liệu và tài sản số mà còn làm gián đoạn hoạt động của các tổ chức, doanh nghiệp, thậm chí đe dọa an ninh quốc gia. Thiệt hại từ các vụ tấn công mạng có thể lên đến hàng tỷ USD mỗi năm, bao gồm mất dữ liệu, gián đoạn kinh doanh và thiệt hại danh tiếng. Bên cạnh đó, đối với cá nhân, việc mất thông tin cá nhân hay bị xâm phạm quyền riêng tư đang trở thành nỗi lo thường trực trong thời đại mà đâu đâu cũng là Internet. Điều này không chỉ đe dọa sự an toàn của hệ thống thông tin mà còn đặt ra những thách thức lớn trong việc bảo vệ dữ liệu và tài sản trong kỷ nguyên kỹ thuật số. Như lời Robert Mueller, cựu Giám đốc FBI, từng nói: “*There are only two types of companies: those that have been hacked, and those that will be.*”. Câu nói này cũng đã nhấn mạnh mức độ nghiêm trọng và tất yếu của các cuộc tấn công mạng đối với mọi tổ chức và doanh nghiệp.

Một trong những kỹ thuật đã xuất hiện từ khá lâu và được các tin tặc sử dụng rộng rãi để che giấu mã độc là đóng gói (packing). Việc đóng gói không chỉ giúp giảm kích thước tập tin mà còn che giấu mã nguồn, làm phức tạp quá trình phân tích và phát hiện mã độc của các chuyên gia an ninh mạng. Kỹ thuật này tạo ra những rào cản lớn trong việc nhận diện và ngăn chặn các cuộc tấn công mạng, đồng thời tiềm ẩn nguy cơ cao cho hệ thống thông tin của các doanh nghiệp và người dùng cá nhân. Do đó, việc nghiên cứu và phát triển các phương pháp hiệu quả để phát hiện và giải nén các tập tin thực thi đóng gói là vô cùng cần thiết. Việc này không chỉ giúp nâng cao khả năng bảo vệ hệ thống thông tin mà còn giảm thiểu những rủi ro từ các mối đe dọa ngày càng gia tăng khi thế giới đang chuyển đổi số hóa.

## Thách thức và phương pháp

### Thách thức

Trong lĩnh vực an toàn thông tin, việc phát hiện và phân tích các tập tin thực thi đóng gói (packed executables) đặt ra khá nhiều thách thức do tính phức tạp và đa dạng của các kỹ thuật đóng gói. Hiện nay, có hàng trăm loại công cụ đóng gói khác nhau, mỗi loại sử dụng các phương pháp bảo vệ và mã hóa phức tạp, gây khó khăn cho việc nhận diện và phân tích, đòi hỏi các chuyên gia phải có kiến thức sâu rộng và các công cụ hỗ trợ mạnh mẽ để xử lý.

Một trong những thách thức lớn nhất là khả năng che giấu mã độc của các phần mềm đóng gói. Tin tặc thường sử dụng kỹ thuật này để tránh bị các chương trình chống mã độc phát hiện, làm cho quá trình phân tích đòi hỏi nhiều thời gian và công sức hơn. Mỗi bước phân tích đều cần được thực hiện cẩn thận để đảm bảo rằng mã độc được trích xuất chính xác và đầy đủ. Ngoài ra, nhiều công cụ đóng gói còn tích hợp các cơ chế chống dịch ngược (anti-reverse engineering), khiến các nhà phân tích gặp khó khăn hơn trong việc trích xuất thông tin từ tập tin thực thi. Những cơ chế này không chỉ làm giảm hiệu quả của các công cụ phân tích mà còn làm tăng nguy cơ bỏ sót mã độc.

Và trên hết, việc phân tích và giải nén các tập tin thực thi là một quá trình phức tạp, đòi hỏi sự kết hợp chặt chẽ giữa tài nguyên, công nghệ và kỹ năng chuyên môn cao. Các chuyên gia cần có kiến thức chuyên sâu, cũng như quyền truy cập vào các công cụ phân tích chuyên biệt. Tuy nhiên, không phải tổ chức nào cũng có đủ điều kiện để sở hữu những công cụ này. Sự thiếu hụt tài nguyên và chuyên môn không chỉ làm chậm quá trình phản ứng trước các cuộc tấn công mạng mà còn tạo ra những lỗ hổng bảo mật nghiêm trọng, hình thành một rào cản đáng kể trong việc bảo vệ hệ thống thông tin trước các mối đe dọa an ninh mạng.

### Phương pháp

Để vượt qua các thách thức trên, nhóm chúng em sẽ sử dụng bốn công cụ chính, bao gồm: Unipacker, PEiD, Detect It Easy (DIE) và ClamAV để phát hiện, giải nén và phân tích các tập tin thực thi Windows bị đóng gói.

Mỗi công cụ đều có những ưu và nhược điểm khác nhau. Unipacker sẽ được sử dụng để phát hiện và giải nén các tập tin thực thi đã bị đóng gói, giúp trích xuất mã gốc từ các tập tin này. Công cụ này có khả năng giải nén một số loại đóng gói phổ biến, giúp tiết kiệm thời gian khi phân tích. PEiD sẽ hỗ trợ phát hiện loại packer được sử dụng, từ đó cung cấp thông tin quan trọng để xác định phương thức bảo vệ mã nguồn. Detect It Easy (DIE) là một công cụ mạnh mẽ giúp nhận diện các loại mã độc và packer thông qua phân tích chi tiết tệp, tìm ra các mẫu và đặc điểm đáng ngờ. DIE có thể giúp phát hiện các dạng packer ít phổ biến, điều mà PEiD đôi khi bỏ qua. Cuối cùng, ClamAV, một phần mềm diệt virus mã nguồn mở, sẽ được sử dụng để kiểm tra sự hiện diện của các mã độc và các trình đóng gói có trong các tập tin đã được giải nén, hỗ trợ phân tích bảo mật và phát hiện các mối đe dọa tiềm ẩn.

## Mục tiêu, nội dung cụ thể

### Mục tiêu

Mục tiêu chính của bài báo cáo này là so sánh chức năng và hiệu suất của bốn công cụ Unipacker, PEiD, Detect It Easy (DIE) và ClamAV trong việc phát hiện và giải nén các tập tin thực thi Windows bị đóng gói (packed). Cụ thể, báo cáo sẽ tập trung vào việc đánh giá khả năng của từng công cụ trong việc nhận diện các loại packer được sử dụng trong các tập tin thực thi, từ đó xác định mức độ hiệu quả của từng công cụ trong việc phát hiện packer. Mỗi công cụ sẽ được kiểm tra trong khả năng nhận diện và xử lý các packer phổ biến và ít phổ biến để xem xét độ chính xác của chúng trong việc xác định phương thức bảo vệ mã nguồn.

Bên cạnh đó, nhóm em cũng sẽ so sánh hiệu quả giải nén của các công cụ, bao gồm khả năng phục hồi mã nguồn và độ chính xác của quá trình giải nén các tập tin đóng gói. Việc đánh giá hiệu suất của từng công cụ sẽ không chỉ xem xét tốc độ xử lý mà còn kiểm tra độ chính xác của các kết quả giải nén. Mục tiêu cuối cùng là cung cấp một cái nhìn tổng quan về các công cụ này, giúp xác định công cụ phù hợp cho từng trường hợp cụ thể trong việc phát hiện và giải nén các tập tin thực thi đóng gói, từ đó hỗ trợ việc bảo vệ hệ thống thông tin hiệu quả hơn.

### Nội dung cụ thể

**Nội dung 1:** Tìm hiểu về tập tin thực thi Windows

Tập tin thực thi Windows, còn được gọi là file PE (Portable Executable), là định dạng tập tin được sử dụng phổ biến trên hệ điều hành Windows để chạy ứng dụng hoặc có thể là mã độc. File PE chứa thông tin về mã máy, thư viện động (DLL), và các tài nguyên khác cần thiết để thực thi ứng dụng. Tìm hiểu về file PE rất quan trọng trong việc phân tích an ninh mạng, bởi vì nó giúp các chuyên gia nhận diện và phân tích các tập tin thực thi bị đóng gói, từ đó hiểu được cách thức hoạt động của các mối đe dọa an ninh mạng và thực hiện các biện pháp bảo vệ hệ thống.

**Nội dung 2:** Tìm hiểu các trình đóng gói packer hiện có trên thị trường

Trình đóng gói (packer) là công cụ dùng để nén hoặc mã hóa các tập tin thực thi, nhằm bảo vệ mã nguồn và ngăn ngừa phân tích ngược. Một số packer phổ biến hiện nay bao gồm UPX, ASPack, FSG, .... Các packer này sử dụng các kỹ thuật như mã hóa hoặc thay đổi cấu trúc tập tin để che giấu nội dung mã độc, khiến việc phát hiện và phân tích trở nên khó khăn hơn. Việc hiểu và nắm vững các loại packer giúp nâng cao khả năng phát hiện và xử lý các mối đe dọa an ninh mạng.

**Nội dung 3:** Tìm hiểu và triển khai các công cụ phát hiện và giải nén tập tin thực thi Windows.

Để thực hiện nội dung này, nhóm sẽ triển khai các công cụ phát hiện và giải nén tập tin thực thi Windows đã giới thiệu trước đó. Ở bước này, nhóm sẽ chỉ kiểm tra thử các chức năng mà mỗi công cụ đang có, xem xét xem từng công cụ có những khả năng gì chứ chưa triển khai so sánh chúng. Mục tiêu là hiểu rõ chức năng của từng công cụ trước khi tiến hành đánh giá và so sánh hiệu quả của chúng trong các bước tiếp theo.

**Nội dung 4:** So sánh chức năng và hiệu suất của cả 4 công cụ trong việc phát hiện và giải nén các tập tin thực thi Windows.

Trong bước này, nhóm sẽ so sánh chức năng và hiệu suất của bốn công cụ Unipacker, PEiD, Detect It Easy (DIE) và ClamAV trong việc phát hiện và giải nén các tập tin thực thi Windows bị đóng gói. Cụ thể, nhóm sẽ đánh giá khả năng nhận diện các loại packer của mỗi công cụ, sự chính xác trong việc giải nén mã nguồn và tốc độ xử lý các tập tin đóng gói. Qua đó, nhóm sẽ đưa ra cái nhìn tổng quan về hiệu quả và ứng dụng của từng công cụ trong các tình huống thực tế.

# CƠ SỞ LÝ THUYẾT

## Tập tin thực thi Windows

Tập thực thi Windows (Windows Executable) là các tệp chứa mã lệnh và dữ liệu cần thiết để hệ điều hành Windows nạp và thực thi một chương trình. Trong đó, file Portable Executable (PE) là một định dạng tiêu chuẩn được Windows sử dụng. Hãy cùng tìm hiểu một file PE sẽ có những gì.

### Portable Executable – PE là gì?

File **Portable Executable (PE)** là định dạng tệp nhị phân gốc của hệ điều hành Windows. Nó được thiết kế để lưu trữ các chương trình thực thi (.exe), thư viện liên kết động (.dll), kernel modules (.srv), và các loại tệp nhị phân khác. Định dạng này có tính độc lập nền tảng, nghĩa là nó sẽ cho phép các tệp PE hoạt động trên bất kỳ máy Windows nào, miễn là máy đó chạy đúng phiên bản hệ điều hành và kiến trúc bộ xử lý (x86 hoặc x64) mà tệp được biên dịch.

Những trước khi khám phá định dạng Portable Executable trên Windows, hãy làm rõ một số khái niệm cơ bản. Hãy lùi lại một bước và tìm hiểu về khái niệm nền tảng của COFF - nguồn gốc của Portable Executable (PE).

Khi bạn biên dịch mã nguồn của một chương trình, trình biên dịch sẽ chuyển đổi mã nguồn của một chương trình thành mã máy, nó tạo ra một tệp đối tượng (.obj). Tệp đối tượng này chứa các lệnh dưới dạng nhị phân để máy tính thực thi. Và **Common Object File Format (COFF)** là một tập hợp tiêu chuẩn dùng để biểu diễn các lệnh nhị phân đó. COFF giúp duy trì khả năng tương thích đa nền tảng thông qua việc tổ chức mã và dữ liệu theo các quy tắc chung, giúp chương trình dễ dàng được liên kết và thực thi trên nhiều hệ thống khác nhau. COFF ban đầu được phát triển cho các hệ thống \*NIX, nhưng hiện nay nó phổ biến trên hầu hết các nền tảng, bao gồm cả Windows và Linux.

Định dạng PE là một phiên bản sửa đổi của COFF, được phát triển đặc biệt cho hệ điều hành Windows. Trong khi COFF cung cấp một định dạng tiêu chuẩn cho các tệp đối tượng (.obj), thì PE mở rộng khả năng này để cung cấp một định dạng tiêu chuẩn cho các tệp thực thi (.exe), thư viện liên kết động (.dll), và các tệp nhị phân liên quan. Điểm khác biệt chính giữa hai định dạng là PE được thiết kế để hoạt động độc quyền trên các hệ thống Windows 32-bit (PE32) và 64-bit (PE32+), phù hợp với kiến trúc và yêu cầu của hệ điều hành này.

### Cấu trúc file PE

Định dạng file Portable Executable (PE) bao gồm nhiều thành phần và mỗi thành phần có một mục đích cụ thể. Các thành phần này bao gồm:

A screen shot of a computer

Description automatically generated

Hình . Cấu trúc một file PE

1. **DOS Header**

Mỗi tệp PE đều bắt đầu với một cấu trúc dài 64 byte gọi là DOS Header. Thành phần này giúp tệp PE được nhận dạng như một tệp thực thi MS-DOS. Mặc dù hệ điều hành Windows hiện đại không còn phụ thuộc vào MS-DOS, nhưng phần này vẫn được giữ lại để đảm bảo tính tương thích ngược và hỗ trợ các công cụ phân tích, đọc tệp PE.

DOS Header bao gồm các trường dữ liệu quan trọng, trong đó phổ biến nhất là:

* e\_magic (2 byte): là magic number xác định tệp thuộc định dạng DOS. Giá trị thông thường là MZ, đại diện cho tên của Mark Zbikowski, người phát triển định dạng tệp DOS ban đầu.
* e\_lfanew (4 byte): chứa offset đến PE Header. Trường này luôn nằm tại vị trí offset 0x3C và có kiểu dữ liệu “long”, vì vậy nó có kích thước 32 bit (4 byte). Đây là trường quan trọng nhất trong DOS Header, giúp hệ điều hành hoặc công cụ xử lý nhảy đến vị trí bắt đầu của PE Header trong tệp.

Ngoài ra, DOS Header còn chứa các trường khác cung cấp thông tin về định dạng và cách thực thi tệp trong môi trường MS-DOS, nhưng phần lớn chúng không được sử dụng trên hệ thống Windows hiện đại.

1. **DOS Stub**

DOS Stub là một đoạn mã nhỏ nằm ngay sau DOS Header trong tệp PE. Đoạn mã này được thiết kế để hoạt động trên các hệ thống MS-DOS và có nhiệm vụ xử lý tình huống khi tệp PE được chạy trong môi trường không hỗ trợ định dạng PE, chẳng hạn như MS-DOS hoặc các hệ điều hành cũ không phải Windows. Khi chương trình được chạy trong chế độ DOS, DOS stub sẽ hiển thị một thông báo lỗi:

"*This program cannot be run in DOS mode*"

(Chương trình này không thể chạy trong chế độ DOS).

1. **NT Header**

NT Header là một thành phần quan trọng trong cấu trúc tệp PE, chứa các thông tin cần thiết để hệ điều hành Windows có thể nạp và thực thi tệp. NT Header nằm ngay sau DOS Stub và bao gồm ba phần chính:

* **PE Signature**: là một giá trị 4 byte xác định tệp là tệp Portable Executable. Giá trị của PE Signature là 0x00004550 ("PE\0\0"), giúp phân biệt tệp PE với các định dạng tệp khác.
* **File Header**: một tiêu đề COFF tiêu chuẩn, cung cấp thông tin cơ bản về tệp PE để hệ điều hành hiểu cách xử lý tệp. PE Header có các trường quan trọng như**:**
  + Machine: Loại kiến trúc bộ xử lý (ví dụ: x86, x64).
  + NumberOfSections: Số lượng phần (sections) trong tệp.
  + TimeDateStamp: Thời gian tệp được tạo.
  + SizeOfOptionalHeader: Kích thước của Optional Header.
  + Characteristics: Các đặc điểm của tệp (ví dụ: tệp là thực thi, thư viện liên kết động, v.v.).
* **Optional Header:** mặc dù được gọi là "Optional Header" (tiêu đề tùy chọn) nhưng phần này lại bắt buộc đối với các tệp hình ảnh (Image Only) (như tệp .exe). Đây là phần quan trọng nhất của NT Header. Optional Header có các trường quan trọng như:
  + AddressOfEntryPoint: Điểm bắt đầu thực thi của tệp.
  + ImageBase: Địa chỉ cơ sở nơi hình ảnh sẽ được nạp vào bộ nhớ.
  + SectionAlignment: Cách các phần được căn chỉnh trong bộ nhớ.
  + Subsystem: Loại môi trường yêu cầu để chạy tệp (ví dụ: GUI hoặc console).
  + SizeOfImage: Tổng kích thước của tệp khi được nạp vào bộ nhớ

NT Header luôn bắt đầu tại offset được chỉ định bởi trường “e\_lfanew” trong DOS Header. Điều này đảm bảo rằng hệ điều hành có thể dễ dàng định vị NT Header dù kích thước của DOS Header hoặc DOS Stub thay đổi.

1. **Section Table**

Section Table là một phần quan trọng của tệp PE, nằm ngay sau Optional Header trong cấu trúc của NT Header. Section Table là một mảng các mục, mỗi mục tương ứng với một Section Header. Mỗi Section Header cung cấp thông tin chi tiết về một phần (section) trong tệp PE, chẳng hạn như vị trí, kích thước, và các thuộc tính của phần đó.

1. *Vai trò của Section Header*

Section Header đóng vai trò quan trọng trong việc quản lý các phần (sections) của tệp PE. Tệp PE được chia thành nhiều phần khác nhau như mã thực thi, dữ liệu tĩnh, tài nguyên, và thông tin gỡ lỗi. Section Header cung cấp thông tin chi tiết về từng phần này, giúp hệ điều hành ánh xạ chúng vào bộ nhớ và sử dụng đúng cách. Nhờ đó, các thành phần của tệp PE được tổ chức và vận hành hiệu quả.

Ngoài ra, Section Header còn hỗ trợ hệ điều hành xác định các thuộc tính của từng phần. Mỗi mục trong Section Header mô tả đặc điểm của một phần cụ thể, chẳng hạn như phần đó có thể thực thi, chỉ đọc, hay có thể ghi được. Điều này cho phép hệ điều hành áp dụng các quyền truy cập thích hợp, đảm bảo rằng tệp PE hoạt động một cách an toàn và chính xác.

1. *Cấu trúc của Section Header*

Mỗi Section Header trong Section Table chứa các trường dữ liệu quan trọng, bao gồm:

* Name (8 byte): Tên của phần, thường được đặt ngắn gọn như .text, .data, hoặc .rdata.
* VirtualSize (4 byte): Kích thước thực sự của phần khi được nạp vào bộ nhớ.
* VirtualAddress (4 byte): Địa chỉ tương đối của phần trong không gian địa chỉ ảo.
* SizeOfRawData (4 byte): Kích thước của phần trong tệp PE.
* PointerToRawData (4 byte): Offset (độ dời) từ đầu tệp đến dữ liệu thực tế của phần này.
* Characteristics (4 byte): Thuộc tính của phần, chẳng hạn như có thể thực thi, chỉ đọc, hoặc ghi được.

1. *Các thành phần phổ biến trong Section Header*

Dưới đây là một số phần thường thấy trong tệp PE và vai trò của chúng:

* .text: Chứa mã thực thi của chương trình. Đây là phần chính mà CPU sẽ thực thi.
* .data: Chứa dữ liệu toàn cục, có thể được thay đổi trong quá trình chạy chương trình.
* .rdata: Chứa dữ liệu chỉ đọc, chẳng hạn như chuỗi văn bản hằng số.
* .rsrc: Chứa tài nguyên của chương trình, chẳng hạn như biểu tượng, hộp thoại, hoặc tệp nhúng.
* .reloc: Chứa thông tin về các mục cần được sửa đổi nếu tệp PE được nạp vào một địa chỉ khác với địa chỉ mặc định.

1. *Cách hoạt động của Section Table*

Khi hệ điều hành nạp tệp PE vào bộ nhớ, nó sẽ bắt đầu bằng việc đọc Section Table để hiểu cách thức phân chia tệp thành các phần khác nhau. Section Table cung cấp thông tin về từng phần của tệp, bao gồm vị trí, kích thước và các đặc tính của các phần như mã thực thi, dữ liệu, tài nguyên, ... Dựa trên thông tin này, hệ điều hành sẽ ánh xạ các phần của tệp vào bộ nhớ với các địa chỉ tương ứng, giúp chương trình có thể sử dụng tài nguyên và dữ liệu đúng cách khi thực thi.

Sau khi ánh xạ các phần của tệp vào bộ nhớ, hệ điều hành tiếp tục xử lý các quyền truy cập đối với từng phần dựa trên các thuộc tính được mô tả trong Section Header. Các thuộc tính này có thể xác định liệu phần đó có thể chỉ đọc, ghi hoặc thực thi được. Hệ điều hành sẽ áp dụng các quyền truy cập này để bảo vệ bộ nhớ và đảm bảo rằng chỉ những phần cần thiết mới có thể bị thay đổi hoặc thực thi, giúp duy trì tính an toàn và hiệu quả của chương trình.

Ngoài ra, các phần như .rsrc, chứa tài nguyên của chương trình, sẽ được hệ điều hành xử lý đặc biệt để quản lý các tài nguyên như biểu tượng, chuỗi văn bản hay giao diện người dùng. Hệ điều hành sử dụng thông tin từ Section Table để truy xuất và cung cấp các tài nguyên này cho chương trình khi cần, đảm bảo rằng các tài nguyên được sử dụng đúng lúc và không bị trùng lặp hoặc sai sót.

1. **Section**

Và cuối cùng, trong file PE, Section là một phần cơ bản và quan trọng chứa các dữ liệu thực thi hoặc các tài nguyên cần thiết để chương trình hoạt động. Mỗi phần trong tệp PE có mục đích riêng biệt, từ mã thực thi, dữ liệu chương trình cho đến tài nguyên và thông tin gỡ lỗi. Các phần này được quản lý bởi hệ điều hành thông qua Section Table, nơi lưu trữ các thông tin mô tả về từng phần.

1. *Các loại Section phổ biến trong tệp PE*

* **.text**  
  Đây là phần chứa mã thực thi của chương trình. Khi tệp PE được nạp vào bộ nhớ, phần này sẽ được ánh xạ vào vùng bộ nhớ có thể thực thi. “.text” sẽ chứa các chỉ thị mà CPU sẽ thực hiện khi chương trình chạy.
* **.data**  
  Phần này chứa dữ liệu toàn cục mà chương trình sử dụng và có thể thay đổi trong suốt quá trình thực thi. Các biến toàn cục, dữ liệu cần lưu trữ tạm thời trong khi chương trình đang chạy thường được lưu trữ trong phần “.data”.
* **.rdata**  
  “.rdata” chứa các dữ liệu chỉ đọc, chẳng hạn như chuỗi văn bản hằng số hoặc thông tin cấu hình mà chương trình không cần thay đổi trong suốt quá trình thực thi.
* **.rsrc**  
  Đây là phần chứa các tài nguyên mà chương trình cần, chẳng hạn như biểu tượng, đồ họa, văn bản giao diện người dùng, hoặc các tệp nhúng khác. “.rsrc” giúp chương trình truy xuất và sử dụng các tài nguyên này khi cần thiết.
* **.reloc**  
  “.reloc” chứa thông tin về các địa chỉ cần được điều chỉnh nếu tệp PE được nạp vào một vị trí bộ nhớ không phải là vị trí mặc định, giúp chương trình hoạt động trên các địa chỉ bộ nhớ khác nhau mà không gặp sự cố.

1. *Cấu trúc Section*

Mỗi section trong tệp PE được mô tả bởi một Section Header trong Section Table. Mỗi Section Header cung cấp các thông tin quan trọng về phần đó, bao gồm:

* Tên phần (Name): Một chuỗi ký tự (thường là 8 byte) dùng để xác định phần, ví dụ như .text, .data.
* Kích thước của phần trong bộ nhớ ảo (VirtualSize): Kích thước thực tế của phần khi nó được nạp vào bộ nhớ ảo.
* Địa chỉ bộ nhớ ảo (VirtualAddress): Địa chỉ nơi phần này sẽ được nạp vào bộ nhớ.
* Kích thước phần trong tệp (SizeOfRawData): Kích thước của phần khi lưu trữ trong tệp PE, trước khi được nạp vào bộ nhớ.
* Offset tới dữ liệu thực tế trong tệp (PointerToRawData): Độ dời từ đầu tệp đến vị trí chứa dữ liệu thực tế của phần này trong tệp.
* Thuộc tính phần (Characteristics): Các đặc điểm của phần, ví dụ như có thể thực thi, chỉ đọc, hay có thể ghi được.

1. *Vai trò của Section*

* Chứa mã thực thi: Phần “.text” chứa mã máy mà hệ điều hành sẽ thực thi khi chương trình được khởi động.
* Chứa dữ liệu và tài nguyên: Các phần như .data và .rsrc lưu trữ dữ liệu toàn cục và tài nguyên cần thiết cho chương trình, giúp chương trình dễ dàng truy xuất và sử dụng tài nguyên trong quá trình thực thi.
* Tăng tính mở rộng: Các phần như “.reloc” và “.rsrc” giúp chương trình có thể hoạt động linh hoạt hơn trong các môi trường khác nhau, ví dụ, khi được nạp vào các địa chỉ bộ nhớ khác nhau.
* Quản lý bộ nhớ: Các thuộc tính trong các Section Header giúp hệ điều hành quyết định quyền truy cập của từng phần, từ đó tăng tính bảo mật và hiệu quả trong việc quản lý bộ nhớ.

## Trình đóng gói

### Trình đóng gói (Packer) là gì?

Trình đóng gói (Packer) thường được hiểu là các công cụ nén tệp có khả năng giải nén tự động khi tệp được thực thi, đặc biệt là trong môi trường runtime (runtime packer). Cụ thể, Packer sẽ nén tệp thực thi và giải nén nó trong bộ nhớ khi chương trình được khởi chạy. Kỹ thuật này còn được gọi là "nén thực thi" (executable compression), với mục đích làm cho tệp nhỏ hơn, giúp giảm bớt dung lượng bộ nhớ cần thiết khi tải xuống và truyền tải các tệp qua internet. Điều này mang lại lợi ích về mặt giảm kích thước tệp, từ đó tiết kiệm dung lượng ổ đĩa và băng thông mạng, đồng thời tránh yêu cầu người dùng phải giải nén tệp thủ công trước khi chạy chương trình.

Trước đây, với kích thước tệp lớn và kết nối internet chậm, việc nén tệp thực thi trở nên vô cùng quan trọng, vì nó giúp tiết kiệm không gian lưu trữ và giảm thiểu thời gian tải tệp từ internet. Tuy nhiên, trong thời đại hiện nay, với sự phát triển của công nghệ lưu trữ và tốc độ internet ngày càng nhanh, nhu cầu về việc nén tệp thực thi không còn quan trọng như trước. Kích thước của các tệp dữ liệu đa phương tiện đã lớn hơn rất nhiều, và việc nén tệp không còn mang lại lợi ích lớn như trước đây. Do đó, ngày nay, khi gặp phải các tệp thực thi đã được nén hoặc đóng gói, người ta thường nghi ngờ rằng mục đích sử dụng Packer không còn chỉ để giảm kích thước tệp, mà là để bảo vệ mã nguồn hoặc che giấu phần mềm độc hại.

Hiện nay, Packer chủ yếu được sử dụng với mục đích bảo vệ các chương trình độc hại, giúp chúng dễ dàng vượt qua các công cụ phân tích và phát hiện phần mềm độc hại. Việc sử dụng Packer giúp cho việc dịch ngược mã trở nên khó khăn hơn, làm tăng độ phức tạp khi nghiên cứu và phân tích mã nguồn. Đây là một lợi thế đặc biệt cho các tác giả phần mềm độc hại, khi họ muốn ẩn mã độc (payload) trong tệp thực thi nén, giảm thiểu khả năng bị phát hiện. Các dấu vết liên quan đến mã độc, như kích thước của phần mềm hoặc các manh mối về hành vi của phần mềm độc hại, sẽ trở nên nhỏ hơn và khó bị phát hiện trên máy bị nhiễm.

Với sự trợ giúp của kỹ thuật nén và đóng gói này, phần mềm độc hại có thể tồn tại lâu dài hơn trên các hệ thống bị tấn công mà không bị phát hiện, khiến việc xử lý và tiêu diệt phần mềm độc hại trở nên khó khăn hơn cho các chuyên gia an ninh mạng.

### Cách đóng gói tập tin

Về bản chất, các trình đóng gói hiện nay được xem như những công cụ được sử dụng để che giấu tập tin độc hại. Các trình đóng gói này có thể mã hóa, nén hoặc đơn giản là thay đổi định dạng của tệp phần mềm độc hại để làm cho nó trông giống như một thứ hoàn toàn khác. Quá trình nén hoặc mã hóa của trình đóng gói đưa tập tin từ mã gốc của nó sang trạng thái mới bằng cách sử dụng các kỹ thuật làm rối "thử và đúng" (tried-and-true obfuscation).

A screen shot of a diagram

Description automatically generated

Hình . Một tập tin thực thi đã bị đóng gói

*Vậy kỹ thuật làm rối Tried-and-True Obfuscation là gì?*

Kỹ thuật làm rối là một phần quan trọng trong hoạt động của trình đóng gói, được sử dụng để che giấu mã độc và gây khó khăn cho việc phân tích hoặc phát hiện. "Tried-and-True" ở đây ám chỉ việc áp dụng những phương pháp đã được kiểm chứng qua thời gian và chứng minh hiệu quả trong việc làm phức tạp quá trình phân tích mã. Dưới đây là một số kỹ thuật nổi bật được trình đóng gói sử dụng để làm rối mã độc:

**1. Xáo trộn mã (Code Scrambling)**

Kỹ thuật xáo trộn mã là quá trình thay đổi trật tự các lệnh hoặc thêm các đoạn mã thừa vào mã nguồn ban đầu, nhằm tạo ra sự rối rắm, bao gồm một số kỹ thuật như:

* Thay đổi trật tự lệnh: Các lệnh quan trọng của mã độc có thể được sắp xếp lại một cách phi logic hoặc được lồng ghép với các lệnh không liên quan. Điều này làm cho mã nguồn trở nên khó hiểu, khiến việc phân tích mã trở thành một thách thức lớn.
* Thêm mã thừa: Chèn các đoạn mã không có chức năng thực tế, ví dụ như vòng lặp vô dụng hoặc các lệnh không hoạt động. Các đoạn mã này làm tăng độ dài mã nguồn, tạo cảm giác phức tạp, và khiến các nhà phân tích phải mất thời gian loại bỏ chúng để tìm ra mã thực sự quan trọng.

Ví dụ: Một đoạn mã thực hiện hành vi độc hại có thể được "lồng ghép" với hàng trăm dòng mã vô dụng, gây khó khăn cho việc tách biệt và phân tích.

**2. Sử dụng mã hóa phức tạp (Advanced Encryption)**

Mã hóa là một trong những kỹ thuật mạnh mẽ nhất được sử dụng để làm rối mã độc. Hiện nay có rất nhiều cách để thực hiện mã hóa một chương trình, tuy nhiên, việc sử dụng những loại mã hóa đơn giản thường không khó để có thể giải mã. Vì vậy, một số cách mã hóa phức tạp ra đời, không chỉ làm rối mã độc mà còn đảm bảo rằng ngay cả khi tệp bị phát hiện, việc phân tích nội dung bên trong cũng trở nên rất khó khăn.

* Mã hóa nhiều lớp: Thay vì chỉ mã hóa một lần, mã độc có thể được mã hóa nhiều lớp, mỗi lớp sử dụng một thuật toán hoặc khóa mã hóa khác nhau. Khi thực thi, từng lớp mã hóa sẽ được giải mã một cách tuần tự, cuối cùng trả về mã gốc.
* Thuật toán khó phá giải: Sử dụng các thuật toán mã hóa tiên tiến, phức tạp để làm cho việc giải mã thủ công trở nên gần như không thể nếu không có khóa giải mã chính xác.

**3. Tự sửa đổi mã (Self-Modifying Code)**

Tự sửa đổi mã là một kỹ thuật độc đáo và hiệu quả trong việc làm rối, nơi mã độc có khả năng thay đổi chính nó trong quá trình thực thi. Về cách kỹ thuật này hoạt động, khi được thực thi, mã độc tự thay đổi cấu trúc hoặc nội dung của nó để tránh bị nhận diện. Mã độc có thể tự điều chỉnh cách thức hoạt động của mình tùy thuộc vào môi trường thực thi, ví dụ như tránh hoạt động trên máy ảo để tránh bị phân tích. Một ví dụ khác, một đoạn mã ban đầu có thể tự giải mã, thay đổi hành vi hoặc che giấu các lệnh quan trọng khi được phân tích.

Bằng cách tự sửa đổi, mã độc có thể vượt qua các công cụ phân tích tĩnh (static analysis) thường chỉ hoạt động trên mã cố định. Kỹ thuật tự sửa đổi tạo ra một lớp bảo vệ bổ sung cho mã độc, làm tăng độ khó trong việc nghiên cứu và phân tích.Việc này buộc các nhà phân tích phải sử dụng các phương pháp phân tích động (dynamic analysis) phức tạp hơn. Kỹ thuật này được sử dụng để làm cho mã độc trở nên "linh hoạt," tức là nó có thể hoạt động khác nhau trong các môi trường khác nhau, làm tăng độ khó khi phân tích.

**4. Kết hợp các kỹ thuật để tăng hiệu quả**

Thông thường, các tác giả mã độc sẽ không chỉ sử dụng một kỹ thuật duy nhất mà kết hợp nhiều kỹ thuật làm rối để đạt hiệu quả tối ưu. Ví dụ: Một mã độc có thể được mã hóa nhiều lớp, sau đó thêm các đoạn mã thừa và sử dụng mã tự sửa đổi khi thực thi. Việc kết hợp nhiều kỹ thuật khiến cho quá trình phân tích mã trở nên khó khăn và tốn kém hơn, đồng thời làm giảm hiệu quả của các hệ thống bảo mật dựa trên mẫu nhận diện (signature-based detection).

Nhờ các kỹ thuật làm rối "thử và đúng," các trình đóng gói đã giúp mã độc trở nên khó phát hiện và khó phân tích hơn. Các công cụ bảo mật phải đối mặt với những thách thức lớn hơn khi cố gắng phân biệt giữa mã hợp pháp và mã độc hại, làm tăng nguy cơ tấn công thành công và gây thiệt hại lớn hơn trong các hệ thống mục tiêu.

### Các công cụ đóng gói phổ biến

* 1. **UPX**

A screenshot of a computer screen

Description automatically generated

Hình . Công cụ nén tệp tin UPX

UPX (Ultimate Packer for Executables) là một công cụ nén tệp thực thi mạnh mẽ và tiên tiến. Với khả năng giảm kích thước tệp của các chương trình và thư viện DLL xuống từ 50% đến 70%, UPX mang lại nhiều lợi ích như tiết kiệm không gian lưu trữ, giảm thời gian tải qua mạng, rút ngắn thời gian tải xuống, và tối ưu hóa chi phí phân phối cũng như lưu trữ.

Điểm nổi bật của UPX là các chương trình và thư viện được nén bởi công cụ này vẫn hoàn toàn tự chứa, không cần phần mềm bổ sung để chạy và hoạt động chính xác như phiên bản gốc. Đặc biệt, việc nén không gây ra bất kỳ tác động nào đến hiệu suất runtime hoặc bộ nhớ đối với hầu hết các định dạng mà UPX hỗ trợ. UPX hỗ trợ nhiều định dạng thực thi khác nhau, bao gồm các chương trình và thư viện DLL trên hệ điều hành Windows, ứng dụng trên macOS, và tệp thực thi trên Linux. Điều này giúp UPX trở thành một công cụ linh hoạt, phù hợp với nhiều nền tảng và nhu cầu khác nhau.

Là một phần mềm mã nguồn mở, UPX được phân phối miễn phí theo Giấy phép Công cộng GNU (GNU General Public License). Mã nguồn đầy đủ của UPX được cung cấp công khai, cho phép người dùng tùy chỉnh hoặc tích hợp theo nhu cầu. UPX có thể được sử dụng tự do, bao gồm cả trong các ứng dụng thương mại, khiến nó trở thành một giải pháp lý tưởng cho cả cá nhân và doanh nghiệp.

**Các ưu điểm nổi bật của UPX:**

* Bảo mật: UPX là phần mềm mã nguồn mở được công khai và tài liệu hóa trong nhiều năm, do đó các phần mềm bảo mật hoặc phần mềm diệt virus đáng tin cậy có thể kiểm tra bên trong các ứng dụng được nén bởi UPX để xác minh tính an toàn.
* Tỷ lệ nén xuất sắc: UPX thường nén hiệu quả hơn so với định dạng Zip. Công cụ này giúp giảm kích thước các tệp phân phối một cách đáng kể, tối ưu hóa dung lượng lưu trữ và thời gian tải xuống.
* Giải nén cực nhanh: Tốc độ giải nén đạt hơn 500 MB/giây trên bất kỳ máy tính hiện đại nào.
* Không tiêu tốn bộ nhớ: Các tệp thực thi nén bởi UPX được giải nén trực tiếp tại chỗ, không gây bất kỳ ảnh hưởng nào đến dung lượng bộ nhớ.
* An toàn: UPX cho phép liệt kê, kiểm tra và giải nén các tệp thực thi. Đồng thời, nó duy trì checksum (mã kiểm tra) của cả tệp đã nén và tệp gốc không nén.
* Tính phổ quát: UPX hỗ trợ nhiều định dạng thực thi khác nhau, bao gồm các chương trình và thư viện DLL trên Windows, ứng dụng trên macOS, và tệp thực thi trên Linux.
* Tính di động: UPX được viết bằng ngôn ngữ C++ có khả năng tương thích tốt giữa các kiến trúc máy tính, không bị ảnh hưởng bởi sự khác biệt về endian (cách lưu trữ byte).
* Khả năng mở rộng: Với cách tổ chức lớp (class layout), việc thêm định dạng tệp thực thi mới hoặc thuật toán nén mới vào UPX trở nên dễ dàng.
* Miễn phí: UPX được phát hành kèm mã nguồn đầy đủ theo điều khoản của Giấy phép Công cộng GNU phiên bản 2 trở lên (GPLv2+). Người dùng có thể sử dụng miễn phí, bao gồm cả trong các chương trình thương mại, theo các quy định cụ thể trong Thỏa thuận Giấy phép UPX.

Với những tính năng vượt trội trên, không khó hiểu tại sao UPX được gọi là "Ultimate Packer for Executables" (Trình đóng gói tối ưu cho các tệp thực thi). Và đây cũng là một trong những packer được sử dụng phổ biến nhất hiện nay.

**Một số nhược điểm của UPX:**

* Giới hạn kích thước tệp: UPX hoạt động tốt với các tệp thực thi nhỏ và vừa, nhưng khi nén các tệp lớn, hiệu quả nén có thể giảm đi, hoặc thậm chí gây ra lỗi nếu hệ thống không đủ tài nguyên để xử lý.
* Giới hạn trong môi trường nhúng: Trên các thiết bị nhúng hoặc hệ thống có tài nguyên hạn chế, việc nén/giải nén liên tục có thể gây ra vấn đề hiệu năng hoặc làm tăng độ trễ.
* Tương thích hạn chế: Một số phần mềm yêu cầu kiểm tra toàn vẹn (integrity check) có thể không hoạt động chính xác nếu sử dụng UPX. Ngoài ra, các chương trình sử dụng các kỹ thuật nén hoặc mã hóa đặc biệt khác có thể gặp xung đột khi kết hợp với UPX.
  1. **ASPack**



Hình . Công nghệ nén tệp tin ASPack

ASPack là một giải pháp tiên tiến dành cho việc nén tệp thực thi Win32 (EXE) và bảo vệ chúng trước các hoạt động dịch ngược mã không chuyên. Công cụ này giúp giảm kích thước chương trình và thư viện Windows lên đến 70%, với tỷ lệ nén cao hơn tiêu chuẩn ZIP từ 10-20%, không chỉ tối ưu hóa dung lượng lưu trữ mà còn giảm thời gian tải qua mạng và thời gian tải xuống từ internet một cách đáng kể. Với khả năng nén tệp, mã chương trình, dữ liệu và tài nguyên, ASPack cung cấp một giải pháp toàn diện, phù hợp với các nhà phát triển cần giảm kích thước tệp và cải thiện hiệu suất hệ thống.

Ngoài khả năng nén mạnh mẽ, ASPack còn cung cấp lớp bảo mật bổ sung, bảo vệ chương trình và ứng dụng khỏi các phân tích trái phép, các công cụ gỡ lỗi và dịch ngược. Các chương trình được nén bằng ASPack vẫn duy trì đầy đủ chức năng và hiệu suất, hoạt động như trước mà không gây bất kỳ ảnh hưởng tiêu cực nào đến trải nghiệm người dùng. Công cụ này hoạt động độc lập và không yêu cầu phần mềm bổ sung, đồng thời hỗ trợ các tên tệp dài, mang lại sự linh hoạt cho người dùng trong các dự án với yêu cầu đặc biệt.

**Ưu điểm nổi bật của ASPack:**

* Giảm kích thước tệp thực thi lên đến 70%, giúp tối ưu hóa dung lượng lưu trữ.
* Giảm thời gian tải qua mạng và thời gian tải xuống từ internet, nâng cao hiệu suất.
* Các ứng dụng Windows nhúng yêu cầu ít không gian lưu trữ hơn đáng kể.
* Bảo vệ tài nguyên và mã nguồn khỏi việc xem trộm, công cụ phân tích mã (disassemblers) và dịch ngược (decompilers).
* Không yêu cầu phí bản quyền khi phân phối các chương trình đã nén.
* Tương thích với các tệp thực thi được tạo ra từ Microsoft Visual C++, Visual Basic, Embarcadero Delphi, C++ Builder và các trình biên dịch Win32 khác.

**Một số nhược điểm của ASPack:**

* Giới hạn đối với mã máy: ASPack chủ yếu tối ưu cho các tệp thực thi Win32 và có thể không tương thích hoàn toàn với các mã máy phức tạp hoặc các ứng dụng sử dụng cấu trúc tệp khác.
* Chỉ hỗ trợ các phiên bản Windows cũ: Mặc dù công cụ này hoạt động tốt trên các phiên bản Windows phổ biến như Windows XP, Vista, và Windows 7, nhưng nó có thể gặp vấn đề tương thích với các phiên bản Windows mới hơn như Windows 10 và Windows 11.
* Khả năng tương thích với các công cụ phát triển khác: Dù hỗ trợ tốt các công cụ phát triển phổ biến như Microsoft Visual C++ và Visual Basic, ASPack có thể gặp khó khăn khi tương thích với một số trình biên dịch và các nền tảng phát triển mới.
* Hiệu suất giảm khi nén một số tệp lớn: Trong một số trường hợp, việc nén các tệp EXE lớn có thể gây giảm hiệu suất, đặc biệt khi giải nén chương trình, mặc dù thuật toán giải nén của ASPack được tối ưu hóa, nhưng quá trình này vẫn có thể tốn thời gian trong một số tình huống đặc biệt.

ASPack thích hợp nhất cho các ứng dụng Windows nhẹ, cần tối ưu hóa dung lượng và bảo mật mã nguồn, đồng thời hỗ trợ tốt các công cụ phát triển phần mềm truyền thống. Tuy ASPack là một công cụ mạnh mẽ và hiệu quả, nhưng nó cũng có những nhược điểm cần được cân nhắc khi quyết định sử dụng cho các dự án phát triển phần mềm hiện đại.

* 1. **MPRESS**

A close up of a logo

Description automatically generated

Hình . Công cụ nén tệp tin MPRESS

MPRESS là một công cụ miễn phí dành cho việc nén các tệp thực thi, hỗ trợ nhiều định dạng như PE32, PE32+, .NET, và MAC-DARWIN. Với khả năng giảm kích thước chương trình và thư viện, MPRESS giúp tối ưu hóa dung lượng tệp, đồng thời cải thiện thời gian khởi động khi ứng dụng được tải từ các thiết bị lưu trữ chậm hoặc qua mạng. Đây là giải pháp lý tưởng để cải thiện hiệu quả hoạt động của các ứng dụng.

Một trong những tính năng nổi bật của MPRESS là công nghệ giải nén tại chỗ (in-place decompression), cho phép giải nén các tệp thực thi mà không gây áp lực lên bộ nhớ hoặc ảnh hưởng tiêu cực đến hiệu suất. Ngoài ra, MPRESS còn cung cấp lớp bảo vệ chống lại các hành vi dịch ngược và phân tích mã từ các hacker không chuyên, đảm bảo tính bảo mật cho mã nguồn của bạn. Đặc biệt, các chương trình được nén bằng MPRESS hoạt động mượt mà như trước đây, không gặp bất kỳ suy giảm hiệu suất nào.

MPRESS sử dụng thư viện nén dữ liệu LZMAT với tốc độ cực kỳ nhanh. Kể từ phiên bản 2.00, phát hành ngày 21 tháng 3 năm 2009, MPRESS còn hỗ trợ nén tùy chọn bằng thuật toán LZMA, mang lại hiệu quả nén vượt trội hơn. Đây là công cụ hoàn toàn miễn phí, không yêu cầu bất kỳ chi phí bản quyền nào, phù hợp với mọi đối tượng sử dụng.

Vào tháng 3 năm 2011, ông DonDD đã phát triển và phát hành giao diện đồ họa (GUI) cho MPRESS, giúp người dùng dễ dàng thao tác hơn

**Ưu điểm nổi bật của MPRESS:**

* Nén nâng cao cho nhiều loại tệp thực thi: Hỗ trợ nén các tệp .NET (anyCPU, x86, AMD64, IA64 EXE) với khả năng tương thích Microsoft Framework 1.1/2.0/3.0/4.0 mà không yêu cầu cài đặt .NET Framework. Đồng thời, cung cấp tính năng nén tối ưu cho các tệp PE32/PE32+ (EXE, DLL, OCX, v.v.) và ứng dụng macOS như mac-darwin-i386, mac-darwin-x86\_64, mac-darwin-ub.
* Công nghệ nén hiệu quả và bảo mật: Sử dụng thuật toán LZMA tùy chọn, nén mã chương trình, dữ liệu và tài nguyên một cách tối ưu. MPRESS còn giúp loại bỏ thông tin nhạy cảm như relocation, debug information và exceptions, bảo vệ ứng dụng khỏi các rủi ro bảo mật.
* Giải nén nhanh và hiệu quả: Với tốc độ giải nén lên đến ~210 MB/giây trên AMD 2500+, MPRESS sử dụng công nghệ giải nén tại chỗ (in-place decompression), không tạo thêm áp lực bộ nhớ hay gây suy giảm hiệu suất.
* Hoạt động linh hoạt và toàn diện: Hỗ trợ UNICODE, hoạt động hoàn toàn minh bạch và tự chứa. Giao diện dòng lệnh cho phép tích hợp dễ dàng vào các tập lệnh batch hoặc tệp makefile để tự động hóa quá trình nén.

**Một số nhược điểm của MPRESS:**

* Tương thích hệ điều hành hạn chế: Mặc dù hỗ trợ nhiều phiên bản Windows từ 9x đến Windows 8, MPRESS không được cập nhật để tương thích tốt với các hệ điều hành mới hơn như Windows 10, Windows 11 hoặc các môi trường hiện đại hơn.
* Chỉ bảo vệ ở mức cơ bản: MPRESS cung cấp một số tính năng bảo vệ chống dịch ngược (reverse engineering), nhưng không hiệu quả trước các hacker chuyên nghiệp hoặc các công cụ mạnh hơn dành cho việc phân tích mã.
* Không có bản cập nhật thường xuyên: Phiên bản mới nhất (2.19) đã không được cập nhật trong nhiều năm, dẫn đến việc thiếu hỗ trợ cho các công nghệ hoặc định dạng mới.

MPRESS phù hợp nhất để nén các tệp thực thi và thư viện nhỏ, giảm dung lượng và tăng tốc khởi động trên các hệ thống cũ hoặc tài nguyên hạn chế. Nó cũng là lựa chọn tốt để bảo vệ cơ bản chống dịch ngược mà không phát sinh chi phí.

* 1. **PECompact**

A close up of a text

Description automatically generated

Hình . Công cụ nén tệp tin PECompact

PECompact là một công cụ nén tệp thực thi trên Windows. Bạn sẽ không tìm thấy bất kỳ trình nén nào khác có các tính năng giống như PECompact. Công cụ này có một hệ thống plugin cung cấp khả năng tùy chỉnh gần như không giới hạn, làm cho nó khác biệt hoàn toàn so với các đối thủ cạnh tranh khác.

Theo thông tin của website Bitsum, PECompact nén tốt hơn ASPack, tốt hơn nhiều so với PEtite, và nén ngang ngửa với UPX (nhưng không có chế độ giải nén như UPX). PECompact giữ mã và dữ liệu bị nén ở mức tốt nhất có thể, cung cấp sự cân bằng hoàn hảo giữa bảo mật, độ ổn định, và khả năng tương thích. Các tệp thực thi được nén bằng PECompact có thể hoạt động ở mọi nơi mà không cần bất kỳ thay đổi nào. PECompact còn tương thích với tính năng DEP. *DEP là gì*? DEP – Data Execution Prevention là một tính năng bảo mật trong hệ điều hành nhằm ngăn chặn các mã độc hại thực thi từ các vùng bộ nhớ không được đánh dấu là có thể thực thi (executable).

**Ưu điểm nổi bật của PECompact:**

* Khả năng nén vượt trội: So với các trình nén thực thi khác, tỷ lệ nén của PECompact hoặc là vượt trội, hoặc ít nhất là tương đương, tùy thuộc vào EXE và các yếu tố khác.
* Mở rộng linh hoạt: Cung cấp danh sách phong phú các plug-in để sử dụng, hỗ trợ tạo plug-in tùy chỉnh hoặc bộ nạp thay thế bằng Loader SDK (cần mua riêng).
* Hỗ trợ DEP đầy đủ: Hỗ trợ tính năng DEP (Data Execution Prevention) mặc định, với khả năng nâng cao là ngăn chặn việc thực thi trên các trang không được đánh dấu là có thể thực thi.
* Thực hiện đúng cách (Does things ‘right’): PECompact thực hiện mọi việc một cách đúng đắn, như bỏ qua việc nén các vùng chia sẻ của DLL, bảo toàn thông tin gỡ lỗi.

**Một số nhược điểm của PECompact:**

* Không hỗ trợ driver thiết bị (device drivers) và một số assembly .NET, hạn chế phạm vi sử dụng trong các ứng dụng chuyên biệt.
* Chưa hỗ trợ PE+ (PE64), tức là các tệp thực thi 64-bit, vì điều này yêu cầu viết lại hoàn toàn bộ nạp (loader).
* Quy trình nén tệp và việc ký mã sau khi nén (code signing) có thể làm tăng thêm bước xử lý và yêu cầu quản lý cẩn thận.
* Một số tính năng bổ sung, như Loader SDK để tạo bộ nạp tùy chỉnh, yêu cầu mua riêng, làm tăng chi phí tổng thể.
* Mặc dù cung cấp nhiều plug-in, việc tích hợp và tối ưu hóa các plug-in này có thể yêu cầu kiến thức kỹ thuật sâu.
* Hơi khó ứng dụng trong thực tiễn vì không có khả năng giải nén.

1. **FSG**

A close up of a logo

Description automatically generated

Hình . Công cụ nén tập tin FSG

FSG (Fast Small Good) là một công cụ nén tệp thực thi nhỏ gọn và hiệu quả, được thiết kế để tối ưu hóa kích thước các tệp PE (Portable Executable) trên hệ điều hành Windows, được phát triển bởi Bartosz Gorski và xuất hiện lần đầu vào khoảng năm 2002. Với mục tiêu giảm thiểu kích thước tệp mà vẫn duy trì hiệu suất runtime, FSG đã được sử dụng rộng rãi trong các ứng dụng yêu cầu phân phối gọn nhẹ và nhanh chóng. Các tệp thực thi nén bằng FSG thường có tốc độ giải nén nhanh và không gây ảnh hưởng đáng kể đến hiệu suất runtime của ứng dụng. Với thiết kế tối giản, FSG dễ sử dụng và không yêu cầu cấu hình phức tạp, phù hợp với cả người dùng cơ bản và chuyên gia.Công cụ này tương thích tốt với nhiều phiên bản hệ điều hành Windows, hỗ trợ các tệp thực thi được tạo từ các ngôn ngữ lập trình phổ biến như C++, Delphi, và VB.

**Ưu điểm nổi bật của FSG:**

* Dung lượng nhỏ: FSG không chỉ nén các tệp thực thi, mà chính công cụ này cũng có kích thước rất nhỏ, dễ dàng tích hợp vào các quy trình phát triển phần mềm.
* Giải nén nhanh: Tốc độ giải nén cao giúp tối ưu hóa thời gian khởi chạy chương trình, đặc biệt phù hợp với các ứng dụng yêu cầu tốc độ.
* Không yêu cầu phần mềm bổ sung: Các tệp nén bởi FSG vẫn hoàn toàn tự chứa, không cần cài đặt thêm công cụ giải nén để hoạt động.
* Hiệu suất nén tốt: Mặc dù không phải là công cụ nén có tỷ lệ cao nhất, FSG vẫn đạt hiệu suất nén đáng kể, đặc biệt trên các tệp PE nhỏ hoặc vừa.
* Đơn giản hóa việc phân phối: Nhờ khả năng giảm kích thước, FSG giúp tiết kiệm chi phí lưu trữ và tối ưu hóa việc phân phối phần mềm qua mạng.

**Một số nhược điểm của FSG:**

* Giới hạn định dạng: FSG chủ yếu hỗ trợ các tệp PE trên Windows, không thể nén các định dạng hoặc kiến trúc khác như 64-bit (PE+).
* Bị nhận diện nhầm: Do từng bị sử dụng bởi một số phần mềm độc hại, các tệp nén bởi FSG đôi khi bị phần mềm diệt virus đánh dấu nhầm là nguy hiểm.
* Tính năng hạn chế: So với các công cụ nén tiên tiến như UPX, FSG thiếu các tùy chọn mở rộng hoặc hỗ trợ plug-in.
* Tương thích không hoàn hảo: Một số ứng dụng sử dụng tính năng kiểm tra toàn vẹn (integrity check) có thể không hoạt động đúng khi được nén bởi FSG.

FSG là một công cụ nén tệp thực thi đơn giản, hiệu quả, và dễ sử dụng, phù hợp cho các ứng dụng nhỏ hoặc vừa cần giảm kích thước để tối ưu hóa việc phân phối. Mặc dù có một số hạn chế so với các công cụ nén hiện đại, FSG vẫn là một lựa chọn đáng cân nhắc trong các trường hợp cần giải pháp nén nhẹ và nhanh chóng.

1. **MEW**



Hình . Công cụ nén tệp tin MEW

MEW (Minimal Executable Wrapper) là một công cụ nén tệp thực thi được thiết kế đặc biệt để xử lý các tệp nhỏ, nhưng cũng có thể nén hiệu quả các tệp lớn hơn. MEW giúp giảm kích thước tệp thực thi, tối ưu hóa dung lượng lưu trữ và giảm thời gian tải xuống, mang lại lợi ích rõ ràng trong việc phân phối và lưu trữ tệp. Một trong những điểm mạnh của MEW là khả năng nén tệp EXE mà không làm mất tính tự chứa của chương trình, nghĩa là tệp nén vẫn có thể chạy mà không cần phần mềm bổ sung. MEW cũng cung cấp nhiều tùy chọn nén cho người dùng, bao gồm nén tài nguyên, xóa tài nguyên không cần thiết và hỗ trợ thuật toán LZMA để đạt được tỷ lệ nén tốt hơn.

MEW hỗ trợ các hệ điều hành Windows từ phiên bản Windows 95 đến Windows 7, mặc dù không hỗ trợ các hệ điều hành 64-bit, nhưng có thể hoạt động tốt trên các máy tính 32-bit. Công cụ này có thể nén các tệp EXE được viết bằng các ngôn ngữ lập trình phổ biến như C++ và Delphi.

**Các ưu điểm nổi bật của MEW:**

* Tỷ lệ nén tốt: MEW giúp giảm kích thước tệp EXE, tối ưu hóa dung lượng lưu trữ và thời gian tải xuống.
* Tốc độ nén nhanh: MEW cung cấp khả năng nén nhanh chóng, làm cho quá trình nén trở nên hiệu quả hơn.
* Tính linh hoạt: MEW cung cấp nhiều tùy chọn nén và hỗ trợ thuật toán LZMA, giúp người dùng tối ưu hóa tỷ lệ nén.
* Đảm bảo tính tương thích: Tệp EXE nén bởi MEW vẫn có thể chạy bình thường mà không gặp phải vấn đề tương thích với hệ điều hành Windows.
* Mã nguồn mở: MEW là phần mềm mã nguồn mở, cho phép người dùng tùy chỉnh và sử dụng miễn phí.

**Một số nhược điểm của MEW:**

* Giao diện người dùng chưa thân thiện: Giao diện của MEW không được tối ưu và thiếu các tài liệu hướng dẫn, điều này có thể gây khó khăn cho người dùng mới.
* Không hỗ trợ hệ điều hành 64-bit: MEW không hỗ trợ các hệ điều hành 64-bit, điều này khiến nó không phù hợp với các nền tảng hiện đại.
* Không có tệp trợ giúp: Ứng dụng không đi kèm với tài liệu hướng dẫn, người dùng phải tìm kiếm trực tuyến để hiểu cách sử dụng.

Với những tính năng nổi bật và khả năng nén hiệu quả, MEW là một công cụ hữu ích cho việc nén tệp EXE. Tuy nhiên, giao diện người dùng và sự hỗ trợ cho hệ điều hành 64-bit vẫn là những vấn đề cần được cải thiện để đạt được sự hoàn thiện hơn.

1. **PETITE**

A close-up of a logo

Description automatically generated

Hình . Công cụ nén tệp tin PETITE

PETITE là một công cụ nén tệp thực thi miễn phí dành cho hệ điều hành Win32, bao gồm các phiên bản Windows 95/98/2000/NT/XP/Vista/7 và các hệ điều hành tương tự. Petite giúp giảm kích thước tệp EXE, DLL và các tệp thực thi khác mà không ảnh hưởng đến khả năng thực thi của chúng, tệp nén sẽ tự giải nén khi chạy và hoạt động như các phiên bản không nén gốc. Một trong những điểm nổi bật của Petite là tính năng tự kiểm tra virus, giúp bảo vệ các tệp thực thi khỏi sự nhiễm độc khi được sử dụng.

Petite cung cấp hai phiên bản: phiên bản giao diện người dùng (GUI) và phiên bản dòng lệnh. Phiên bản GUI giúp người dùng dễ dàng nén tệp, trong khi phiên bản dòng lệnh phục vụ cho các nhà phát triển, giúp họ tích hợp Petite vào quy trình biên dịch của các trình biên dịch và makefile. Petite cũng có thể được sử dụng như một tiện ích mở rộng trong Windows Explorer, cho phép người dùng nén tệp chỉ bằng cách nhấp chuột phải vào tệp cần nén.

**Ưu điểm nổi bật của Petite:**

* Tỷ lệ nén tốt: Petite giúp giảm kích thước tệp thực thi một cách hiệu quả, tối ưu hóa không gian lưu trữ và giảm thời gian tải tệp.
* Tính tự giải nén: Các tệp thực thi nén sẽ tự giải nén khi chạy, giữ nguyên khả năng hoạt động như tệp không nén.
* Tính năng kiểm tra virus: Petite tích hợp khả năng tự kiểm tra virus trong mỗi tệp nén, đảm bảo bảo mật khi thực thi.
* Tính linh hoạt: Phiên bản giao diện người dùng và dòng lệnh của Petite đáp ứng nhu cầu của cả người dùng thông thường và nhà phát triển phần mềm.
* Tính tiện dụng: Petite có thể được sử dụng như một tiện ích mở rộng trong Windows Explorer, giúp người dùng nén tệp dễ dàng mà không cần mở ứng dụng.

**Một số nhược điểm của Petite:**

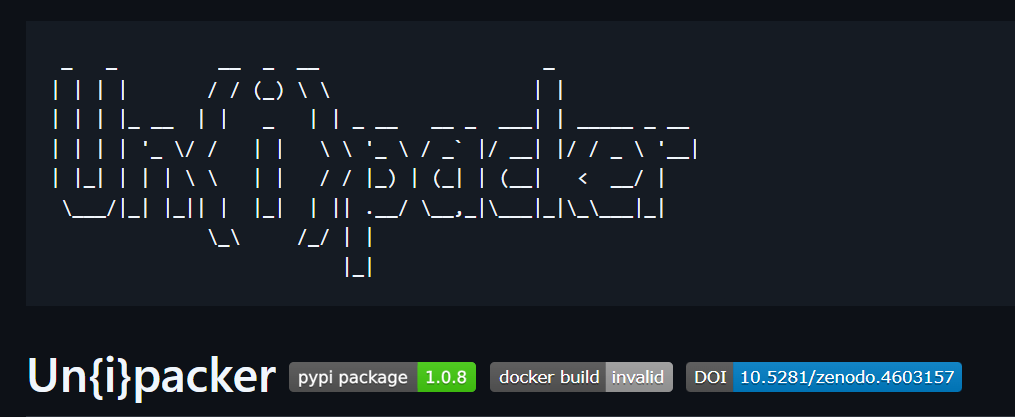
* Giao diện người dùng chưa tối ưu: Mặc dù phiên bản GUI của Petite giúp nén tệp dễ dàng, nhưng giao diện vẫn còn đơn giản và không phải là thân thiện nhất.
* Không hỗ trợ hệ điều hành 64-bit: Petite không hỗ trợ các hệ điều hành 64-bit, điều này khiến nó không phù hợp với nền tảng hiện đại.
* Tốc độ nén không phải lúc nào cũng nhanh: Mặc dù Petite nén hiệu quả, nhưng so với một số công cụ nén khác, tốc độ nén có thể chậm hơn đối với các tệp lớn.

Với khả năng nén hiệu quả và tính năng kiểm tra virus tích hợp, Petite là một công cụ hữu ích cho việc nén các tệp thực thi trên hệ điều hành Win32. Tuy nhiên, giao diện người dùng và sự hỗ trợ cho các hệ điều hành 64-bit cần được cải thiện để đáp ứng tốt hơn nhu cầu của người dùng hiện đại.

## Các công cụ phát hiện và giải nén các tập tin thực thi Windows

### Unipacker

Unipacker là một công cụ phát hiện và giải nén tệp thực thi Windows bị nén một cách mạnh mẽ, chủ yếu được sử dụng trong việc phân tích bảo mật, đặc biệt khi cần truy xuất và giải nén các tệp thực thi đã được nén bằng các công cụ packer như UPX, FSG hoặc Petite. Công cụ này giúp phát hiện và giải nén tệp mà không làm mất dữ liệu hoặc ảnh hưởng đến chức năng của chúng. Unipacker đặc biệt hữu ích trong việc kiểm tra mã độc hoặc khi cần tiếp cận trực tiếp nội dung bên trong một tệp thực thi nén mà không cần phải chạy nó. Nó hỗ trợ nhiều định dạng tệp nén và cung cấp các tùy chọn mạnh mẽ để xử lý tệp thực thi một cách hiệu quả.



Hình . Công cụ Unipacker

Lưu ý: Công cụ này được triển khai trên Linux và chỉ hỗ trợ các tệp thực thi định dạng PE32 (Portable Executable) của Windows và không hỗ trợ các tệp ELF (Executable and Linkable Format).

**Các packer được hỗ trợ bởi Unipacker**

Unipacker có khả năng hỗ trợ đa dạng các trình nén tệp tin hiện tại trên thị trường:

* **ASPack**: Trình packer thương mại cao cấp với tỷ lệ nén cao.
* **FSG**: Phần mềm miễn phí, giải nén nhanh chóng.
* **MEW**: Thiết kế đặc biệt cho các tệp nhị phân nhỏ.
* **MPRESS**: Miễn phí, trình packer phức tạp hơn.
* **PEtite**: Trình packer miễn phí, tương tự như ASPack.
* **UPX**: Trình packer mã nguồn mở, hỗ trợ đa nền tảng.
* **YZPack**: (Không có mô tả cụ thể).

Các packer khác cũng có thể hoạt động nếu Unipacker đã hỗ trợ các hàm API cần thiết. Đối với những packer không được nhận diện, bạn sẽ cần chỉ định thủ công địa chỉ bắt đầu và kết thúc để mô phỏng.

**Cách phát hiện các trình nén tệp tin**

Các dấu hiệu nhận dạng sẽ được tìm kiếm theo các quy tắc YARA (được lưu trữ trong file packer\_signatures.yar) và chọn ra packer thích hợp.

A screen shot of a computer code

Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi UPX

*A computer screen shot of a computer code

Description automatically generated*

Hình . Dấu hiệu nhận biết file bị nén bởi FSG

A screen shot of a computer code

Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi ASPack

A computer screen shot of a program code

Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi PECompact

A computer code with text

Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi PETITE

A computer screen shot of a computer code

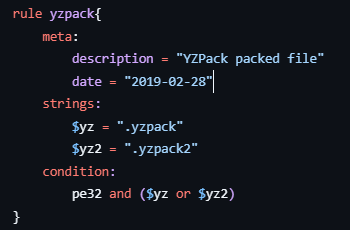
Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi MEW

A screen shot of a computer program

Description automatically generated

Hình . Dấu hiệu nhận biết file bị nén bởi MPress



Hình . Dấu hiệu nhận biết file bị nén bởi YZPack

**Quá trình giải nén các tệp tin**

Phương thức để giải nén tệp tin sẽ được lưu trữ trong file “unpack.py”, luồng thực thi quá trình giải nén sẽ diễn ra như sau:

* 1. **Class DefaultUnpacker**

Lớp này chịu trách nhiệm chính trong việc giải nén tệp. Nó kiểm tra các phần (section) của tệp, quản lý bộ nhớ ảo, và trích xuất thông tin cần thiết.

* **Phương thức khởi tạo (\_\_init\_\_)**:

Thiết lập các tham số như:

* Entry point (điểm bắt đầu thực thi).
* Các section được chấp nhận.
* Địa chỉ bộ nhớ được phép.
* **Chức năng chính**:

Chức năng chính của lớp DefaultUnpacker là xác định dải địa chỉ hợp lệ bằng cách sử dụng phương thức “get\_allowed\_addr\_ranges”, giúp tạo ra danh sách các vùng nhớ được chấp nhận để thực thi. Đồng thời, nó kiểm tra xem một địa chỉ cụ thể có nằm trong dải địa chỉ hợp lệ hay không thông qua phương thức “is\_allowed”. Sau khi hoàn tất quá trình giải nén, lớp này sử dụng phương thức dump để ghi nội dung bộ nhớ đã giải nén ra tệp kết quả, thường là unpacked.exe.

* 1. **Class AutomaticDefaultUnpacker**

Đây là lớp con của DefaultUnpacker, giúp tự động hóa việc chọn điểm bắt đầu và kết thúc mà không cần người dùng nhập thủ công.

* 1. **Class UPXUnpacker (và các lớp tương tự cho các packer khác)**

Lớp này được thiết kế riêng để giải nén các tệp được nén bằng UPX (hoặc các trình giải nén). Nó kế thừa từ AutomaticDefaultUnpacker, nhưng có một số điều chỉnh cụ thể để phù hợp với từng loại packer.

Lưu ý: Chỉ xử lý các section có dữ liệu thực (SizeOfRawData > 0).

* 1. **Hàm identifypacker(sample, yar)**

Đoạn mã này dùng để xác định loại packer được sử dụng cho tệp nhị phân đã cung cấp. Nó sử dụng các YARA rules để khớp mẫu với tệp và xác định các trình nén tệp tin. Nếu không thể xác định được loại packer, hàm sẽ trả về “unknown”.

* 1. **Hàm generate\_label(match)**

Hàm “generate\_label(match)” có nhiệm vụ phân loại loại packer dựa trên kết quả khớp (matches) từ YARA. Liên kết với hàm trên, “identifypacker(sample, yar)” là hàm bao quát hơn, chịu trách nhiệm chạy các quy tắc YARA trên tệp nhị phân (sample) để tìm kết quả khớp, sau đó gọi generate\_label(match) để xác định loại packer và trả về kết quả cuối cùng.

* 1. **Hàm get\_unpacker(sample, auto\_default\_unpacker=True)**

Cuối cùng, hàm này được thiết kế để lựa chọn và tạo ra một một trình giải nén phù hợp cho tệp nhị phân dựa trên các trình nén tệp tin đã xác định.

A screen shot of a computer program

Description automatically generated

Hình 19. Trình giải nén công cụ UPX

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ PETITE

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ FSG

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ ASPack

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ YZPack

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ MEW

A computer screen shot of text

Description automatically generated

Hình . Trình giải nén công cụ MPress

A screen shot of a computer program

Description automatically generated

Hình . Trình giải nén công cụ PECompact

Trong đó, các class như “ImportRebuilderDump” hoặc các class như “MEWDump” là để hỗ trợ quá trình sửa chữa và phục hồi các file thực thi đã được giải nén, tập trung vào hai nhiệm vụ chính: 1. Sửa chữa bảng import, đảm bảo các tham chiếu đến DLL và hàm được khôi phục hoặc xây dựng lại chính xác, giúp file hoạt động bình thường; 2. Điều chỉnh quyền bộ nhớ của các section, thiết lập quyền đọc, ghi, và thực thi cho các section để đảm bảo tính toàn vẹn và khả năng chạy của file trong môi trường thực tế.

### ClamAV

ClamAV là một công cụ diệt virus mã nguồn mở được phát triển dưới giấy phép GPLv2 và được thiết kế đặc biệt để quét email trên các cổng thư điện tử. Với khả năng phát hiện các loại trojan, virus, phần mềm độc hại và các mối đe dọa khác, ClamAV trở thành một lựa chọn lý tưởng để bảo mật hệ thống. Công cụ này cung cấp một loạt các tiện ích, bao gồm một dịch vụ quét đa luồng, quét dòng lệnh và các công cụ nâng cao để tự động cập nhật cơ sở dữ liệu.

Và trong dữ cảnh sử dụng cho các trình đóng gói tệp tin, ClamAV cũng hoàn thành rất xuất sắc trong việc phát hiện và giải nén các tệp tin bị đóng gói.

A red devil face with horns and text

Description automatically generated

Hình . Công cụ ClamAV

Một trong những ưu điểm lớn của ClamAV là hiệu suất cao và tính linh hoạt. Với dịch vụ quét đa luồng, ClamAV có thể xử lý nhiều tác vụ quét đồng thời, giúp tăng tốc quá trình bảo vệ hệ thống mà không làm giảm hiệu suất. Ngoài ra, công cụ này hỗ trợ nhiều định dạng tệp và ngôn ngữ chữ ký, cũng như khả năng giải nén tệp và lưu trữ, cho phép nó phát hiện mối đe dọa trong nhiều loại tệp khác nhau.

ClamAV cũng nổi bật với tính open source và dễ dàng tùy chỉnh. Là phần mềm mã nguồn mở, người dùng có thể truy cập và tùy biến các phiên bản của ClamAV phù hợp với hệ điều hành của mình, giúp linh hoạt trong việc triển khai và duy trì phần mềm bảo mật cho hệ thống. Cùng với khả năng cập nhật cơ sở dữ liệu tự động, ClamAV giúp người dùng luôn được bảo vệ trước những mối đe dọa mới nhất.

**Các packer được hỗ trợ bởi ClamAV**

Tương tự như Unipacker, ClamAV cũng hỗ trợ đa dạng các trình nén tệp tin hiện tại trên thị trường. Tuy nhiên, có điểm nổi bật hơn là ClamAV hỗ trợ luôn cho các tệp tin PE 64 bits. Ngoài các công cụ đã giới thiệu ở trên như UPX, ASPack, FSG, Petite, MEW, ClamAV còn hỗ trợ thêm:

* PeSpin: Công cụ nén và bảo vệ tệp PE, tạo sự khó khăn cho việc phân tích.
* NsPack: Giảm dung lượng mà không ảnh hưởng đến khả năng thực thi.
* wwpack32: Công cụ nén PE mạnh mẽ, giữ nguyên tính tương thích và hiệu suất của tệp.

**Cách phát hiện tệp tin bị nén và cách giải nén**

ClamAV sẽ tổng hợp các signature và các dấu hiệu cho thấy một tệp tin bị đóng gói vào các tệp tin khác nhau tương ứng với từng trình đóng gói. Dưới đây là một ví dụ về công cụ UPX.

Để tìm thấy các tệp tin liên quan đến công cụ, chúng ta có thể đến địa chỉ “libclamav”.

A screenshot of a computer

Description automatically generated

Hình . Các file chứa đoạn code liên quan đến UPX

Trong file “upx.c” sẽ một số dấu hiệu nhận biết tệp tin bị nén bởi UPX như sau:

* Phát hiện "magic bytes" của UPX: Đoạn mã kiểm tra các đặc trưng của file UPX bằng cách tìm kiếm các byte đặc biệt trong vùng dữ liệu của file. Cụ thể, mã tìm kiếm các byte \x8d\xbe trong đoạn mã máy của file, đây là một chỉ dấu phổ biến trong các file nén UPX.
* Cấu trúc của PE header: Đoạn mã xác minh header của file PE. Trong khi kiểm tra các thành phần của PE header, mã tìm kiếm một số thông tin đặc trưng cho file nén UPX, như là các chỉ số trong bảng mục nhập, các đoạn mã cụ thể như “lea edi, ...” và “mov eax, [edi]”, những đặc điểm này có thể là dấu hiệu của file nén UPX.
* Kiểm tra các chỉ số và section trong PE header: Mã kiểm tra các section của file PE và các thông số như realstuffsz, valign, và sectcnt để đảm bảo rằng cấu trúc của file tuân theo chuẩn của UPX.

Một hàm cho việc tìm kiếm dấu hiệu của UPX:

A screenshot of a computer program

Description automatically generated

Hình . Đoạn code nhận biết UPX

Trong file này cũng chứa cả cách để giải nén cho công cụ này.

A computer screen with white text

Description automatically generated

Hình . Đoạn code giải nén upx\_inflate2b

A computer screen with white text and blue text

Description automatically generated

Hình . Đoạn code giải nén upx\_inflate2d

Hàm upx\_inflate2b và upx\_inflate2d: Đây là các hàm chịu trách nhiệm giải nén dữ liệu từ định dạng nén của UPX. Chúng thực hiện giải mã các phần của file đã được nén và phục hồi lại dữ liệu gốc.

* upx\_inflate2b: Hàm này có thể liên quan đến việc giải nén dữ liệu nén theo thuật toán đặc biệt của UPX, có thể sử dụng các phương pháp như thuật toán nén LZMA, zlib, hoặc các cơ chế nén khác mà UPX áp dụng.
* upx\_inflate2d: Đây là hàm bổ sung, có thể là một phần của quy trình giải nén, giúp xử lý phần dữ liệu đã được giải nén một cách hoàn chỉnh hơn.

Bên cạnh đó còn có hàm decompress hoặc các hàm phụ trợ cung cấp một cách giải nén chung, thông qua các chỉ dẫn trong header PE và các byte chỉ định vị trí. Sau khi nhận diện file nén là UPX, hệ thống sẽ tiếp tục vào các bước giải nén thực sự.

Tương tự cho các trình đóng gói tệp tin khác.

A black and white text

Description automatically generated

Hình . Các file chứa đoạn code liên quan đến FSG

A black background with white text

Description automatically generated

Hình . Các file chứa đoạn code liên quan đến ASPack

A black and white text

Description automatically generated

Hình . Các file chứa đoạn code liên quan đến PETITE

A black background with white text

Description automatically generated

Hình . Các file chứa đoạn code liên quan đến MEW

### Detect It Easy

Detect It Easy (DiE) là một công cụ mạnh mẽ dùng để nhận diện loại tệp tin, phổ biến trong giới phân tích mã độc, chuyên gia an ninh mạng và dịch ngược trên toàn thế giới. DiE hỗ trợ cả phân tích dựa trên signature và phân tích heuristic, giúp kiểm tra tệp hiệu quả trên nhiều nền tảng, bao gồm Windows, Linux và MacOS. Kiến trúc phát hiện linh hoạt khiến DiE trở thành một trong những công cụ đa năng nhất trong lĩnh vực này.

A screenshot of a computer

Description automatically generated

Hình . Công cụ Detect It Easy

**Tại sao nên sử dụng Detect It Easy?**

Hệ thống signature linh hoạt và khả năng lập trình của DiE khiến nó trở thành công cụ thiết yếu trong phân tích mã độc và pháp chứng số. Với các công cụ phân tích tĩnh truyền thống thường bị giới hạn về phạm vi và dễ gặp phải cảnh báo sai, thiết kế có thể tùy chỉnh của DiE cho phép tích hợp logic phát hiện mới một cách chính xác, đảm bảo kết quả đáng tin cậy trên nhiều loại tệp khác nhau.

Tuy nhiên, đối với các trình đóng gói tệp tin, DiE chỉ cung cấp tính năng phát hiện chứ không có tính năng giải nén như hai công cụ được giới thiệu trước đó. Đây cũng là một nhược điểm rất lớn trong quá trình phục hồi tệp tin bị nén.

Detect It Easy cung cấp ba phiên bản khác nhau để người dùng có thể lựa chọn tùy theo nhu cầu sử dụng:

1. **die (Graphical Interface)**: Là phiên bản với giao diện đồ họa, giúp người dùng dễ dàng thao tác và trực quan hơn trong việc kiểm tra và phân tích các tệp. Phiên bản này thích hợp cho những ai muốn sử dụng công cụ một cách đơn giản và không yêu cầu quá nhiều thao tác dòng lệnh.
2. **diec (Command-line version for batch processing)**: Phiên bản dòng lệnh này được thiết kế dành cho những người làm việc với khối lượng tệp lớn hoặc cần tự động hóa quy trình phân tích tệp. "diec" hỗ trợ xử lý theo lô (batch processing), giúp tiết kiệm thời gian và tăng hiệu quả khi xử lý nhiều tệp cùng lúc.
3. **diel (Lightweight GUI version)**: Là một phiên bản nhẹ với giao diện đồ họa, được tối ưu cho các máy tính cấu hình thấp hoặc khi người dùng cần một công cụ đơn giản nhưng vẫn giữ được các tính năng quan trọng của DiE. Phiên bản này là lựa chọn tuyệt vời cho những ai cần một công cụ nhanh chóng và dễ sử dụng mà không chiếm quá nhiều tài nguyên hệ thống.

Mỗi phiên bản đều được thiết kế để phục vụ các mục đích khác nhau, từ người dùng thông thường cho đến những chuyên gia cần xử lý lượng dữ liệu lớn một cách hiệu quả.

**Các packer được hỗ trợ bởi DiE**

Detect It Easy (DiE) có khả năng phát hiện nhiều loại packer phổ biến như UPX, ASPack, FSG, PECompact, Mpress, MEW … Ngoài ra còn có:

* Themida: một packer chuyên dụng để bảo vệ phần mềm khỏi việc phân tích
* kkrunchy: một packer nhẹ, hiệu quả được thiết kế để nén các tệp EXE.
* NSIS (Nullsoft Scriptable Install System): Mặc dù NSIS chủ yếu được sử dụng để tạo các trình cài đặt, nhưng nó cũng có thể nén tệp trong quá trình tạo trình cài đặt.

DiE cũng có thể phát hiện các packer tùy chỉnh hoặc các packer không phổ biến, nếu chúng tuân thủ các nguyên lý nén hoặc mã hóa được hỗ trợ.

**Cách phát hiện các trình đóng gói**

Đối với các trình đóng gói, DiE sẽ viết các signature và các dấu hiệu nhận biết vào các file riêng biệt tương ứng với từng loại trình đóng gói. Các file này cũng được cập nhật liên tục để có thể thích nghi với việc các trình đóng gói tệp tin ngày càng nhiều và vô cùng tinh vi. Dưới đây là một số file dùng để nhận biết:

A screenshot of a computer program

Description automatically generated

Hình . Các file trong DiE liên quan đến UPX



Hình . Các file trong DiE liên quan đến ASPack



Hình . Các file trong DiE liên quan đến Mpress

Một ví dụ về cách DiE phát hiện tệp tin bị nén bới công cụ FSG.

A screenshot of a computer program

Description automatically generated

Hình . DiE nhận biết tệp tin bị nén bởi FSG

Đoạn mã trên là một rule file được sử dụng để phát hiện các tệp thực thi đã được nén hoặc đóng gói bằng FSG packer. Mỗi PE.compareEP so sánh điểm kết thúc (Entry Point) của tệp thực thi với các mẫu byte đặc trưng cho các phiên bản khác nhau của FSG.

* PE.compareEP(): Đây là một hàm trong DiE dùng để so sánh điểm kết thúc của một tệp thực thi với một mẫu byte cụ thể. Mỗi mẫu byte là đặc trưng cho một phiên bản hoặc kiểu đóng gói của tệp.
* Các mẫu byte: Mỗi điều kiện kiểm tra trong mã so sánh một chuỗi byte cụ thể với tệp thực thi. Nếu điểm kết thúc của tệp khớp với mẫu, DiE sẽ nhận diện được loại packer và cung cấp thông tin về phiên bản.
* Thông tin về phiên bản và tùy chọn: Khi phát hiện ra tệp sử dụng packer FSG, DiE sẽ xác định phiên bản của packer (ví dụ: 1.0, 1.10, 2.0...) và có thể cung cấp thêm thông tin về các tùy chọn nén (chẳng hạn như "Borland C++", "Microsoft Visual C++", "MASM32").

Mỗi khi DiE gặp một tệp thực thi, nó sẽ kiểm tra điểm kết thúc của tệp này và so sánh với các mẫu byte đã được định nghĩa. Nếu khớp, nó sẽ trả về thông tin về packer và phiên bản.

### PEiD

PEiD (Portable Executable Identifier), ra mắt vào năm 2008, là một công cụ mạnh mẽ dùng để phát hiện các packer, mã hóa, và trình biên dịch phổ biến cho các tệp thực thi PE. Hiện tại, PEiD có khả năng nhận diện hơn 600 mẫu chữ ký khác nhau trong các tệp PE. Công cụ này nổi bật nhờ những tính năng độc đáo so với các phần mềm nhận diện khác trên thị trường.

A close-up of a logo

Description automatically generated

Hình . Công cụ PeID

PEiD cung cấp giao diện người dùng (GUI) thân thiện, trực quan, và dễ sử dụng. Tỷ lệ phát hiện của công cụ nằm trong số tốt nhất so với các công cụ nhận diện hiện có. Ngoài ra, PeiD còn hỗ trợ các chế độ quét đặc biệt giúp phát hiện những tệp đã bị chỉnh sửa hoặc chưa được nhận diện. Công cụ này cũng tích hợp các tính năng như hỗ trợ dòng lệnh, chế độ hiển thị luôn trên cùng, khả năng kéo và thả, cùng khả năng quét nhiều tệp hoặc thư mục với tính năng đệ quy. Bên cạnh đó, PEiD còn cung cấp trình xem và quản lý tác vụ, cho phép người dùng dễ dàng theo dõi và kiểm soát các tiến trình đang chạy. Hơn nữa, giao diện plugin của công cụ hỗ trợ các plugin hữu ích như Generic OEP Finder và Krypto ANALyzer, giúp tăng cường khả năng phân tích. Để nâng cao hiệu quả phát hiện, PEiD sử dụng các kỹ thuật quét tiên tiến, bao gồm quét heuristic, và cung cấp thêm các tính năng như xem chi tiết tệp PE, các phần nhập (Imports), xuất (Exports), và TLS.

Công cụ này cũng được tích hợp sẵn một trình giải mã nhanh, giúp phân tích cú pháp mã hiệu quả. Ngoài ra, trình xem hex tích hợp cho phép người dùng xem nội dung hex của tệp một cách nhanh chóng. PEiD hỗ trợ quét nhiều tệp cùng lúc một cách hiệu quả. Người dùng chỉ cần kéo và thả các tệp vào giao diện chính của công cụ, sau đó một cửa sổ kết quả sẽ hiển thị thông tin chi tiết về từng tệp. Đặc biệt, PEiD hỗ trợ giao diện cập nhật chữ ký bên ngoài, cho phép người dùng tự cập nhật danh sách chữ ký để bắt kịp với các thay đổi trong môi trường.

**Các chế độ quét độc đáo trong PEiD**

PEiD cung cấp ba chế độ quét khác nhau, mỗi chế độ phù hợp với các nhu cầu cụ thể của người dùng.

1. Chế độ bình thường (Normal Mode): Chế độ này quét điểm bắt đầu (Entry Point) của tệp PE để tìm các chữ ký đã được ghi nhận. Đây là phương pháp quét cơ bản mà hầu hết các công cụ nhận diện khác cũng sử dụng.
2. Chế độ sâu (Deep Mode): Chế độ này quét toàn bộ phần chứa Entry Point của tệp PE để phát hiện các chữ ký. Phương pháp này đảm bảo phát hiện được khoảng 80% các tệp đã bị chỉnh sửa hoặc mã hóa.
3. Chế độ toàn diện (Hardcore Mode): Đây là chế độ quét toàn bộ tệp PE để tìm tất cả các chữ ký đã ghi nhận. Chế độ này phù hợp khi các chế độ khác không đạt kết quả như mong đợi. Tuy nhiên, do các chữ ký nhỏ có thể xuất hiện nhiều lần trong các tệp khác nhau, kết quả quét có thể không chính xác.

Dù vậy, PEiD đã tích hợp các cơ chế kiểm soát lỗi để đảm bảo kết quả chính xác ngay cả khi sử dụng chế độ toàn diện. Hai chế độ đầu tiên cho kết quả nhanh chóng, trong khi chế độ toàn diện có thể chậm hơn do phạm vi quét rộng hơn.

Lưu ý: PEiD là một file thực thi trên hệ điều hành Windows, nhưng đừng lo, để chạy trên hệ điều hành Linux, chỉ cần sử dụng lệnh “wine peid.exe”.

**Các packer được hỗ trợ bởi PEiD**

Hầu hết các trình đóng gói mà các công cụ được giới thiệu trên phát hiện được thì PEid cũng phát hiện được hết. Tuy nhiên, có một điểm đặc biệt ở PEiD, đó là việc lưu trực signature cho các trình đóng gói. Mỗi trình đóng gói đều có cho mình rất nhiều signature với đa dạng các phiên bản được thu thập từ nhiều nguồn khác nhau do nhóm phát triển cam kết cập nhật signature thường xuyên để theo kịp với những thay đổi trong lĩnh vực công nghệ. Vì thế, việc phát hiện file PE bị nén trở nên tối ưu hơn.

Các signature này sẽ được lưu trữ trong tệp tin “userdb.txt”. Với định dạng đơn giản này, người dùng có thể dễ dàng chỉnh sửa, thêm, xóa các signature.

A screenshot of a computer program

Description automatically generated

Hình . Signature của trình đóng gói FSG trong PEiD

A screenshot of a computer program

Description automatically generated

Hình . Signature của trình đóng gói UPX trong PEiD

Có một signature đặc biệt, nó được thiết kế để phát hiện một phiên bản UPX đặc biệt có bảo vệ mật khẩu, được gọi là Password Protector for the UPX 0.30, được phát triển bởi "g0d".

A black background with white text

Description automatically generated

Hình . Signature của trình đóng gói UPX đặc biệt trong PEiD

PEiD hoạt động như một công cụ phân tích tĩnh (static analysis), tập trung vào việc so khớp signatures với mục tiêu chính là phát hiện các công cụ đóng gói được sử dụng. Điểm mạnh của nó là cơ sở dữ liệu signature phong phú và tốc độ quét nhanh, phù hợp để phân tích sơ bộ các tệp tin PE.

**Tích hợp các tính năng bổ sung**:

* Task Viewer: Quét các tiến trình đang chạy trong bộ nhớ.
* Hex Viewer: Xem dữ liệu thô của file dưới dạng mã hex.
* Disassembler: Phân rã mã lệnh trong file để phân tích thủ công.
* Multiple File Scan: Hỗ trợ quét nhiều file hoặc thư mục cùng lúc.

# TRIỂN KHAI THỰC NGHIỆM

## Triển khai các công cụ đóng gói

### UPX

Download tại: <https://github.com/upx/upx/releases/tag/v4.2.3>

Cách cài đặt cũng như cách sử dụng UPX rất đơn giản. Đầu tiên, tải xuống phiên bản muốn sử dụng. Sau khi giải nén file tải được, chúng ta sẽ có được thư mục sau:

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

Description automatically generated

Hình . Thư mục chứa công cụ UPX

**A screenshot of a computer

Description automatically generated**

Hình . Giao diện chính của UPX

**Cách sử dụng UPX**

* Dùng lệnh “upx -o <output> <tên file muốn đóng gói>” để đóng gói file.

A screen shot of a computer

Description automatically generated

Hình . Dùng UPX để đóng gói tệp tin

* Kiểm tra kích thước tệp tin đã bị đóng gói



Hình . Trước khi đóng gói bằng UPX



Hình . Sau khi đóng gói bằng UPX

Có thể thấy, tệp tin trước khi đóng gói có kích thước là 73kb và sau khi đóng gói chỉ còn 47kb, kích thước được giảm đi một cách đáng kể.

* Cách giải nén tệp tin được đóng gói bằng UPX: sử dụng lệnh “upx -d <tên file muốn unpack>”

A screen shot of a computer

Description automatically generated

Hình . Giải nén tệp tin bị đóng gói

**Đặc trưng của file**

Có thể sử dụng các cách dưới đây để nhận biết một tệp tin bị đóng gói bởi công cụ UPX:

* Sử dụng các công cụ phân tích file như PEview hoặc CFF Explorer để xem thông tin định dạng của file PE. Trong phần "Sections" hoặc "Headers", có thể tìm kiếm các dấu hiệu của UPX, chẳng hạn như một section có tên "UPX" như "UPX0", “UPX1 hoặc "UPX2" hoặc các thuộc tính khác thường xuất hiện trong file được nén bởi UPX.
* Mở tệp trong PEView và vào text section và kiểm tra kích thước của raw data và virtual size. Nếu chênh lệch giữa hai giá trị rất lớn thì coi như tệp đã bị đóng gói. Đôi khi tất cả nội dung của text sections đều bằng không.

### FSG

Download tại: [FSG - Packers - Mente Binária](https://www.mentebinaria.com.br/files/file/24-fsg/)

Tệp tin tải về từ trang web sẽ là 1 tệp tin được đóng gói có định dạng “.zip”. Unzip file vừa tải về ta sẽ có được thư mục chứa các phiên bản khác nhau của công cụ FSG.

Ảnh có chứa ảnh chụp màn hình, thiết kế

Description automatically generated

Hình . Thư mục chứa file cài đặt FSG

Thư mục này chưa tổng cộng 5 phiên bản của công cụ FSG, trong đó có cả một phiên bản FSG dành cho mã hóa “FSG Crypter”. Trong mỗi thư mục này sẽ chứa các tệp tin thực thi tương ứng với từng phiên bản.

*Ảnh có chứa thiết kế, ảnh chụp màn hình

Description automatically generated*

Hình . Thư mục chứa các tệp tin thực thi FSG cho từng phiên bản

Vì FSG có định dạng thực thi là “.exe” tức là tệp tin thực thi dành cho hệ điều hành Windows nên để thực thi file “fsg.exe”, chúng ta cần sử dụng công cụ wine với dòng lệnh: “wine FSG.exe”.

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

Description automatically generated

Hình . Thực thi công cụ FSG

Do FSG là một công cụ sử dụng giao diện đồ họa nên bây giờ hộp thoại của công cụ FSG sẽ được hiển thị. Để sử dụng FSG trong việc đóng gói tệp tin cũng rất đơn giản, chúng ta chỉ cần chọn tệp tin cần cần đóng gói rồi nhấn open.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Description automatically generated

Hình . Chọn tệp tin để đóng gói trong FSG

Kết quả sau khi đóng gói tệp tin được thể hiện một cách trực quan. Hãy nhìn vào Hình 55, có thể dễ dàng thấy sự thay đổi trong kích thước của tệp tin được chọn.

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

Description automatically generated

Hình . Tệp tin được đóng gói bởi FSG

**Đặc trưng của file**

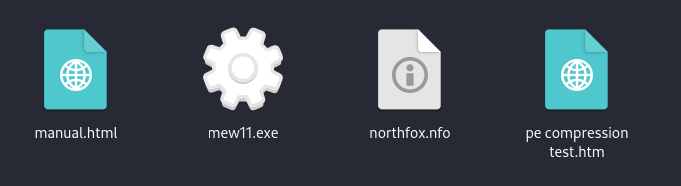
* Các tệp tin đã được mã hóa bằng FSG thường sẽ có một "signature" đặc biệt trong nội dung, ví dụ như chuỗi byte "FSG" hoặc một cấu trúc metadata riêng. Điều này có thể được dùng để nhận biết công cụ đã được sử dụng.
* Tuy nhiên, một số phiên bản FSG mới hơn có thể sẽ không có signature rõ ràng này, vì chúng sử dụng kỹ thuật mã hóa tinh vi hơn để che dấu dấu vết.

### MEW

Download tại:

<https://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/MEW-SE.shtml>

Tương tự như công cụ FSG ở trên, tệp tin MEW tải về cũng là một tệp tin có định dạng “.zip”. Giải nén tệp tin “.zip” vừa tải ta sẽ có thư mục chứa tệp tin thực thi.



Hình . Thư mục chứa công cụ MEW

Cũng là tệp tin thực thi dành cho hệ điều hành Windows, vì thế để thực thi MEW, chúng ta vẫn sẽ sử dụng “wine”. Có 2 cách để chạy tệp tin thực thi mew11.exe bằng lệnh wine:

* Cách 1: Sử dụng giao diện đồ họa.

Dùng lệnh “wine mew11.exe” để khởi chạy giao diện của MEW, sau đó, chọn tệp tin cần đóng gói tương tự như FSG.



Hình . Câu lệnh sử dụng giao diện đồ họa của MEW

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

Description automatically generated

Hình . Chọn tệp tin muốn đóng gói trong MEW

Hình 59. bên dưới là hình ảnh sau khi MEW đã đóng gói tệp tin thành công.

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

Description automatically generated

Hình . Kết quả đóng gói tệp tin bởi MEW

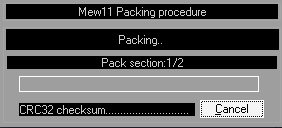
* Cách 2: Sử dụng câu lệnh.

Giao diện đồ họa ở trên tuy thân thiện với người dùng nhưng lại khá bất tiện trong việc đóng gói số lượng lớn tệp tin. Do đó, sử dụng câu lệnh sẽ được ưu tiên để tiết kiệm thời gian và tự động hóa.

Dùng câu lệnh “wine mew11.exe <file\_path>” với file\_path là đường dẫn đến tệp tin cần đóng gói.



Hình . Sử dụng câu lệnh để khởi chạy MEW



Hình . Quá trình đóng gói hàng loạt trong MEW

**Đặc trưng của file**

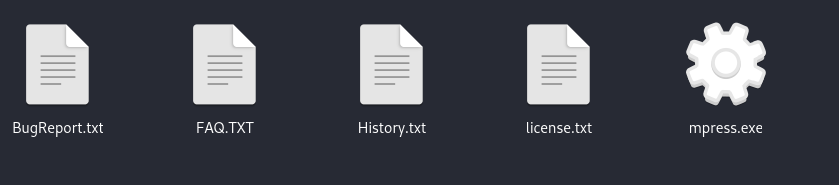
Dưới đây là một số dấu hiệu cho thấy tệp tin đã bị đóng gói bởi MEW:

* Nếu bạn sử dụng công cụ phân tích như CFF Explorer hoặc PE Explorer, có thể thấy thông tin packer hoặc các ký hiệu liên quan đến MEW. Tên các section có thể bị thay đổi hoặc bất thường, ví dụ: .MEW, .pack, hoặc tên rút gọn không liên quan.
* Khi mở tệp bằng hex editor, chúng ta có thể thấy các chuỗi đặc trưng của MEW, như: MEW hoặc MicroJoin xuất hiện trong phần đầu hoặc cuối của tệp.
* Tệp tin được đóng gói thường có entropy cao (trên 7.5), biểu thị rằng dữ liệu trong tệp đã bị nén hoặc mã hóa.

### MPRESS

Downlode tại: <https://github.com/requaos/mPress>

Không khác với các công cụ trên là mấy, sau khi tải công cụ Mpress về chúng ta cũng sẽ có được thư mục như bên dưới.

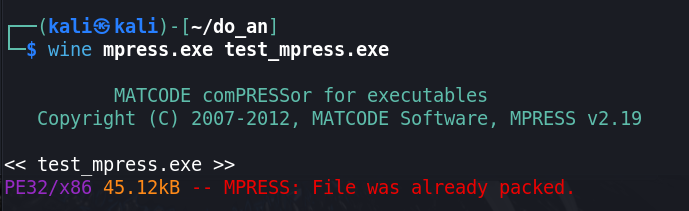


Hình . Thư mục chứa công cụ MPress

Tương tự, để chạy tệp tin thực thi mpress.exe, ta sẽ dùng công cụ wine để hỗ trợ. Cụ thể, dòng lệnh như sau:

“wine mpress.exe <file\_path>”

Với file\_path là đường dẫn đến tệp tin cần đóng gói.



Hình . Thực hiện đóng gói tệp tin bằng Mpress

**Đặc trưng của file**

Mpress cũng có một số dấu hiệu khá đặc trưng để nhận biết:

* Sử dụng công cụ phân tích như CFF Explorer hoặc PE Explorer và xem tên các sections như .MPRESS1, .MPRESS2, hoặc tên không chuẩn. Số lượng sections có thể giảm đáng kể (do tệp đã được nén, một số sections được hợp nhất).
* Khi mở tệp bằng Hex Editor, chúng ta có thể tìm thấy các chuỗi đặc trưng như “MPRESS” hoặc “mpress”. Chuỗi này có thể nằm ở đầu hoặc cuối tệp, tùy thuộc vào cách nén.
* Đặc điểm runtime khó chịu vì MPRESS có thể sử dụng các kỹ thuật chống dịch ngược, bao gồm sử dụng các lệnh không hợp lệ hoặc không chuẩn để gây lỗi khi phân tích bằng trình debugger và gây khó khăn cho việc breakpoint trong trình gỡ lỗi.

## Triển khai các công cụ phát hiện và giải nén các tập thực thi Windows

### Unipacker

Download tại đường dẫn: [unipacker/unipacker: Automatic and platform-independent unpacker for Windows binaries based on emulation](https://github.com/unipacker/unipacker)

Sau khi tải công cụ về, chúng ta có thư mục chứa rất nhiều tệp tin. Ngoài các tệp tin để cài đặt, thư mục này còn chứa một số sample cho người dùng sử dụng để thử nghiệm công cụ mà không cần tốn thời gian thu thập thêm.



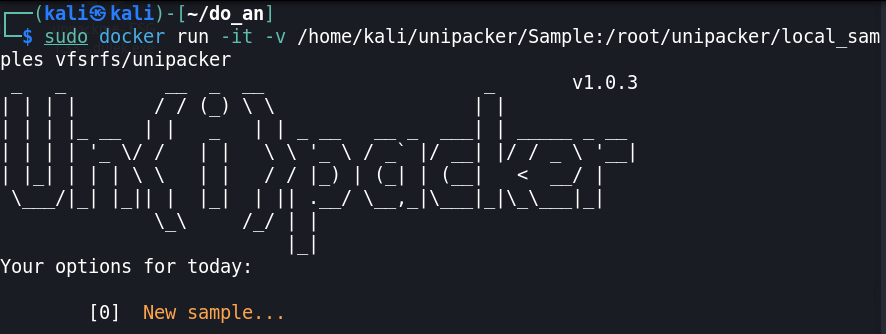
Hình . Thư mục chứa công cụ Unipacker

Sau đó ta có thể chạy file setup.py để có thể thực thi unipacker trên cmd. Tuy nhiên, nhóm đã gặp khó khăn trong việc cài đặt vì lỗi liên quan đến yara. Khi nhóm thử chạy trên máy không gặp lỗi yara thì vẫn không thể chạy unipacker trực tiếp trên cmd, nên nhóm đã chọn thực thi bằng cách chạy trên docker với máy gặp lỗi yara và thực hiện thông qua ~/unipacker/shell.py.

Để thực thi bằng docker, chúng ta thực hiện câu lệnh sau:

“sudo docker run -it -v <folder\_path>:/root/unipacker/local\_samples vfsrfs/unipacker”

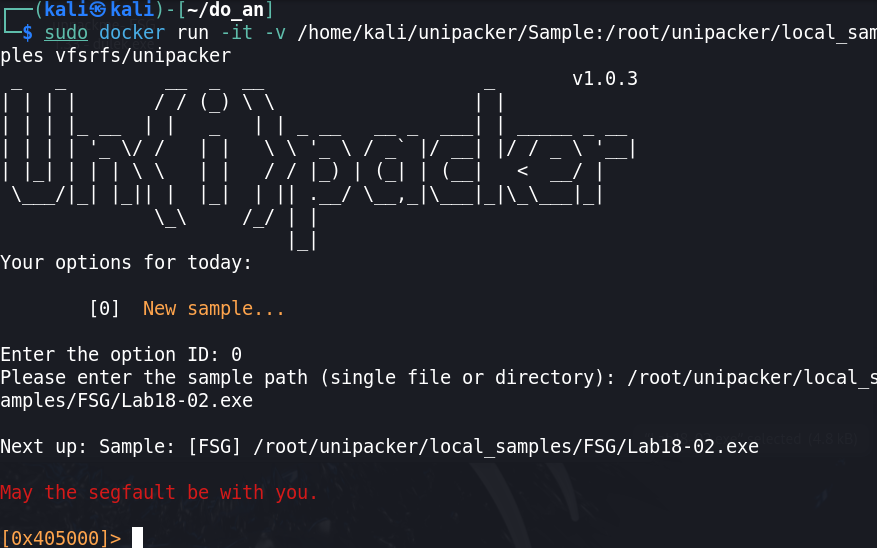
Với folder\_path là đường dẫn đến thư mục chứa các sample muốn đóng gói.



Hình . Giao diện chính của công cụ Unipacker

Khi cmd hiện dòng “Enter the option ID:”, lúc này chúng ta sẽ được lựa chọn các tệp tin muốn kiểm tra. Hiện tại trên màn hình chỉ mới có option “[0] New Sample” vì đây là lần đầu tiên ta mở công cụ nên chưa có các option khác. Nếu đã thử đưa các tệp tin vào kiểm tra rồi, thì màn hình sẽ hiện thêm các option chứa các tệp tin đã kiểm tra đó, giúp người dùng chọn lại mà không nhất thiết phải tra lại đường dẫn đã từng kiểm tra.

Ở đây, nhóm sẽ thử kiểm tra một tệp tin mới, nhập ID [0] vào rồi nhập đường dẫn đến tệp tin muốn kiểm tra:



Hình . Kiểm tra một tệp tin bị đóng gói bằng Unipacker

Để xem thông tin chi tiết của tệp tin, nhập lệnh “aaa”:



Hình . Kiểm tra thông tin tệp trong Unipacker

Theo Hình 67, ta có được thông tin là file được phân tích là file PE32 và được đóng gói bởi công cụ FSG.

**Thống kê PE**

Trong PE stats sẽ hiển thị kích thước bộ nhớ được khai báo mà tệp PE sẽ chiếm và bao nhiêu thực tế được sử dụng (để phát hiện sự khác biệt dễ dàng). Thêm vào đó, còn có địa chỉ cơ sở image được hiển thị (thường là 0x400000 hoặc 0x10000000) và các không gian địa chỉ đã dự trữ cho bộ nhớ stack và hook API-call.

A screen shot of a computer

Description automatically generated

Hình . Các thông số PE thống kê được trong Unipacker

Giải thích:

* Declared virtual memory size: 0x17000: Đây là kích thước bộ nhớ ảo mà chương trình yêu cầu khi được tải vào bộ nhớ (tương đương với 94,208 bytes).
* Actual loaded image size: 0x5200: Đây là kích thước thực tế của chương trình (image) khi nó được tải vào bộ nhớ (tương đương với 20,99 bytes).

**Import**

Trong trường Import sẽ liệt kê các thư viện được khai báo trong PE header bao gồm cả thư viện tĩnh lẫn thư viện động.

A screenshot of a computer program

Description automatically generated

Hình . Thư viện tĩnh trong PE header

*A screen shot of a computer

Description automatically generated*

Hình . Thư viện động trong PE header

**Trạng thái thanh ghi**

Trong trường Register status sẽ hiển thị giá trị của các thanh ghi mô phỏng. Tại thời điểm này, chúng vẫn là các thiết lập từ quá trình tải.

A black background with white text

Description automatically generated

Hình . Trạng thái của các thanh ghi trong file PE

**Giải nén tệp tin**

Sử dụng lệnh “r” để giải nén tệp tin đang khai thác.

A computer screen shot of a computer program

Description automatically generated

Hình . Giải nén tệp tin trong Unipacker

Quá trình giải nén sẽ được diễn ra và tệp tin đã giải nén sẽ có tên là “unpacked.exe”. Tuy nhiên, do nhóm đang sử dụng docker để triển khai công cụ này nên file giải nén trên sẽ không thể được tìm thấy bằng các lệnh mặc định trong Unipacker.

Trong dockerfile của nhà phát hành đặt ENTRYPOINT là “unipacker” nên khi chúng ta chạy lệnh “docker run” nó sẽ mặc định gọi luôn công cụ.

A screen shot of a computer

Description automatically generated

Hình . Dockerfile của Unipacker

Vì thế, để có thể lấy được file “unpacked.exe”, chúng ta cần mở shell để tiến hành tìm kiếm bằng cách đặt lại ENTRYPOINT là /bin/sh

sudo docker run -it --entrypoint /bin/sh -v ~/DACN/unipacker/Sample:/root/unipacker/local\_samples -v ~/DACN/unipacker/results:/root/unipacker/results vfsrfs/unipacker



Hình . Truy cập shell để lấy tệp tin giải nén trong Unipacker

Chú thích:

* ~/DACN/unipacker/Sample: thư mục trên máy chủ chứa file cần khai thác.
* /root/unipacker/local\_samples: thư mục trong container của docker chứa file cần khai thác.
* ~/DACN/unipacker/results: thư mục trên máy chủ chứa file trả về.
* /root/unipacker/results: thư mục trong container của docker chứa về file unpack trả về.
* Lệnh -v để mout từ docker sang máy chủ.

Sau khi chạy lệnh trên ta sẽ vào được shell và dễ dàng tương tác hơn. Bây giờ trong container đã có 2 thư mục mà ta đã mout vào.

A screen shot of a computer program

Description automatically generated

Hình . Mở shell trong Unipacker

Thực hiện gọi công cụ Unipacker. Trong đó, lệnh “-d” là destination chứa thư mục trả về mong muốn trong docker.

A screen shot of a computer screen

Description automatically generated

Hình . Bắt đầu sử dụng Unipacker như bình thường

Tiến hành phân tích và giải nén như các bước trên, ta sẽ có được tệp tin đã giải nén “unpacked.exe” trong thư mục results.



Hình . Kết quả thu được tệp tin đã giải nén

### ClamAV

Cài đặt công cụ ClamAV tương đối dễ hơn, thực thi câu lệnh “sudo apt install clamav clamav-daemon” để cài đặt.

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

Description automatically generated

Hình . Cài đặt ClamAV

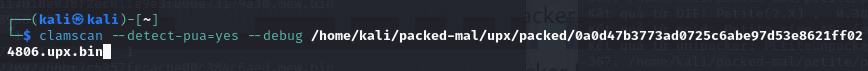
Sau đó dùng câu lệnh “sudo freshclam” để đảm bảo rằng cơ sở dữ liệu của clamav được cập nhật.

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

Description automatically generated

Hình . Cập nhật cơ sở dữ liệu cho ClamAV

Dùng câu lệnh “clamscan --debug --detect-pua=yes <file\_path>” với file\_path là đường dẫn đến tệp tin cần phân tích để tiến hành quét.



Hình . Tiến hành quét tệp tin bằng ClamAV

Kết quả sau khi chạy câu lệnh trên là tệp tin bị nén bởi trình đóng gói UPX.

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

Description automatically generated

Hình . Kết quả khi quét tệp tin bằng ClamAV

Để trích xuất tệp mà clamav giải nén được, ta sử dụng tùy chọn “–leave-temp=yes”.



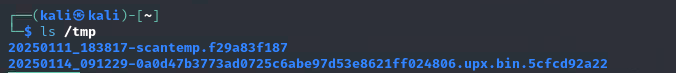
Hình . Tùy chọn để lấy tệp tin đã giải nén bằng ClamAV

Sau khi chạy câu lệnh, ta thấy trong phần command line có thông báo như Hình 83. Lúc này, tệp tin đã được giải nén nhưng chỉ nằm ở bộ nhớ tạm.



Hình . Đường dẫn đến tệp tin đã giải nén bằng ClamAV

Đi theo đường dẫn lấy được ở bên trên, ta sẽ đến thư mục /tmp, ở đây sẽ chứa tệp tin đã được giải nén cũng như những thông tin liên quan đến tệp được quét.



Hình . Tệp tin đã được giải nén bằng ClamAV

ClamAV có khả năng phát hiện và giải nén các tệp tin khá tốt, tuy nhiên, thời gian chạy của nó nhiều nhất trong số bốn công cụ được đề xuất.

### PEiD

PEiD được cài đặt bằng pipx, nên trước đó cần phải chạy dòng lệnh sau: “pipx ensurepath --force”. Câu lệnh này được sử dụng để đảm bảo rằng thư mục cài đặt của pipx đã được thêm vào biến môi trường PATH trên hệ thống.

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

Description automatically generated

Hình . Cài đặt biến môi trường cho pipx

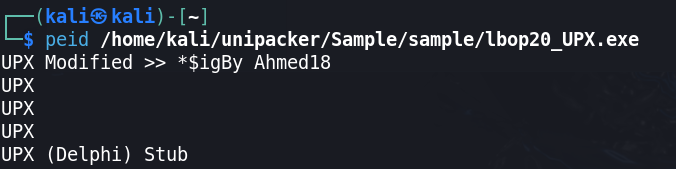
Sau đó chạy câu lệnh ‘pipx install peid’ để cài đặt peid.

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

Description automatically generated

Hình . Cài đặt PEiD bằng pipx

Sau khi đã cài đặt xong, ta sử dụng câu lệnh “peid <file\_path>” (với file\_path là đường dẫn đến tệp tin cần phân tích) để phân tích packer mà tệp tin đã sử dụng. Lưu ý: câu lệnh bên dưới dùng cho phiên bản dòng lệnh trên Linux, nếu muốn sử dụng phiên bản có giao diện, người dùng có thể cài đặt và sử dụng PEiD bằng công cụ wine.



Hình . Quét tệp tin bằng PEiD phiên bản câu lệnh

A screenshot of a computer program

Description automatically generated

Hình . Quét tệp tin bằng PEiD phiên bản giao diện người dùng

Có thể thấy PEiD ở cả hai phiên bản đều có thể quét đúng trình đóng gói cho tệp tin được chọn. Tuy nhiên, ở phiên bản người dùng thì trực quan hơn, có thêm phiên bản của trình đóng gói được sử dụng cũng như thông tin cụ thể của tệp tin. Và đặc biệt, phiên bản command line không thể quét nhiều tệp tin cùng một lúc, trong khi phiên bản giao diện thì có thể (sử dụng tab Multi Scan)

Và khi chọn tab Options, người dùng còn có thể chọn chế độ quét gồm Normal Scan, Deep Scan, Hardcore Scan như được giới thiệu ở Chương 2.

A screenshot of a computer program

Description automatically generated

Hình . Ba chế độ quét trong PEiD

### DIE

Download tại:

* <https://github.com/horsicq/DIE-engine> (phiên bản command line)
* <https://github.com/horsicq/Detect-It-Easy/> (phiên bản GUI)

Cài đặt DiE tương đối dễ. Sau khi tải công cụ về, ta sẽ có thư mục chứa hàng loạt file cấu hình.

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

Description automatically generated

Hình . Thư mục chứa tệp tin cấu hình DiE

Dùng câu lệnh “chmod +x install.sh” để cấp quyền thực thi cho tệp tin install.sh – một tệp tin dùng để cài đặt DiE. Sau đó dùng lệnh “sudo ./install.sh” để tiến hành thực thi.

Sau khi cài đặt xong dùng lệnh “diec <file\_path>” với file\_path là đường dẫn tới tệp tin cần phân tích để sử dụng phiên bản command line.

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

Description automatically generated

Hình . Phân tích tệp tin với DiE phiên bản command line

DiE đã phát hiện được tệp tin được chọn bị đóng gói bởi UPX phiên bản 3.95.

Ngoài ra, còn có phiên bản giao diện người dùng. Chỉ cần lệnh “die” để mở phiên bản này. Phiên bản này trực quan hơn, kết quả cho ra tương tự phiên bản command line nhưng hiển thị chi tiết và nhiều thông tin hơn về tệp tin bị đóng gói.

A screenshot of a computer program

Description automatically generated

Hình . Phân tích tệp tin bằng DiE phiên bản GUI

Và lệnh “diel” dành cho phiên bản có giao diện nhẹ nhất. Tuy nhiên, phiên bản này hầu như không thể phát hiện được các trình đóng gói bởi cấu hình cũng như database đơn giản.

A screenshot of a computer

Description automatically generated

Hình . Phân tích tệp tin bằng DiE phiên bản GUI nhẹ

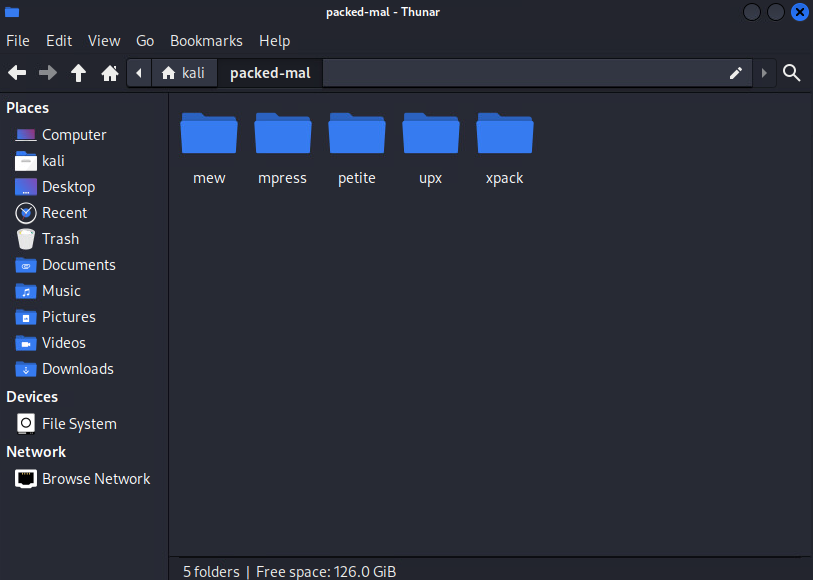
# KẾT QUẢ THỰC NGHIỆM

## Bộ dữ liệu

Trong phần này, nhóm đã thực hiện đánh giá hiệu quả của các công cụ phát hiện packer đã đề cập trong phần lý thuyết trên để thực hiện phát hiện các packer trong tập dữ liệu.

Dataset được sử dụng cho phần này bao gồm các tệp thực thi đã được đóng gói (packed) bằng các packer phổ biến. Dataset được tổ chức thành các thư mục, trong đó mỗi thư mục tương ứng với một loại packer. Tên của packer được đặt làm tên thư mục, giúp dễ dàng phân loại và quản lý. Chi tiết dataset như sau:

* Thư mục xpack: Chứa 9980 tệp đã được đóng gói bởi packer xpack.
* Thư mục mew: Chứa 7350 tệp đã được đóng gói bởi packer mew.
* Thư mục petite: Chứa 6530 tệp đã được đóng gói bởi packer petite.
* Thư mục mpress: Chứa 4790 tệp đã được đóng gói bởi packer mpress.
* Thư mục upx: Chứa 5200 tệp đã được đóng gói bởi packer upx.



Hình . Tập dataset

Để đảm bảo quá trình thực nghiệm hiệu quả và tiết kiệm thời gian, nhóm đã chọn ngẫu nhiên:

* 2000 tệp từ mỗi thư mục để kiểm thử với các công cụ DIE, PEiD, và unipacker.
* 200 tệp từ mỗi thư mục để kiểm thử với clamav (do thời gian chạy của clamav tương đối dài).

Mỗi công cụ sẽ hỗ trợ những loại packer khác nhau, nên nhóm chỉ thực hiện thực nghiệm và lấy số liệu đối với những packer mà tài liệu của công cụ xác nhận rằng có thể nhận biết được.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | DIE | PEiD | clamav | Unipacker |
| xpack | X | X |  |  |
| MEW | X | X | X | X |
| Petite | X | X | X | X |
| MPRESS | X | X |  | X |
| UPX | X | X | X | X |

Bảng . Các packer mà công cụ có thể phát hiện được

*Giải thích*: ô chứa dấu “x” nghĩa là công cụ có thể phát hiện được packer tương ứng.

Có thể thấy, với số lượng trình đóng gói mà nhóm sử dụng để thực nghiệm, DiE và PEiD đều có thể phát hiện được hết bởi dataset của hai công cụ này là rất lớn và được cập nhật liên tục. Còn ClamAV và Unipacker có vẻ kém lợi thế hơn một chút.

## Cấu hình máy sử dụng cho thực nghiệm

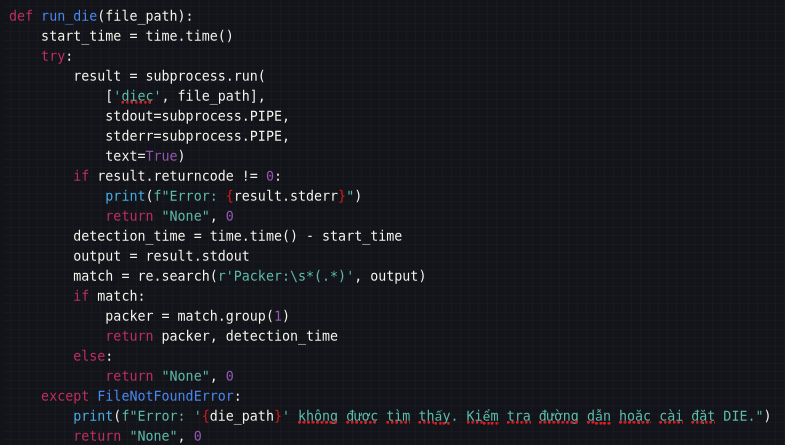
Khi thực hiện thực nghiệm, nhóm đã sử dụng máy ảo với cấu hình sau:

* Hệ điều hành: kali linux
* Dung lượng RAM: 6GB
* Số bộ xử lý (Processors): 4CPU
* Dung lượng ổ cứng: 200GB

## Tự động hóa

Nhằm tự động hóa việc chạy các công cụ trên, nhóm đã viết đoạn code sau:

* Đoạn code tự động hóa của công cụ DiE



Hình . Đoạn code tự động hóa của công cụ DiE

Trong đoạn code trên, nhóm đã cho chạy lệnh “diec <file\_path>”. Sau đó dùng biến “re” để tìm dòng ‘Packer: ’ trong phần output được trả về, nếu tìm thấy sẽ trả về tên packer đó kèm theo thời gian chạy lệnh. Thời gian chạy này được tính từ lúc vừa vào hàm cho đến lúc lấy được output.

* Đoạn code tự động hóa của công cụ PEiD.

Ảnh có chứa văn bản, ảnh chụp màn hình

Description automatically generated

Hình . Đoạn code tự động hóa của công cụ PEiD

Cách hoạt động của đoạn code này khá giống với đoạn code dùng để tự động hóa DIE, chỉ khác nhau ở cách lấy kết quả đầu ra. Ở đoạn code này, kết quả đầu ra được lấy bằng cách lấy dòng đầu tiên của output.

* Đoạn code tự động hóa của công cụ Unipacker.

Ảnh có chứa văn bản, ảnh chụp màn hình

Description automatically generated

Hình . Đoạn code tự động hóa của công cụ Unipacker

Đoạn code này thực hiện chạy dòng lệnh “python <uni\_path>” với uni\_path là đường dấn đến tệp tin python chạy Unipacker. Đợi đến khi có dòng ‘Enter the option ID: ’ khi đó ta sẽ nhập id vào. Id này được tính bằng cách đếm số dòng sau dòng ‘Your option for today: ’. Tiếp theo ta lần lượt nhập file\_path của tệp tin cần phát hiện và câu lệnh “aaa” để lấy thông tin của tệp được chọn. Đợi đến khi unipacker trả về kết quả, tách packer từ output của câu lệnh rồi trả về packer và thời gian chạy của unipacker.

* Đoạn code tự động hóa của công cụ Clamav.

Ảnh có chứa văn bản, ảnh chụp màn hình

Description automatically generated

Hình . Đoạn code tự động hóa của công cụ ClamAV

Vì Clamav thường xuyên gặp lỗi “zsh: killed” nên ở đây nhóm hạn chế số lần chạy clamav là 5 lần, nếu số lần chạy chưa đạt đến 5 mà gặp lỗi thì chương trình sẽ chạy lại. Câu lệnh chạy clamav trong cmd là ‘clamscan –debug –detect-pua=yes <file\_path>’. Sau đó lấy packer được trả về trong output bằng các nội dung đặc trưng trong output của clamav.

Sau khi chạy tất cả các đoạn code trên, nhóm sẽ lưu các kết quả vào một file có định dạng .csv để tiện cho việc theo dõi và thống kê.

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

Description automatically generated

Hình . Lưu kết quả tự động hóa

Để chạy số lượng tệp tin lớn như vậy, việc sử dụng cả thư mục chứa các tệp tin có thể là một lựa chọn tốt. Tuy nhiên, không phải công cụ nào cũng có thể làm được như vậy. Vì thế cần một đoạn code để đưa lần lượt các tệp tin vào từng công cụ.

* Đoạn code đọc các tệp tin trong folder đầu vào, chạy các công cụ phát hiện và lưu kết quả vào file csv.

Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn

Description automatically generated

Hình . Đoạn code đọc các tệp tin trong folder đầu vào

* Hàm main nhận đầu vào là 1 folder chứa các tệp tin cần phân tích

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

Description automatically generated

Hình . Hàm lấy folder đầu vào

Sau khi chạy đoạn code trên, nhóm sẽ có các file .csv tại thư mục result. Các file chứa thông tin về từng tệp tin như: File Name, Detected Packer, Detection Time, Correct/Incorrect.

Ảnh có chứa văn bản, ảnh chụp màn hình

Description automatically generated

Hình . Kết quả quét lần lượt các tệp tin

Để đánh giá so sánh kết quả của các công cụ với nhau, nhóm đã viết 1 đoạn code để tính toán phần trăm phát hiện được packer, phần trăm chính xác sau khi đã phát hiện được packer và thời gian chạy của của từng công cụ, cụ thể như sau:

* Hàm đếm tổng các file đã được phát hiện:

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

Description automatically generated

Hình . Hàm đếm các file đã phát hiện được

* Hàm tính phần trăm phát hiện được packer của các công cụ bằng cách lấy giá trinh của hàm count\_number\_detected\_file(file\_path) chia cho tổng số file đã thực nghiệm (200 đối với clamav và 2000 đối với các công cụ khác):

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

Description automatically generated

Hình . Hàm tính tính phần trăm phát hiện packer của các công cụ

* Hàm tính phần trăm chính xác của công cụ. Hàm này được tính bằng cách lấy tổng của cột ‘Correct/Incorrect’ (cột này lưu giá trị ‘1’ nếu phát hiện đúng và ‘0’ nếu phát hiện sai hoặc không phát hiện) chia cho số lượng các file bị pack mà công cụ đã phát hiện được:

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

Description automatically generated

Hình . Hàm tính phần trăm chính xác của các công cụ

* Hàm tính thời gian trung bình khi chạy công cụ:

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

Description automatically generated

Hình . Hàm tính thời gian chạy trung bình của các công cụ

* Sau cùng là hàm xử lý chính của chương trình. Hàm này thực hiện nhiệm vụ là scan các file trong thư mục con, gọi các hàm đã viết ở phía trên rồi in kết quả ra màn hình, đồng thời dựa vào những số liệu đó tính toán các số liệu như: thời gian chạy trung bình, phần trăm phát hiện và phần trăm chính xác của các loại công cụ trên, rồi in kết quả ra màn hình:

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

Description automatically generated

Hình . Hàm scan file, thực hiện gọi hàm, tính toán rồi in kết quả ra màn hình

Kết quả khi chạy đoạn code trên (kết quả chỉ xét trên những packer mà tài liệu công bố của công cụ ghi có thể phát hiện được):

Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, tài liệu

Description automatically generated

Hình . Thời gian trung bình của các công cụ

Ta có thể thấy, đối với mỗi loại packer, các công cụ lại có thời gian chạy và độ chính xác khác nhau. Dưới đây là bảng minh họa cho sự khác biệt đó, qua đó giúp ta đánh giá được hiệu suất và độ tin cậy của từng loại công cụ trong thực nghiệm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **DIE** | **PEiD** | **Clamav** | **Unipacker** |
| **Thời gian chạy** | xpack | 0.20s | 0.34s | x | x |
| UPX | 0.30s | 0.36s | 43.74s | 0.27s |
| MEW | 0.26s | 0.36s | 42.15s | 0.23s |
| PEtite | 0.28s | 0.36s | 44.39s | 0.24s |
| MPRESS | 0.27s | 0.34s | x | 0.22s |
| **Phần trăm phát hiện** | xpack | 74.7% | 94.20% | x | x |
| UPX | 99.4% | 99.6% | 75.5% | 99.25% |
| MEW | 99.6% | 99.65% | 95.5% | 99.15% |
| Petite | 99.85% | 99.95% | 81% | 99.6% |
| MPRESS | 93.4% | 95.95% | x | 83.95% |
| **Độ chính xác** | xpack | 99.67% | 79.14% | x | x |
| UPX | 100% | 96.67% | 93.38% | 100% |
| MEW | 100% | 99.80% | 98.43% | 97.68% |
| Petite | 100% | 99.85% | 99.38% | 95.88% |
| MPRESS | 99.73% | 86.40% | x | 94.22% |

Bảng . Thời gian chạy và độ chính xác của từng công cụ

Kết hợp lý thuyết và kết quả trên, ta có bảng so sánh các công cụ như sau (chỉ xét trên các loại packer mà công cụ có thể phát hiện):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **DIE** | **PEiD** | **Clamav** | **Unipacker** |
| **Cài đặt dễ hay khó?** | Dễ | Dễ | Dễ | Khó |
| **Môi trường sử dụng** | Windows, Linux, macOS | Windows, Linux | Windows, Linux, macOS | Windows, Linux |
| **Ưu điểm** | Phát hiện heuristic và signature linh hoạt | Cơ sở dữ liệu signature phông phú; tốc độ quét nhanh; giao diện trực quan, dễ sử dụng | Hỗ trợ phát hiện virus và malware trên nhiều nền tảng | Hỗ trợ phân tích runtime |
| **Nhược điểm** | Không hỗ trợ giải nén | Phụ thuộc vào signature, không phát hiện được các packer mới hoặc phức tạp | Khả năng phát hiện packer yếu, tập trung vào malware hơn là phân tích tệp nén | Chỉ hỗ trợ phát hiện một số loại packer cụ thể |
| **Có phát hiện file nén không?** | Có thể phát hiện 93.39% file nén | Có thể phát hiện 98.35% file nén | Có thể phát hiện 84% file nén | Có thể phát hiện 99.33% file nén |
| **Phát hiện đúng hay sai?** | Đúng 99.88% | Đúng 92.38% | Đúng 97.06% | Đúng 96.95% |
| **Thời gian chạy trung bình** | 0.26s | 0.35s | 43.43s | 0.24s |
| **Có thể giải nén không?** | Không | Có thể nếu dùng plugins và chỉ 1 ít packer | Có thể nhưng chỉ đối với vài packer | Có thể với 1 vài packer |

Bảng . So sánh các công cụ

# HƯỚNG PHÁT TRIỂN

## Kết luận

Trong lĩnh vực phát hiện và giải nén các tệp tin thực thi Windows bị đóng gói, cả bốn công cụ Unipacker, ClamAV, Detect It Easy (DIE) và PEiD đều đã khẳng định được giá trị và khả năng vượt trội của mình. Mỗi công cụ đều mang đến những dòng lệnh, những cách thức và phương pháp riêng biệt nhưng nhìn chung, chúng đều hướng tới một mục đích chung, đó là hỗ trợ các chuyên gia bảo mật, các nhà nghiên cứu, các doanh nghiệp và cá nhân thực hiện các nhiệm vụ như phát hiện, giải mã và phân tích mã độc.

Unipacker là công cụ chuyên dụng cho việc giải nén các tệp tin bị đóng gói. Với khả năng tự động hóa cao, nó giảm thiểu đáng kể thời gian xử lý thủ công, giúp các chuyên gia nhanh chóng tiếp cận với mã nguồn gốc của các tệp tin để phân tích sâu hơn. Tuy nhiên, Unipacker đôi khi có thể gặp khó khăn với các tệp tin sử dụng trình đóng gói mới hoặc ít phổ biến hơn, điều này đòi hỏi sự kết hợp với các công cụ khác để đạt được hiệu quả tối ưu. Và đặc biệt, trong việc giải nén, Unipacker vẫn chưa thực sự thuần thục và khiến cho việc truy xuất dữ liệu gốc vẫn còn khó khăn.

ClamAV, nổi tiếng với khả năng phát hiện mã độc và xử lý các tệp tin bị nhiễm, không chỉ dừng lại ở chức năng quét mà còn hỗ trợ giải nén một số tệp tin bị đóng gói. Ưu điểm lớn của ClamAV là khả năng cập nhật cơ sở dữ liệu liên tục, giúp nó nhận diện các mối đe dọa mới một cách nhanh chóng. So với Unipacker, ClamAV có phần nhỉnh hơn về tính linh hoạt và phạm vi ứng dụng, đặc biệt khi giải nén tệp tin. Một tệp tin được giải nén bởi ClamAV nhìn chung có sự thay đổi về kích thước so với tệp tin ban đầu, tuy nhiên việc này không ảnh hưởng đáng kể đến chức năng của tệp tin bởi phần kích thước chênh lệch chỉ là signature của ClamAV được thêm vào mà thôi.

Detect It Easy (DIE) lại tập trung vào việc nhận diện định dạng tệp tin và các trình đóng gói, cryptor, hoặc compiler. Công cụ này nổi bật nhờ giao diện thân thiện, khả năng tùy chỉnh cao, và hỗ trợ mở rộng với các plugin. Detect It Easy cung cấp thông tin chi tiết về cấu trúc tệp tin, giúp người dùng nắm bắt được những đặc điểm quan trọng mà không cần đi sâu vào phân tích mã. Điều này làm cho DIE trở thành một phần không thể thiếu trong việc phát hiện các trình đóng gói phổ biến và ít phổ biến hơn.

PEiD là một công cụ đã quá quen thuộc trong việc phát hiện các trình đóng gói nhờ khả năng nhận diện hàng trăm signature khác nhau. Với ba chế độ quét linh hoạt – Normal, Deep, và Hardcore – PEiD có thể xử lý cả các tệp tin bị chỉnh sửa hoặc biến đổi phức tạp. Bên cạnh đó, các tính năng như xem thông tin chi tiết của tệp tin PE, tích hợp trình xem hex, và hỗ trợ plugin giúp PEiD mở rộng khả năng phân tích của mình. Dù đã ra mắt từ lâu, công cụ này vẫn giữ được sự hữu ích nhờ tính ổn định và hiệu quả.

Nhìn chung, mặc dù mỗi công cụ đều có điểm mạnh và hạn chế riêng, sự kết hợp của chúng giúp khắc phục được nhiều vấn đề trong việc phát hiện và giải nén tệp tin thực thi bị đóng gói. Unipacker và ClamAV nổi bật trong khả năng giải nén, Detect It Easy và PEiD lại đóng vai trò quan trọng trong việc nhận diện các trình đóng gói, mang đến thông tin chính xác về cấu trúc tệp tin và chữ ký của chúng vì liên tục được cập nhật.

Sự phát triển không ngừng của công nghệ đóng gói và mã hóa tệp tin đòi hỏi các công cụ này phải được cải tiến liên tục. Tuy nhiên, với những tính năng hiện tại, chúng đã và đang hỗ trợ đắc lực cho các chuyên gia bảo mật, góp phần đảm bảo an toàn thông tin trước những mối đe dọa ngày càng tinh vi và đa dạng.

## Hướng phát triển

Mặc dù các công cụ Unipacker, ClamAV, Detect It Easy (DIE) và PEiD đã chứng minh được hiệu quả trong việc phát hiện và giải nén tệp tin bị đóng gói, nhưng nếu chúng ta có thể kết hợp các tính năng của chúng lại với nhau, sẽ tạo ra một sản phẩm mạnh mẽ và toàn diện hơn, phục vụ hiệu quả hơn cho các chuyên gia bảo mật và những ai cần phân tích các tệp tin thực thi.

Một trong những hướng phát triển tiềm năng là tích hợp các công cụ lại thành một hệ thống duy nhất, nơi mỗi công cụ sẽ hỗ trợ lẫn nhau, nâng cao hiệu quả của quy trình phát hiện và giải nén. Ví dụ, PEiD và Detect It Easy có thể được kết hợp để nhận diện chính xác các trình đóng gói và mã hóa tệp tin, trong khi Unipacker và ClamAV có thể đóng vai trò quan trọng trong việc giải nén tệp tin, phát hiện mã độc và làm sạch các tệp tin đã bị nhiễm.

Bên cạnh đó, có thể xây dựng một giao diện người dùng (GUI) tập trung vào việc tích hợp các công cụ này. Giao diện sẽ cho phép người dùng chọn một tệp tin để quét, và hệ thống sẽ tự động sử dụng các công cụ phù hợp trong từng bước: phát hiện trình đóng gói (PEiD, DIE), giải nén (Unipacker, ClamAV), và cuối cùng là quét mã độc (ClamAV).

Bên cạnh đó, việc cập nhật cơ sở dữ liệu signature cho các công cụ sẽ giúp hệ thống này luôn cập nhật và phát hiện được các tệp tin đóng gói mới, hay các tệp tin đã được mã hóa bằng những phương thức chưa được biết đến. Cơ sở dữ liệu signature có thể được tự động cập nhật từ các nguồn uy tín hoặc cho phép người dùng đóng góp thêm chữ ký mới từ những tệp tin đã được phân tích.

Ngoài ra, để tăng tính tự động hóa và tiện lợi, sản phẩm hoàn chỉnh này có thể được phát triển thêm với khả năng quét hàng loạt tệp tin (batch scan) hoặc tích hợp vào các hệ thống bảo mật như SIEM (Security Information and Event Management), giúp giám sát và phát hiện các mối đe dọa trong thời gian thực.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | R. CREW, "Phân biệt vai trò của Packer, Crypter, và Protector trong các phần mềm độc hại," *inseclab.uit.edu.vn,* 2020. |
| [2] | D. Das, "What Is the Windows Portable Executable File Format?," p. MAKE USE OF, 2023. |
| [3] | BLUE\_TEAM, "BÁO CÁO TÌNH HÌNH ATTT (06 tháng đầu năm 2024)," no. Chia sẻ kỹ thuật, p. VNPT Cycber Immunity, 2024. |
| [4] | The UPX Team, "UPX," 2024. |
| [5] | "ASPack - exe file compressor allows to protect application from reverse engineering decompilation and disassemblers". |
| [6] | requaos, "mPress, Github," 2009. |
| [7] | "MEW," 2004. |
| [8] | "Unipacker: Automatic and platform-independent unpacker for Windows binaries based on emulation, Github," 2014. |
| [9] | J. Collake, "PECompact – Windows (PE) Executable Compressor," 2017. |
| [10] | C. Team, "ClamAV". |
| [11] | Horsicq, "Detect It Easy (DiE)," 2012. |
| [12] | wolfram77web, "PEiD detect most common packers, cryptors and compilers for PE files," 2008. |