Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM
Khoa Mạng máy tính và truyền thông

BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

Môn học: Cơ chế hoạt động của mã độc

Tên chủ đề: MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images

Mã nhóm: 04 Mã đề tài: 04

Lớp: NT230.O21.ANTN

1. THÔNG TIN THÀNH VIÊN NHÓM:

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

STT	Họ và tên	MSSV	Email
1	Lưu Gia Huy	21520916	21520916@gm.uit.edu.vn
2	Ngô Thanh Sang	21522543	21522543@gm.uit.edu.vn
3	Đoàn Thị Ánh Dương	21521987	21521987@gm.uit.edu.vn

2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

A. Chủ đề nghiên cứu trong lĩnh vực Mã độc: *(chọn nội dung tương ứng bên dưới)*

☑ Phát hiện mã độc

☐ Đột biến mã độc

☐ Khác: ……

B. Liên kết lưu trữ mã nguồn của nhóm:

Mã nguồn của đề tài đồ án được lưu tại: https://github.com/Hjn4Pwn/ML-Detect-Malware-JPEG.git

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

[1] Ghi nội dung tương ứng theo mô tả

C. Tên bài báo tham khảo chính:

Cohen, A., Nissim, N. and Elovici, Y., 2020. MalJPEG: Machine learning based solution for the detection of malicious JPEG images. IEEE Access, 8, pp.19997-20011.

D. Dịch tên Tiếng Việt cho bài báo:

MalJPEG: Giải pháp dựa trên học máy để phát hiện hình ảnh JPEG độc hại

E. Tóm tắt nội dung chính:

Bài báo giới thiệu giải pháp sử dụng học máy để phát hiện các hình ảnh JPEG độc hại. Trong những năm gần đây, các cuộc tấn công mạng nhằm vào cá nhân, doanh nghiệp và tổ chức đã gia tăng. Tội phạm mạng thường sử dụng hình ảnh như một phương tiện để truyền tải mã độc, do tính phổ biến và thường được cho là an toàn của chúng. Trong đó JPEG là định dạng hình ảnh phổ biến nhất, điều này làm cho JPEG trở thành một đối tượng có thể bị lợi dụng để truyền tải mã độc hiệu quả.

MalJPEG là giải pháp đầu tiên sử dụng học máy để phát hiện một cách hiệu quả các hình ảnh JPEG độc hại chưa được xác định. MalJPEG trích xuất 10 đặc trưng đơn giản nhưng có ý nghĩa quan trọng trong việc xác định mã độc có được chèn vào hay không từ cấu trúc tệp JPEG và sử dụng máy học để phân biệt giữa hình ảnh lành tính và độc hại. Phương pháp này đã được đánh giá trên một tập hợp dữ liệu gồm 156,818 hình ảnh, bao gồm 155,013 hình ảnh an toàn và 1,805 hình ảnh độc hại.

Kết quả cho thấy khi sử dụng bộ phân loại LightGBM, MalJPEG đạt được hiệu suất phát hiện cao nhất với diện tích dưới đường cong ROC (AUC) là 0.997, tỷ lệ dương tính thật (TPR) là 0.951 và tỷ lệ dương tính giả (FPR) rất thấp là 0.004. Các thử nghiệm cũng cho thấy MalJPEG vượt trội hơn so với các

phương pháp trích xuất đặc trưng tổng quát khác và các phần mềm diệt virus hàng đầu.

MalJPEG không chỉ nhanh chóng mà còn có khả năng phát hiện hiệu quả cả các hình ảnh JPEG độc hại đã biết và chưa biết. Đây là một công cụ hữu ích để bảo vệ các tổ chức, dịch vụ đám mây và mạng xã hội khỏi các cuộc tấn công sử dụng hình ảnh JPEG độc hại.

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

Trích xuất đặc trưng từ tệp JPEG:

Vai trò: Các tác giả đã xác định và trích xuất 10 đặc trưng đơn giản nhưng phân biệt từ cấu trúc tệp JPEG. Các đặc trưng này giúp phân biệt giữa hình ảnh JPEG an toàn và độc hại mà không cần phải thực sự hiển thị hình ảnh (tức là phân tích tĩnh).

Đặc trưng được trích xuất bao gồm:

- Số lượng segments cụ thể trong tệp JPEG.
- Kích thước của các segments này.
- Sự hiện diện của dữ liệu sau segments kết thúc tệp (EOI).

Học máy:

Vai trò: Sử dụng các đặc trưng đã trích xuất, các tác giả đã áp dụng các bộ phân loại học máy để phân biệt giữa hình ảnh JPEG an toàn và độc hại.

Bộ phân loại sử dụng:

- Decision Tree
- Random Forest
- Gradient Boosting (XGBoost và LightGBM)

Đánh giá hiệu suất:

Vai trò: Để đánh giá hiệu suất của các bộ phân loại, các tác giả đã sử dụng các chỉ số như:

- Diện tích dưới đường cong ROC (AUC)
- Tỷ lệ dương tính thật (TPR)
- Tỷ lệ dương tính giả (FPR)

Các chỉ số này giúp xác định độ chính xác và hiệu quả của các bộ phân loại trong việc phát hiện hình ảnh độc hại.

So sánh với các phương pháp trích xuất đặc trưng tổng quát:

Vai trò: So sánh MalJPEG với các phương pháp trích xuất đặc trưng tổng quát như:

- Histogram (Histogram byte và Histogram byte entropy)
- Min-Hash (sử dụng shingle kích thước byte và char)

Mục đích của so sánh này là để chứng minh hiệu quả vượt trội của MalJPEG so với các phương pháp khác.

G. Môi trường thực nghiệm của bài báo:

- Cấu hình máy tính: Bài báo không đề cập cụ thể về thông số phần cứng của máy tính được sử dụng.
- Hệ điều hành: Ubuntu 18.04 LTS
- Các công cụ hỗ trợ sẵn có:
  - Giai đoạn trích xuất đặc trưng: Java được sử dụng để viết chương trình tạo tập dữ liệu và trích xuất đặc trưng từ các tệp JPEG.
  - Giai đoạn huấn luyện và đánh giá mô hình: Python sử dụng các thư viện Scikit-learn, XGBoost, và LightGBM để xây dựng và đánh giá các bộ phân loại học máy.

- Ngôn ngữ lập trình để hiện thực phương pháp:
  - Java: Được sử dụng để viết chương trình trích xuất đặc trưng từ các tệp JPEG và tạo tập dữ liệu.
  - Python: Được sử dụng để xây dựng và đánh giá các bộ phân loại học máy.
- Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có):
  - Tập dữ liệu bao gồm 156,818 hình ảnh JPEG, trong đó có 155,013 hình ảnh an toàn và 1,805 hình ảnh độc hại.
  - Hình ảnh an toàn được thu thập từ các mạng xã hội như Facebook, Instagram, WhatsApp.
  - Hình ảnh độc hại được thu thập từ VirusTotal, chỉ bao gồm các hình ảnh được ít nhất 5 công cụ diệt virus nhận diện là độc hại.
- Tiêu chí đánh giá tính hiệu quả của phương pháp:
  - True Positive Rate (TPR): Tỷ lệ phát hiện đúng các hình ảnh độc hại.
  - False Positive Rate (FPR): Tỷ lệ báo động giả đối với các hình ảnh lành tính
  - Area Under the ROC Curve (AUC): Diện tích dưới đường cong ROC, đánh giá tổng quan hiệu suất của bộ phân loại.
  - Integrated Detection Rate (IDR): Được tính bằng công thức $IDR=TPR\times(1-FPR)$

H. Kết quả thực nghiệm của bài báo:

Kết quả thực nghiệm:

So sánh với các phương pháp trích xuất đặc trưng tổng quát:

- MalJPEG sử dụng 10 đặc trưng đơn giản nhưng phân biệt từ cấu trúc tệp JPEG.

- Khi sử dụng bộ phân loại LightGBM, MalJPEG đạt được AUC = 0.997, TPR = 0.951, và FPR = 0.004.
- So sánh với các phương pháp trích xuất đặc trưng tổng quát như byte entropy histogram và Min-Hash, MalJPEG cho kết quả vượt trội cả về TPR và FPR.

Thử nghiệm thời gian trích xuất đặc trưng:

- MalJPEG có thời gian trích xuất đặc trưng nhanh nhất, trung bình là 24ms cho một tệp JPEG có kích thước từ 200KB đến 300KB.
- So sánh với các phương pháp khác, MalJPEG nhanh hơn đáng kể, đặc biệt là so với Min-Hash.

So sánh với các công cụ diệt virus hàng đầu:

- MalJPEG vượt trội hơn tất cả 12 công cụ diệt virus hàng đầu được thử nghiệm, với TPR của LightGBM là 0.951, trong khi công cụ diệt virus tốt nhất (Fortinet) chỉ đạt TPR = 0.823.
- Trung bình, các công cụ diệt virus đạt TPR = 0.73, thấp hơn đáng kể so với MalJPEG.

Thử nghiệm thực tế với các mẫu chưa biết:

- Trên năm tập dữ liệu thực tế khác nhau, LightGBM đạt AUC từ 0.975 đến 0.996, TPR từ 0.865 đến 0.911, và FPR từ 0.001 đến 0.09.
- Các kết quả này cho thấy MalJPEG có khả năng phát hiện hiệu quả các hình ảnh JPEG độc hại chưa biết trong tình huống thực tế.

Nhận xét về phương pháp MalJPEG:

Khả năng:

- Phương pháp MalJPEG hiệu quả cao trong việc phát hiện hình ảnh JPEG độc hại, với TPR cao và FPR thấp.

- MalJPEG có khả năng phát hiện cả các hình ảnh độc hại đã biết và chưa biết, vượt trội hơn so với các công cụ diệt virus hiện tại.

Ưu điểm:

- Hiệu suất cao: MalJPEG đạt AUC gần như tuyệt đối, chứng tỏ khả năng phân loại rất tốt.
- Nhanh chóng: Thời gian trích xuất đặc trưng nhanh, phù hợp cho các hệ thống yêu cầu thời gian thực.
- Đơn giản: Các đặc trưng trích xuất đơn giản nhưng hiệu quả, dễ dàng mở rộng và điều chỉnh khi cần thiết.

Nhược điểm:

- Tập trung vào JPEG: Phương pháp này chỉ áp dụng cho hình ảnh JPEG, không thể phát hiện hình ảnh độc hại ở các định dạng khác.
- Yêu cầu cập nhật: Với các kỹ thuật tấn công mới, MalJPEG có thể cần cập nhật các đặc trưng trích xuất để duy trì hiệu suất cao.
- Cần dữ liệu đại diện: Hiệu quả của phương pháp phụ thuộc vào tính đại diện của tập dữ liệu huấn luyện, cần cập nhật thường xuyên để phù hợp với các xu hướng tấn công mới.

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

- Trích xuất thành công đặc trưng từ file JPEG
- Kiểm thử thành công trên dữ liệu độc hại thật được báo cáo trên MalwareBazaar
- Chạy thực nghiệm so sánh 4 model: Decision Tree, Random Forest, XGBoost và LightGBM với kết quả tốt
- Đã build được thành công 1 API để tích hợp vào website bán hàng, thực hiện kiểm tra các file JPEG được upload từ user

- Thành công demo thêm mã độc hại JavaScript vào segment của file JPEG khi user click vào ảnh thì script sẽ được thực thi, hệ thống đã phân loại đúng được file này trước và sau khi thêm mã độc vào.

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

- Các dữ liệu ảnh JPEG có chứa mã độc thật không public trên internet, không tìm được dữ liệu ảnh thật chứa mã độc với số lượng lớn, chỉ tìm được vài ảnh có mã độc thật để demo, chạy thực nghiệm, dữ liệu train vẫn là file csv, này là điều bất khả kháng, kể cả các project của các anh khóa trước thì vẫn dùng file csv
- Quá nhiều bài tập, đồ án, labs của các môn, trong thực tế thì đây là năm 3 nên sinh viên đã có hướng đi riêng, ngoài việc cung cấp kiến thức tốt qua các deadlines thì cũng có bất cập là không phải hướng mình muốn theo nhưng vẫn phải dành nhiều thời gian để hoàn thành nó

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

95%

4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Đọc paper, tìm hiểu các khái niệm, thuật ngữ, papers khác có liên quan	Lưu Gia Huy, Ngô Thanh Sang, Đoàn Thị Ánh Dương
2	Thu thập dữ liệu	Lưu Gia Huy, Ngô Thanh Sang
3	Tìm hiểu cấu trúc file JPEG	Đoàn Thị Ánh Dương
4	Tìm hiểu và thực hiện trích xuất đặc trưng	Lưu Gia Huy, Ngô Thanh Sang

5	Tìm hiểu các models, thực hiện train, test với dữ liệu đã thu thập	Ngô Thanh Sang, Đoàn Thị Ánh Dương
6	Đánh giá, so sánh kết quả thu được	Đoàn Thị Ánh Dương
7	Thực hiện demo cách thức hoạt động cũng như phương pháp thêm mã độc vào file JPEG	Lưu Gia Huy, Ngô Thanh Sang
8	Build thành 1 API, tích hợp vào hệ thống website bán hàng để check các JPEG file khi người dùng upload	Lưu Gia Huy

BÁO CÁO TỔNG KẾT CHI TIẾT

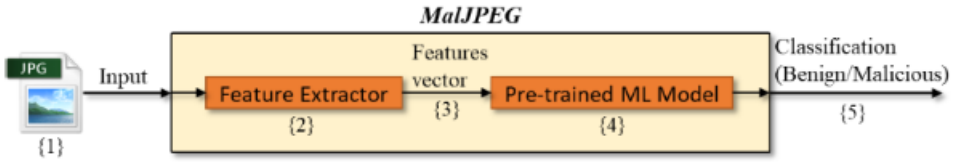
Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.

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

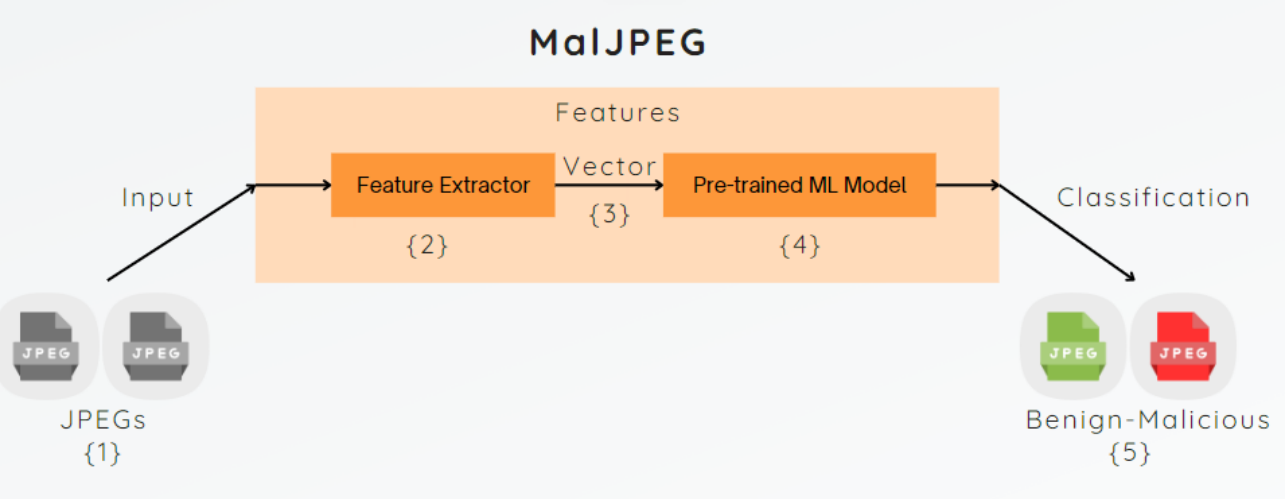
A. Phương pháp thực hiện

Kiến trúc, thành phần của hệ thống trong bài báo và kiến trúc và thành phần mà nhóm đã thực hiện là hoàn toàn giống nhau:

Mô hình triển khai của bài báo:



Tương tự với mô hình mà nhóm đã vẽ lại và triển khai:



Thành phần của hệ thống:

- Data Collection and Preprocessing:
  - Vai trò: Thu thập và xử lý tập dữ liệu JPEG để tạo tập huấn luyện và kiểm thử cho các mô hình học máy.
  - Chi tiết: Tập dữ liệu bao gồm hình ảnh an toàn từ các mạng xã hội và hình ảnh độc hại từ VirusTotal. Các tệp hình ảnh được xác minh và xử lý trước khi sử dụng để huấn luyện và kiểm thử mô hình.
- MalJPEG Feature Extractor:
  - Vai trò: Trích xuất các đặc trưng từ tệp JPEG. Đây là các đặc trưng đơn giản nhưng phân biệt, giúp phân biệt giữa hình ảnh an toàn và độc hại mà không cần phải thực sự hiển thị hình ảnh.
  - Chi tiết: Trình trích xuất đặc trưng của MalJPEG hoạt động trên cấu trúc tệp JPEG, xác định các đặc trưng dựa trên sự hiện diện và kích thước của các đoạn trong tệp.
- Machine Learning Classifier:
  - Vai trò: Phân loại hình ảnh JPEG dựa trên các đặc trưng đã trích xuất.
  - Chi tiết: Hệ thống sử dụng các bộ phân loại học máy như Decision Tree, Random Forest, Gradient Boosting (XGBoost và LightGBM). Các bộ phân loại

này được huấn luyện để nhận diện các hình ảnh độc hại dựa trên các đặc trưng được cung cấp bởi trình trích xuất đặc trưng.

Quy trình hoạt động của hệ thống:

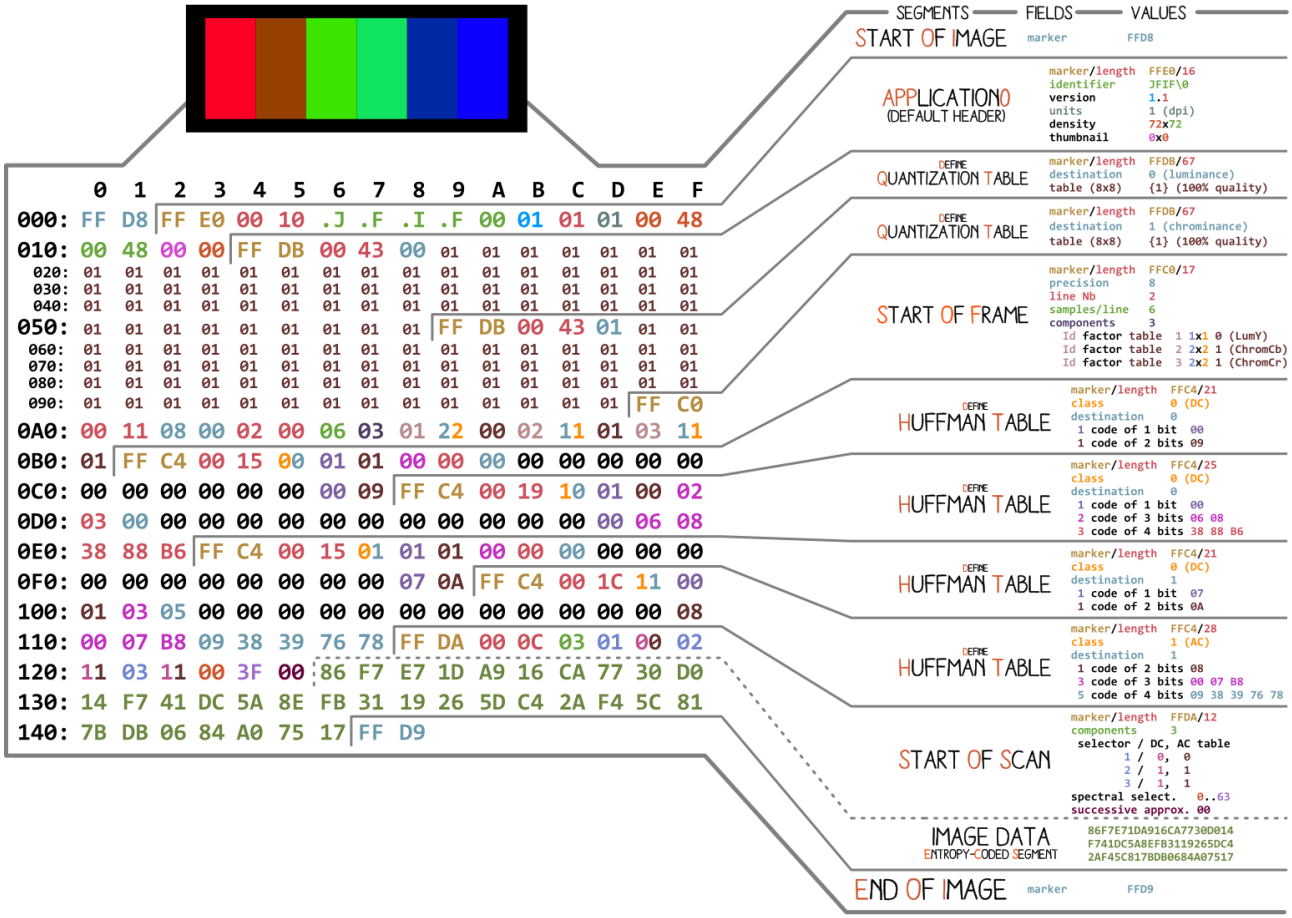
- Input JPEG Image: Hệ thống nhận đầu vào là một hình ảnh JPEG.
- Feature Extraction: Trình trích xuất đặc trưng của MalJPEG phân tích tệp JPEG và trích xuất các đặc trưng thành một vector đặc trưng.
- Classification:
  - Vector đặc trưng được đưa vào bộ phân loại học máy đã được huấn luyện trước đó.
  - Bộ phân loại sẽ quyết định xem hình ảnh là an toàn hay độc hại dựa trên các đặc trưng.
- Output: Hệ thống xuất ra kết quả phân loại: hình ảnh là lành tính hoặc độc hại.

B. Chi tiết cài đặt, hiện thực

1. Môi trường thực nghiệm

- Colab:
  - RAM: 13GB
  - Disk: 108GB
- Ngôn ngữ sử dụng: Python

2. Cấu trúc file JPEG



Các JPEG Marker: https://www.disktuna.com/list-of-jpeg-markers/

Marker Name	Hexadecimal Code	Definition/Purpose
APP$_n$	0xFFE0-0xFFEF	Reserved for **app**lication used
COM	0xFFFE	**Com**ment
DAC	0xFFCC	Define arithmetic conditioning table(s)
DHP	0xFFDE	Define hierarchical progression
DHT	0xFFC4	Define Huffman table(s)
DNL	0xFFDC	Define number of lines
DQT	0xFFDB	Define quantization table(s)
DRI	0xFFDD	Define restart interval
EXP	0xFFDF	**Exp**and reference image(s)
JPG	0xFFC8	Reserved for **JPEG** extensions
JPG$_n$	0xFFF0-0xFFFD	Reserved for **JPEG** extensions
RES	0xFF02-0xFFBF	**Res**erved
RST$_m$	0xFFD0-0xFFD7	**Res**tart with modulo 8 counter m
SOF$_n$	0xFFC0-3, 5-7, 9-B, D-F	Start of *Frame*
SOS	0xFFDA	Start of *Scan*
TEM	0xFF01	For **tem**porary use in arithmetic coding
SOI	0xFFD8	Start of image
EOI	0xFFD9	End of image

Tệp JPEG là tệp nhị phân bao gồm một chuỗi các đoạn (segment). Mỗi segment bắt đầu bằng 2 bytes, byte đầu tiên là 0xFF chỉ định bắt đầu một marker, byte tiếp theo xác định loại marker. Marker giúp chia tệp thành các đoạn khác nhau, chứa các nội dung với chức năng khác nhau.

Một số markers chính trong tệp JPEG:

- Marker SOI (Start of Image - 0xFFD8): Marker bắt đầu của tệp ảnh JPEG.
- Marker APPn (Application-specific markers - 0xFFE0 đến 0xFFEF): Chứa metadata hoặc thông tin mở rộng, ví dụ như EXIF trong APP1.
- Marker DQT (Define Quantization Table - 0xFFDB): Chứa bảng lượng tử hóa để nén dữ liệu ảnh.
- Marker DHT (Define Huffman Table - 0xFFC4): Chứa bảng mã Huffman cho quá trình nén dữ liệu.
- Marker SOF (Start of Frame - 0xFFC0 đến 0xFFCF): Chỉ định cấu trúc khung hình và thông số hình ảnh.
- Marker SOS (Start of Scan - 0xFFDA): Bắt đầu đoạn mã hóa dữ liệu hình ảnh thực tế.
- Marker EOI (End of Image - 0xFFD9): Marker kết thúc của tệp ảnh JPEG.
- Marker COM (Comment - 0xFFFE): Chứa các bình luận hoặc metadata bổ sung.

3. Trích xuất đặc trưng

Bài báo sẽ trích xuất 10 đặc trưng:

#	Feature Name	Description	Info Gain Rank
1	Marker_EOI_content _after_num	Number of bytes after the EOI (end of file) marker.	0.058
2	Marker_DHT_size_max	Maximal DHT marker size found in the file.	0.025
3	File_size	Image file size in bytes.	0.023
4	Marker_APP1_size_max	Maximal APP1 marker size found in the file.	0.023
5	Marker_COM_size_max	Maximal COM marker size found in the file.	0.017
6	Marker_DHT_num	Number of DHT markers found in the file.	0.016
7	File_markers_num	Total number of markers found in the file.	0.014
8	Marker_DQT_num	Number of DQT markers found in the file.	0.012
9	Marker_DQT_size_max	Maximal DQT marker size found in the file.	0.012
10	Marker_APP12_size_max	Maximal APP12 marker size found in the file.	0.011

Ý nghĩa của từng đặc trưng:

- **Số lượng marker DQT (Marker_DQT_num):** Marker DQT xác định cách lượng tử hóa các hệ số DCT trong JPEG. Một số lượng lớn các marker DQT có thể cho thấy sự chèn dữ liệu độc hại trong các bảng này.
- **Số lượng marker DHT (Marker_DHT_num):** Marker DHT chứa bảng mã Huffman cho quá trình nén dữ liệu. Một số lượng lớn các marker DHT có thể chỉ ra rằng hình ảnh chứa mã độc cố gắng che giấu hoặc mã hóa dữ liệu độc hại.
- **Số lượng marker APPn (Marker_APPn_num):** Marker APPn thường được sử dụng để lưu trữ thông tin bổ sung hoặc siêu dữ liệu. Một số lượng lớn các marker APPn có thể chỉ ra sự hiện diện của dữ liệu không bình thường, có thể là mã độc.
- **Kích thước tối đa của các marker DQT (Max_marker_size_DQT):** Kích thước lớn bất thường của các marker DQT có thể là dấu hiệu của việc nhúng dữ liệu độc hại vào bảng lượng tử hóa.
- **Kích thước tối đa của các marker DHT (Max_marker_size_DHT):** Kích thước lớn bất thường của các marker DHT có thể cho thấy mã độc được lưu trữ trong các bảng mã Huffman.

- **Kích thước tối đa của các marker COM (Max_marker_size_COM):** Marker COM chứa các bình luận hoặc siêu dữ liệu. Kích thước lớn bất thường của marker COM có thể là dấu hiệu của dữ liệu độc hại.
- **Kích thước tối đa của các marker APP1 (Max_marker_size_APP1):** Marker APP1 thường chứa siêu dữ liệu như EXIF trong các ảnh chụp bằng máy ảnh kỹ thuật số. Kích thước lớn bất thường của marker này có thể là dấu hiệu của mã độc.
- **Kích thước tối đa của các marker APP12 (Max_marker_size_APP12):** Marker APP12 cũng thường chứa siêu dữ liệu. Kích thước lớn bất thường có thể chỉ ra mã độc.
- **Tổng số lượng marker (File_marker_num):** Tổng số lượng marker trong tệp JPEG cho biết mức độ phức tạp của tệp. Một số lượng marker cao bất thường có thể là dấu hiệu của sự can thiệp độc hại.
- **Kích thước của tệp JPEG (File_size):** Kích thước tệp JPEG có thể giúp xác định sự khác thường trong cấu trúc tệp. Tệp JPEG với kích thước lớn bất thường có thể chứa mã độc.

Code thực hiện:

```python
import os
import sys
import argparse
import pandas as pd
from tqdm import tqdm
from struct import unpack

marker_mapping = {
    0xFFC0: "SOF0",
    0xFFC1: "SOF1",
    0xFFC2: "SOF2",
    0xFFC3: "SOF3",
    0xFFC4: "DHT",
    0xFFC5: "SOF5",
    0xFFC6: "SOF6",
    0xFFC7: "SOF7",
    0xFFC8: "JPG",
    0xFFC9: "SOF9",
    0xFFCA: "SOF10",
    0xFFCB: "SOF11",
    0xFFCC: "DAC",
    0xFFCD: "SOF13",
    0xFFCE: "SOF14",
    0xFFCF: "SOF15",
    0xFFD0: "RST0",
    0xFFD1: "RST1",
    0xFFD2: "RST2",
    0xFFD3: "RST3",
    0xFFD4: "RST4",
    0xFFD5: "RST5",
    0xFFD6: "RST6",
    0xFFD7: "RST7",
    0xFFD8: "SOI",
    0xFFD9: "EOI",
    0xFFDA: "SOS",
    0xFFDB: "DQT",
    0xFFDC: "DNL",
    0xFFDD: "DRI",
    0xFFDE: "DHP",
    0xFFDF: "EXP",
    0xFFE0: "APP0",
    0xFFE1: "APP1",
```

```
    0xFFE7: "APP7",
    0xFFE8: "APP8",
    0xFFE9: "APP9",
    0xFFEA: "APP10",
    0xFFEB: "APP11",
    0xFFEC: "APP12",
    0xFFED: "APP13",
    0xFFEE: "APP14",
    0xFFEF: "APP15",
    0xFFF0: "JPG0",
    0xFFF1: "JPG1",
    0xFFF2: "JPG2",
    0xFFF3: "JPG3",
    0xFFF4: "JPG4",
    0xFFF5: "JPG5",
    0xFFF6: "JPG6",
    0xFFF7: "JPG7",
    0xFFF8: "JPG8",
    0xFFF9: "JPG9",
    0xFFFA: "JPG10",
    0xFFFB: "JPG11",
    0xFFFC: "JPG12",
    0xFFFD: "JPG13",
    0xFFFE: "COM",
    0xFF01: "TEM",
}

feature_name = [
    [
        "marker_EOI_content_after_num",
        "marker_DQT_num",
        "marker_DHT_num",
        "file_markers_num",
        "marker_DQT_size_max",
        "marker_DHT_size_max",
        "file_size",
        "marker_COM_size_max",
        "marker_APP1_size_max",
        "marker_APP12_size_max",
    ]
]
```

```
class JPEG:
    def __init__(self, image_file):
        self.path = image_file
        with open(image_file, "rb") as f:
            self.img_data = f.read()

    def decode(self):
        data = self.img_data
        marker_DQT_num = 0
        marker_DQT_size_max = 0
        marker_DHT_num = 0
        marker_DHT_size_max = 0
        file_markers_num = 0
        marker_EOI_content_after_num = 0
        marker_APP12_size_max = 0
        marker_APP1_size_max = 0
        marker_COM_size_max = 0
        file_size = len(data)
        # print(f"file_size = {file_size}")
        while True:
            try:
                (marker,) = unpack(">H", data[0:2])
            except Exception as e:
                pass
            marker_map = marker_mapping.get(marker)
            if marker_map != None:
                file_markers_num += 1
                if marker_map == "DQT":
                    marker_DQT_num += 1
                    (lenchunk,) = unpack(">H", data[2:4])
                    if lenchunk > marker_DQT_size_max:
                        marker_DQT_size_max = lenchunk
                    data = data[2 + lenchunk :]
                elif marker_map == "SOI":
                    data = data[2:]
                elif marker_map == "DHT":
                    marker_DHT_num += 1
                    (lenchunk,) = unpack(">H", data[2:4])
                    if lenchunk > marker_DHT_size_max:
                        marker_DHT_size_max = lenchunk
                    data = data[2 + lenchunk :]
                elif marker_map == "EOI":
                    rem = data[2:]
                    if len(rem) > marker_EOI_content_after_num:
```

```
                    marker_EOI_content_after_num = len(rem)
                data = rem
            elif marker_map == "SOS":
                data = data[-2:]
            elif marker_map == "APP12":
                (lenchunk,) = unpack(">H", data[2:4])
                if lenchunk > marker_APP12_size_max:
                    marker_APP12_size_max = lenchunk
                data = data[2 + lenchunk :]
            elif marker_map == "APP1":
                (lenchunk,) = unpack(">H", data[2:4])
                if lenchunk > marker_APP1_size_max:
                    marker_APP1_size_max = lenchunk
                data = data[2 + lenchunk :]
            elif marker_map == "COM":
                (lenchunk,) = unpack(">H", data[2:4])
                if lenchunk > marker_COM_size_max:
                    marker_COM_size_max = lenchunk
                data = data[2 + lenchunk :]
            elif marker_map == "TEM":
                data = data[2:]
            elif marker <= 0xFFD9 and marker >= 0xFFD0:
                data = data[2:]
            elif marker <= 0xFFBF and marker >= 0xFF02:
                (lenchunk,) = unpack(">H", data[2:4])
                data = data[2 + lenchunk :]
            else:
                (lenchunk,) = unpack(">H", data[2:4])
                data = data[2 + lenchunk :]
        else:
            data = data[1:]
        if len(data) == 0:
            data_list = [
                marker_EOI_content_after_num,
                marker_DQT_num,
                marker_DHT_num,
                file_markers_num,
                marker_DQT_size_max,
                marker_DHT_size_max,
                file_size,
                marker_COM_size_max,
                marker_APP1_size_max,
                marker_APP12_size_max,
            ]
            return data_list
```

4. Dataset

- Benign: 3012 samples
- Malicious: 674 samples
- 7:3 - train:test (2580:1106)

```python
import pandas as pd
from sklearn.model_selection import train_test_split

def prepare_datasets(benign_path, malicious_path, out_train, out_test):
    print("Preparing dataset...")
    print(f"Output files: {out_train}, {out_test}")
    print(f"Path to the benign dataset: {benign_path}")
    print(f"Path to the malicious dataset: {malicious_path}")

    benign = pd.read_csv(benign_path)
    malicious = pd.read_csv(malicious_path)

    label = [0] * benign.shape[0] + [1] * malicious.shape[0]

    data = pd.concat([benign, malicious])

    X_train, X_test, y_train, y_test = train_test_split(
        data, label, test_size=0.3, stratify=label
    )

    X_train["label"] = y_train
    X_test["label"] = y_test

    X_train.to_csv(out_train, index=False)
    X_test.to_csv(out_test, index=False)

    print(f"Training dataset samples: {X_train.shape[0]}")
    print(f"Testing dataset samples: {X_test.shape[0]}")

benign_path = 'data/benign.csv'
malicious_path = 'data/malicious.csv'
out_train = 'data/train.csv'
out_test = 'data/test.csv'

prepare_datasets(benign_path, malicious_path, out_train, out_test)
```

```
Preparing dataset...
Output files: data/train.csv, data/test.csv
Path to the benign dataset: data/benign.csv
Path to the malicious dataset: data/malicious.csv
Training dataset samples: 2580
Testing dataset samples: 1106
```

5. Mô hình học máy

- Decision Tree
- Random Forest
- XGBoost
- LightGBM

```python
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split
import xgboost as xgb
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier


clf_xgb = xgb.XGBClassifier()
# clf_lgbm = LGBMClassifier(verbose=-1, force_row_wise=True)
clf_lgbm = LGBMClassifier(
    verbose=-1,
    force_row_wise=True,
    num_leaves=100,
    max_depth=-1,
    learning_rate=0.01,
    n_estimators=500,
    boosting_type='gbdt',
    objective='binary',
    subsample=0.7,
    colsample_bytree=0.7,
    reg_alpha=0.05,
    reg_lambda=0.05,
    min_child_weight=0.5
)
clf_rf = RandomForestClassifier()
clf_dt = DecisionTreeClassifier()
```

C. Kết quả thực nghiệm

Thực nghiệm trên JPEG chứa malware thành công: Lấy hình ảnh JPEG có chứa malware thực từ **MalwareBazaar** và thực hiện kiểm tra:

MALWARE bazaar by ABUSE.ch

Browse | Upload | Hunting | API | Export | Statistics | FAQ | About | Login

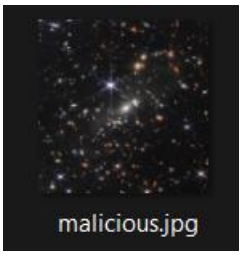
The table below shows all malware samples that are associated with this particulare tag (**max 400**).

Show 50 entries Search:

Firstseen (UTC)	SHA256 hash	Tags	Signature	Reporter
2022-09-03 07:52:39	3bdf6d9f0f35be75d8345...	crt jpeg sansisc	n/a	DSTLabs
2020-05-21 05:57:07	c210625165f0448f38cf69...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:56:55	06750ac0faac65082e722...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:56:28	9809e7b9ac68c6e45309...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:56:11	283315ac54955f11b9e9b...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:56:00	0fa3607af6cb9101d40fd...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:55:48	26b47dc955792b7529da...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:55:37	1dccde0289f175d1d4114...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:55:25	dadcf3ef1ba5f212ec2a93...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:55:13	4b200dd8310e34c8b7c5...	GuLoader jpeg	GuLoader	cocaman
2020-05-21 05:53:32	17e0971dd39c6cede948...	GuLoader jpeg	GuLoader	cocaman
2020-04-28 05:59:41	a1e42b04d0b7c1e14ee2...	jpeg	AgentTesla	cocaman

Showing 1 to 12 of 12 entries Previous 1 Next

© abuse.ch 2024




```python
import pickle
import numpy as np

classifier = pickle.load(open("models/lgbm.pkl", "rb"))

image = JPEG("test/malicious.jpg")
data = image.decode()

data = np.reshape(data, (1, -1))


pred = classifier.predict(data)

if pred == 1:
    print("This is malicious jpeg.")
else:
    print("This is benign image.")
```

```
This is malicious jpeg.
```

Các thông số đánh giá mô hình:

	Accuracy	Precision	Recall	F1 Score	AUC
XGBoost	99.37	98.51	98.02	98.26	98.84
LightGBM	99.54	99.50	98.02	98.75	98.95
Random forest	99.37	98.02	98.51	98.27	99.04
Decision Tree	99.19	97.54	98.02	97.78	98.73

Nhận xét:

- Do dataset khá nhỏ nên kết quả của các model không chênh lệch nhiều và khá cao
- LGBM vẫn có được độ chính xác cao nhất tương tự như kết quả của bài báo

D. Hướng phát triển

Hướng phát triển tiềm năng của đề tài này trong tương lai

1. Mở rộng bộ đặc trưng:

- Phát hiện steganography: Mở rộng khả năng của MalJPEG để không chỉ phát hiện mã độc mà còn phát hiện các kỹ thuật steganography sử dụng để giấu thông tin trong ảnh JPEG.
- Không chỉ JPEG mà còn có thể phát hiện mã độc trong các định dạng ảnh khác như PNG, Webp,...

2. Cải thiện mô hình máy học:

- Tăng cường học sâu (Deep Learning): Áp dụng các mô hình học sâu để cải thiện độ chính xác của việc phát hiện mã độc. Các mô hình này có thể tự động học và phát hiện các mẫu phức tạp mà các thuật toán học máy truyền thống có thể bỏ qua.

- Học tập liên tục (Continual Learning): Phát triển các mô hình có khả năng học từ dữ liệu mới một cách liên tục mà không quên đi kiến thức đã học trước đó. Điều này giúp mô hình luôn cập nhật và hiệu quả trước các mối đe dọa mới.

3. Triển khai và ứng dụng thực tế:

- Ứng dụng trong các hệ thống bảo mật: Tích hợp MalJPEG vào các hệ thống bảo mật hiện tại, chẳng hạn như các giải pháp antivirus, để cải thiện khả năng phát hiện mã độc trong các tệp JPEG.
- Bảo vệ nền tảng mạng xã hội và dịch vụ đám mây: Sử dụng MalJPEG để bảo vệ các nền tảng mạng xã hội (Facebook, Instagram, WhatsApp, v.v.) và các dịch vụ đám mây (Google Drive, Dropbox, v.v.), hay các ứng dụng web khác, nơi mà người dùng có quyền upload hình ảnh, khỏi các tệp JPEG độc hại.
- Ứng dụng trên thiết bị di động: Phát triển phiên bản nhẹ của MalJPEG để sử dụng trên các thiết bị di động, giúp bảo vệ người dùng khỏi các mối đe dọa đến từ các tệp JPEG độc hại được tải về hoặc chia sẻ qua các ứng dụng di động.

Nhận xét về tính ứng dụng của đề tài

Tính ứng dụng cao:

- Phát hiện mã độc hiệu quả: MalJPEG đã chứng minh khả năng phát hiện mã độc trong các tệp JPEG với độ chính xác cao, đạt được AUC = 0.997 và tỷ lệ phát hiện dương tính thực sự (TPR) là 0.951.
- Hiệu quả trong thời gian thực: MalJPEG hoạt động nhanh chóng, đáp ứng yêu cầu thời gian thực trong việc xử lý lượng lớn tệp hình ảnh. Điều này làm cho nó phù hợp để triển khai trong các hệ thống bảo mật lớn và phức tạp.
- Khả năng mở rộng và tích hợp: MalJPEG có thể dễ dàng tích hợp vào các hệ thống bảo mật hiện tại và mở rộng để bao gồm thêm các đặc trưng mới và các phương pháp phát hiện nâng cao.
- Có thể tích hợp vào các ứng dụng web, nơi mà người dùng có thể upload file JPEG lên server

Khó khăn và thách thức:

- Phát triển liên tục: Cần duy trì và cập nhật mô hình liên tục để đối phó với các mối đe dọa mới và các kỹ thuật tấn công ngày càng tinh vi.
- Độ phức tạp của các tệp JPEG: Sự đa dạng và phức tạp của các tệp JPEG trong thực tế có thể đòi hỏi phải phát triển và tinh chỉnh các đặc trưng và mô hình liên tục để duy trì hiệu quả.
- Thiếu hụt dữ liệu thực tế, các ảnh JPEG có chứa malware cần cho model học thì lại khan hiếm

Sinh viên báo cáo các nội dung mà nhóm đã thực hiện, có thể là 1 phần hoặc toàn bộ nội dung của bài báo. Nếu nội dung thực hiện có khác biệt với bài báo (như cấu hình, tập dữ liệu, kết quả,...), sinh viên cần chỉ rõ thêm khác biệt đó và nguyên nhân.

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

YÊU CẦU CHUNG

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

Báo cáo:

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

  Ví dụ: [NT521.N11.ANTT]-Project_Final_Nhom03_CK01.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

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

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

HẾT