ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Thanh Huyền

PHÁT TRIỂN CÔNG CỤ TRÍCH XUẤT VÀ ĐÁNH GIÁ ĐẶC TRƯNG PHỤC VỤ PHÁT HIỆN MÃ ĐỘC TRONG TÀI LIỆU PDF

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Công nghệ thông tin

Hà Nội, 2022

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Thanh Huyền

PHÁT TRIỂN CÔNG CỤ TRÍCH XUẤT VÀ ĐÁNH GIÁ ĐẶC TRƯNG PHỤC VỤ PHÁT HIỆN MÃ ĐỘC TRONG TÀI LIỆU PDF

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: TS. Lê Đình Thanh

Hà Nội, 2022

Tóm tắt

Trong những năm gần đây, PDF hay viết tắt của Portable Document Format, là một định dạng lưu trữ tài liệu được sử dụng rộng rãi trên các nền tảng Internet. Tính linh hoạt trong cấu trúc và khả năng cho phép nhúng những mã thực thi như JavaScript, shellcode của PDF đã thúc đẩy tin tặc sử dụng tài liệu PDF như một phương tiện tấn công mã độc hiệu quả. Do đó việc phát hiện các tệp PDF độc hại là rất quan trọng trong bảo mật thông tin.

Hiện nay, phương pháp được sử dụng phổ biến trong phát hiện tài liệu PDF độc hại là trích xuất đặc trưng và sử dụng các mô hình phân loại học máy. Hiệu quả của các mô hình phân loại phụ thuộc rất lớn vào các tập đặc trưng. Vậy nên mục tiêu của khóa luận là tìm ra những tập đặc trưng tốt nhất để có thể xây dựng được một mô hình phát hiện tệp PDF độc hại hiệu quả.

Các đặc trưng có thể được phân thành 3 nhóm, bao gồm các đặc trưng thống kê, đặc trưng cấu trúc và đặc trưng JavaScript. Trong các nghiên cứu trước đây thường chỉ tập trung vào một nhóm đặc trưng cụ thể, ngược lại, khóa luận này sẽ tiến hành kết hợp cả 3 nhóm. Đầu tiên các đặc trưng cấu trúc và thống kê được kết hợp với nhau, xếp hạng và lựa chọn để tạo ra một mô hình phân loại hiệu quả hơn các mô hình đã có chỉ sử dụng 1 nhóm đặc trưng. Tiếp theo, đối với đặc trưng thứ ba, bởi mã JavaScript thường được làm rối khiến cho các mô hình phân lớp khó có thể phân biệt được mã lành tính với mã độc hại, nên khóa luận đưa ra một cách xử lý là chia quá trình phát hiện thành 2 giai đoạn. Giai đoạn 1 sử dụng mô hình phân loại chỉ sử dụng hai nhóm đặc trưng đầu với mục tiêu sàng lọc những tệp PDF độc không chứa JavaScript hoặc chứa JavaScript dễ phát hiện. Giai đoạn 2, với những tệp khó phát hiện, mã JavScript được phân tích và xử lý bằng những kỹ thuật chuyên sâu nhưng tốn kém hơn. Theo phương pháp xử lý này, khóa luận đã điều chỉnh mô hình phân lớp ở giai đoạn một để đảm bảo không cho ra dương tính giả. Thuật toán được sử dụng là thuật toán lặp Zero False Positive, được đảm bảo tính đúng đắn thông qua chứng minh và thử nghiệm trên tập huấn luyện.

Lời cảm ơn

Đầu tiên tôi xin được gửi lời cảm ơn chân thành tới thầy Lê Đình Thanh đã chỉ dẫn tận tình và đóng góp ý kiến giúp tôi có thể hoàn thành khóa luận tốt nghiệp một cách trọn vẹn nhất.

Tôi cũng xin bày tỏ lòng biết ơn tới các thầy cô cùng Khoa Công Nghệ Thông Tin - Trường Đại học Công Nghệ ĐHQG Hà Nội truyền thụ cho tôi những kiến thức quan trọng, đồng thời tạo điều kiện cho tôi được nghiên cứu và thực hiện đề tài tốt nghiệp này.

Lời cam đoan

Tôi xin cam đoan rằng khóa luận "Phát triển công cụ trích xuất và đánh giá đặc trưng phục vụ phát hiện mã độc trong tài liệu PDF" này không sao chép từ bất kỳ ai, tổ chức nào khác. Toàn bộ nội dung trong tài liệu là cá nhân tôi qua quá trình học tập và nghiên cứu mà hoàn thiện. Mọi tài liệu tham khảo đều được trích dẫn hợp pháp. Nếu lời cam đoan là sai sự thật thì tôi xin chịu mọi trách nhiệm và hình thức kỷ luật theo quy định từ phía nhà trường.

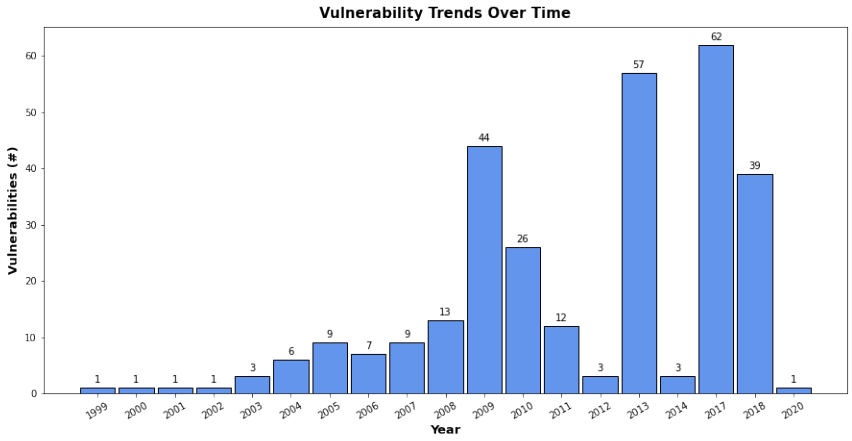
Mục lục

Danh sách hình vẽ

Danh sách bảng

Mở đầu

Những kẻ tấn công bằng nhiều cách có thể cài đặt các phần mềm độc hại trên máy nạn nhân để có được quyền truy cập, thực hiện các hành vi xâm phạm dữ liệu cá nhân hay dữ liệu nhạy cảm, phá hoại hệ thống hoặc sử dụng vào những mục đích xấu khác. Người dùng thường có thể dễ dàng nhận ra mối đe dọa từ những tệp thực thi, ngoài ra nhận thức về mối nguy hiểm từ các tài liệu Microsoft Office cũng ngày càng tăng cao. Tuy nhiên, bỏ qua sự phức tạp về định dạng cấu trúc tệp PDF, người dùng có xu hướng coi các tệp PDF là tài liệu vô hại. Hiện nay, lợi dụng các lỗ hổng bảo mật, các trình đọc tài liệu PDF đang là mục tiêu của các kẻ tấn công mạng. Điển hình với trình đọc PDF Acrobat Reader, từ năm 1999 tới nay, đã có tới 298 lỗ hổng được phát hiện theo thống kê của CVE Details [4](Hình 1).



Hình 1: Biểu đồ các lỗ hổng của trình đọc PDF Acrobat Reader theo các năm[5]

Trong khóa luận này, tôi sẽ trình bày một số phương pháp để phát hiện các tệp PDF độc hại, trong đó các đặc trưng được trích xuất từ tài liệu PDF

[5]`https://www.cvedetails.com`

sẽ được sử dụng để xây dựng bộ phân loại học máy tự động phát hiện tài liệu PDF độc hại, thông qua các chương sau:

Chương 1 Kiến thức nền tảng về định dạng tài liệu PDF, một số thuật toán học máy, từ đó đưa ra một quy trình để phát hiện phần mềm độc hại.

Chương 2 Thực hiện trích xuất đặc trưng trên tập dữ liệu gồm khoảng 22000 tệp PDF, từ đó xây dựng bộ phân loại học máy và đánh giá kết quả khi kết hợp các đặc trưng khác nhau.

Chương 3 Đề xuất phương hướng phát hiện tài liệu PDF độc hại thông qua hai giai đoạn, với giai đoạn đầu xây dựng một thuật toán học máy không gây phân loại dương sai và giai đoạn hai gồm các phương hướng xử lý nâng cao với các tệp được gán nhãn sạch.

Kết luận Đánh giá kết quả đạt được, đưa ra những hạn chế và khó khăn gặp phải.

Chương 1

Kiến thức nền tảng

Chương này gồm ba kiến thức nền tảng. Đầu tiên trình bày sơ bộ về các đối tượng và cách cấu trúc một tài liệu PDF, từ đó là bước đệm để hiểu cách phần mềm độc hại được nhúng vào tệp cũng như là cách để có thể xác định các bất thường trong cấu trúc một tệp PDF. Tiếp theo giới thiệu về một số thuật toán học máy được sử dụng khi xây dựng mô hình phân loại tự động các tệp PDF độc hại. Phần cuối cùng đưa ra cách tiếp cận và quy trình để có thể phát hiện tệp độc hại.

1.1 Định dạng tệp PDF

Tệp pdf là một trong những định dạng phổ biến nhất để lưu trữ tài liệu trên toàn thế giới. Năm 1993, PDF được Adobe Systems định nghĩa và đến năm 2008, PDF đã trở thành một tiêu chuẩn mở được phát hành dưới chuẩn ISO 32000-1.

1.1.1 Các đối tượng tài liệu

Cấu trúc của tệp PDF bao gồm các đối tượng liên kết với nhau trong một cấu trúc cây. Những đối tượng này được chia thành ba loại, gồm có các đối tượng đơn giản, các đối tượng ghép và đối tượng độc lập.

Đối tượng đơn giản

- Boolean: lưu giá trị true/false.
- Number: là đối tượng số.
- String: là các xâu ký tự với kích thước tối đa 65535 bytes.

- Name: là các đối tượng từ khóa được định nghĩa sẵn trong tiêu chuẩn PDF, theo sau bởi ký tự '/', ví dụ '/kids' xác định các đối tượng con, '/filter' chỉ định bộ lọc được sử dụng,...
- Indirect reference: là các tham chiếu tới các đối tượng độc lập.

Đối tượng ghép

- Array: mảng 1 chiều các đối tượng được sắp xếp trong cặp ký tự '[]'.
- Dictionary: giống như một danh sách lưu trữ các cặp từ khóa - giá trị, được đặt trong cặp ký tự '«»'. Từ khóa phải là một đối tượng Name còn giá trị thì có thể là bất kể đối tượng nào. Mỗi từ khóa chỉ xuất hiện 1 lần duy nhất trong một đối tượng dictionary.
- Stream: đối tượng stream giống như một đối tượng string, gồm một chuỗi các bytes, nhưng khác ở việc không giới hạn về độ dài. Do đó đối tượng stream có khả năng chứa một lượng lớn dữ liệu như hình ảnh, các thông tin mô tả trang,... Stream được đặt trong cặp từ khóa 'stream endstream'.

Đối tượng độc lập

Các đối tượng độc lập có định danh là các cặp (number object, generation object) ứng với số hiệu đối tượng và thế hệ của đối tượng. Các đối tượng khác có thể tham chiếu tới qua định danh này. Các đối tượng độc lập được định nghĩa bên trong cặp từ khóa 'obj endobj'.

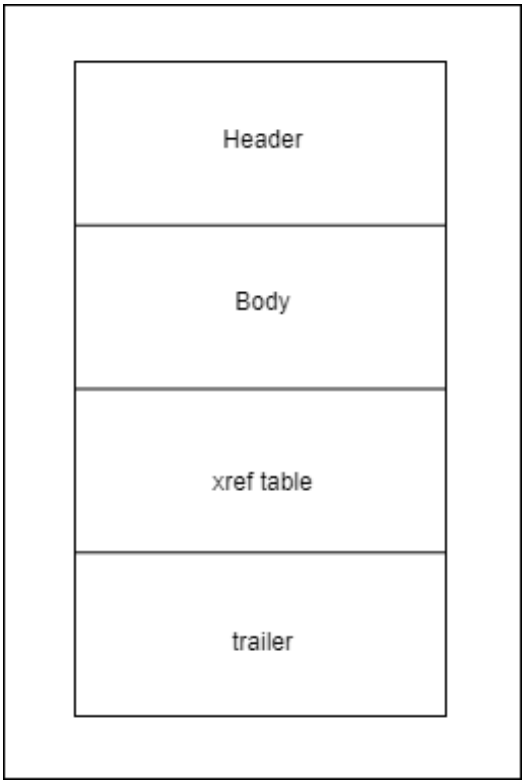
1.1.2 Cấu trúc vật lý

Cấu trúc vật lý của PDF xác định cách đối tượng được lưu trữ trong tệp, cách mà chúng được truy cập và cập nhật. Một tệp PDF cơ bản sẽ được xây dựng bằng bốn yếu tố: header, body, bảng tham chiếu xref table và trailer. Hình 1.1 thể hiện cấu trúc này.

Cấu trúc ban đầu của PDF sẽ được thay đổi sau mỗi lần cập nhật, bằng cách thêm các sự thay đổi vào cuối của tệp, được gọi là cập nhật tăng dần.

Header

Header là dòng đầu tiên trong tệp PDF, xác định phiên bản tệp, được viết dưới định dạng '%PDF-version'. Ví dụ '%PDF-1.7' xác định phiên bản PDF hiện tại là 1.7.



Hình 1.1: Cấu trúc vật lý của tệp PDF

Ngoài ra, nếu tệp PDF chứa dữ liệu nhị phân, header sẽ chứa một dòng comment gồm ít nhất 4 ký tự nhị phân giúp xác định được cần xử lý dữ liệu tệp dưới dạng văn bản hay nhị phân.

Body

Phần body của tệp chứa tất cả các đối tượng lưu trữ những dữ liệu sẽ hiển thị cho người dùng, được xác định từ sau phần header đến khi xuất hiện từ khóa 'xref'. Những dữ liệu trong body sau đó có thể được chỉnh sửa, và phần cập nhật sẽ được lưu trữ ở cuối tệp.

Xref Table

Xref table hay còn được gọi là cross-reference table là một bảng tham chiếu tới các đối tượng độc lập trong tài liệu nhằm mục đích cho phép truy cập ngẫu nhiên vào các đối tượng này mà không cần đọc toàn bộ tài liệu để định vị một đối tượng cụ thể.

Bảng có thể chứa một hoặc nhiều phần. Ban đầu, toàn bộ bảng chỉ bao

gồm một phần duy nhất. Một phần tương ứng sẽ được thêm vào nếu tệp được thay đổi.

```
xref
0 1
0000000023 65535 f
3 1
0000025324 00000 n
21 4
0000025518 00002 n
0000025632 00000 n
0000000024 00001 f
0000000000 00001 f
36 1
0000026900 00000 n
```

Hình 1.2: Một bảng xref trong tài liệu PDF

Ở trên hình 1.2, cho thấy một ví dụ về bảng tham chiếu trong tệp PDF, gồm có 4 phần con, bắt đầu bởi dòng chứa hai giá trị số với giá trị đầu là số thứ tự của đối tượng đầu tiên, giá trị thứ hai là số lượng đối tượng trong phần con này. Ví dụ ở phần thứ 3, giá trị '21 4' cho thấy có 4 đối tượng trong phần con, được đánh số bắt đầu từ 21 đến 24. Theo sau đó là các dòng đại diện cho các đối tượng, xác định bởi một chuỗi gồm 20 byte lưu trữ địa chỉ của đối tượng (offset), số thế hệ của đối tượng và một cờ 'f' hoặc 'n. Cờ 'f' nghĩa là đối tượng vẫn tồn tại trong tệp nhưng được đánh dấu không nên được sử dụng, ngược lại, cờ 'n' xác định đối tượng đang được sử dụng.

Trailer

Trailer PDF được lưu ở cuối tệp, chỉ định vị trí của bảng tham chiếu và các đối tượng đặc biệt khác, bắt đầu bởi từ khóa 'trailer' (Hình 1.3).

Trong đối tượng dictionary của trailer lưu trữ các giá trị:

- Size: xác định số lượng đối tượng trong bảng tham chiếu (có đếm cả các đối tượng trong các phần mới cập nhật). Giá trị này phải là 1 số nguyên, và không thể là một đối tượng tham chiếu.

```
trailer
<<
/Size 22                    % số đối tượng
/Root 2 0 R
>>
startxref
24212                       % offset của trailer
%%EOF
```

Hình 1.3: Một trailer trong tài liệu PDF

- Prev: xác định địa chỉ của bảng tham chiếu được cập nhật trước đó- được sử dụng khi có nhiều bảng tham chiếu.
- Root: tham chiếu tới một đối tượng gốc trong mô hình cấu trúc logic của PDF.

Cập nhật tăng dần

PDF được thiết kế với khả năng cập nhật tăng dần, tức là có thể cập nhật bằng cách thêm các đối tượng thay đổi vào cuối tệp PDF mà không cần phải viết lại toàn bộ tệp, mục tiêu cập nhật nhanh chóng các thay đổi nhỏ vào trong 1 tệp lớn.

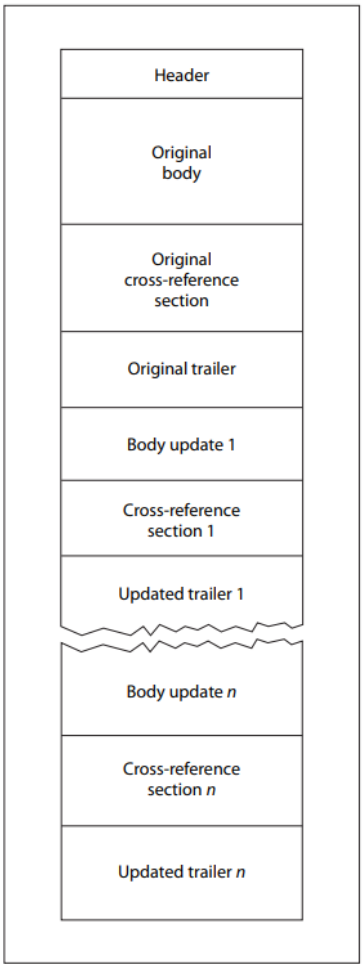
Có thể thấy trong hình 1.4, tệp PDF vẫn chứa các phần header, body, xref table và trailer ban đầu. Tiếp theo sẽ có các body, xref table và trailer được thêm vào cuối tệp. Các bảng tham chiếu được bổ sung sẽ chỉ chứa các đối tượng đã bị thay đổi, thay thế hoặc xóa. Các đối tượng bị xóa sẽ vẫn nằm trong tệp nhưng được đánh dấu với cờ 'f'. Trong phần trailer được thêm mới sẽ có một cặp từ khóa '\Prev' cho biết vị trí của bảng tham chiếu trước đó và cần phải kết thúc bằng ký tự kết thúc "%%EOF"

1.1.3 Cấu trúc logic

Cấu trúc logic của tài liệu PDF chỉ ra cách các đối tượng liên kết với nhau và được tổ chức sao cho các trình đọc PDF có thể hiểu và xử lý tài liệu một cách thích hợp. Các đối tượng này được xây dựng dựa trên một mô hình phân cấp dạng cây, với nút gốc là một đối tượng catalog (Hình 1.5).

Catalog cung cấp thông tin chung về loại đối tượng nào sẽ được hiển thị trong tệp PDF, nội dung của tài liệu và một số hướng dẫn cụ thể cho

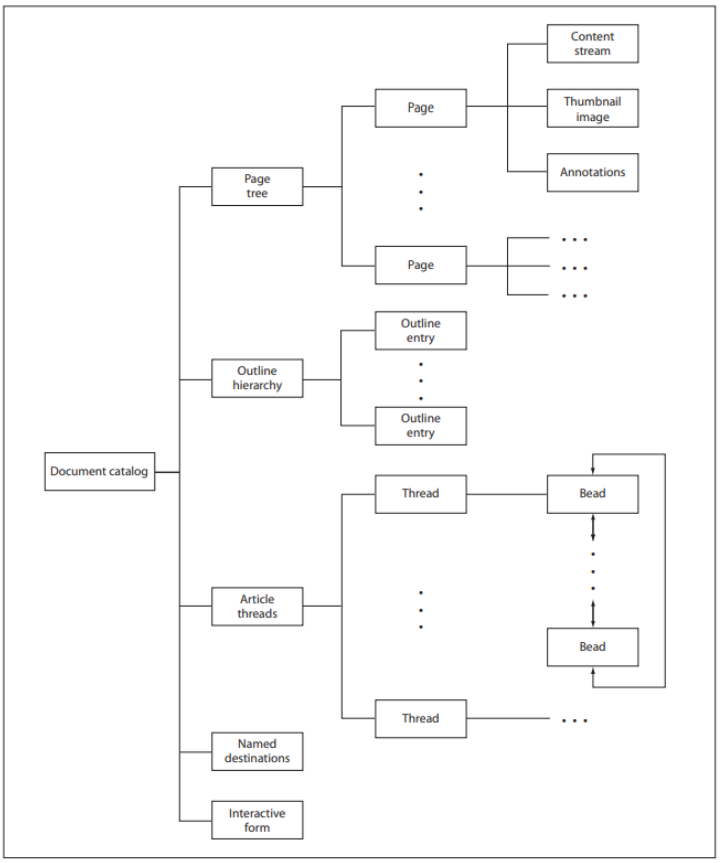
[2] `https://www.iso.org/standard/51502.html`



Hình 1.4: Cấu trúc tệp PDF khi có cập nhật [1]

các trình đọc PDF về cách hiển thị PDF lên màn hình sau khi được mở. Hình 1.6 là một ví dụ về một đối tượng catalog trong tệp PDF, với từ khóa 'PageMode' xác định cách tệp sẽ được hiển thị, 'OpenAction' chỉ định hành động sẽ được thực thi ngay sau khi mở tệp, và 'Pages' tham chiếu tới nút gốc của một cây các trang trong tài liệu PDF.

Page Tree Các trang của tài liệu PDF được liên kết thông qua một cây các trang - định nghĩa thứ tự truy cập. Sử dụng cấu trúc cây cho phép trình đọc nhanh chóng mở các trang trong tài liệu chứa hàng nghìn trang. Nút gốc của cây được chỉ định thông qua tham chiếu /Pages trong đối tượng Catalog.



Hình 1.5: Mô hình phân cấp các đối tượng bên trong cấu trúc logic của tài liệu PDF [2]

```
1 0 obj
<< /Type /Catalog
   /Pages 2 0 R
   /PageMode /UseOutlines
   /Outlines 3 0 R
   /OpenAction 11 0 R
>>
endobj
```

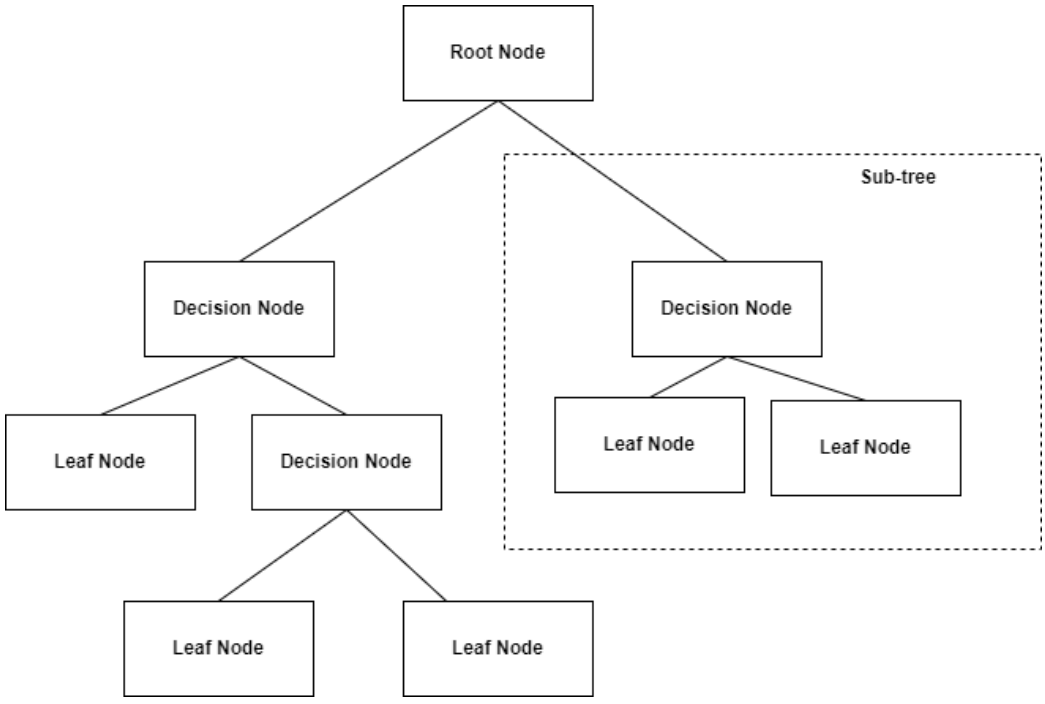
Hình 1.6: Đối tượng catalog trong tài liệu PDF

1.2 Một số thuật toán phân loại

Ngoài việc hiểu cấu trúc tài liệu PDF phục vụ cho việc phân tích và trích xuất những đặc trưng có giá trị thì việc huấn luyện một bộ phân loại hai lớp để phát hiện các mẫu độc hại với độ chính xác cao nhất có thể là vô cùng quan trọng. Thông thường, những mô hình phân loại sẽ được áp dụng vào tập dữ liệu một cách độc lập, từ đó xem xét mô hình nào hoạt động tốt nhất. Ở phạm vi khóa luận này, một số thuật toán phân loại được sử dụng trong các thử nghiệm, bao gồm Cây quyết định, Random Forest và LightGBM.

1.2.1 Cây quyết định

Cây quyết định là một thuật toán học máy có giám sát được sử dụng rộng rãi. Một cây quyết định sử dụng mô hình cấu trúc cây để biểu thị các dự đoán xuất phát từ một loạt các điều kiện phân tách dựa trên các đặc trưng, bắt đầu với nút gốc và kết thúc bằng một quyết định đưa ra bởi nút lá (Hình 1.7).



Hình 1.7: Sơ đồ cấu trúc cây trong thuật toán Cây quyết định

Nút gốc là nút bắt đầu của cây quyết định

Nút quyết định các nút sau nút gốc, chứa những điều kiện để phân chia tới các nút tiếp theo.

Nút lá nút kết thúc, chứa một nhãn duy nhất ứng với kết quả cuối cùng.

Cây con là các phần nhỏ của đồ thị biểu diễn cây quyết định, theo sau bởi nút gốc hoặc một nút quyết định

Cây quyết định phân loại bằng cách duyệt mẫu từ trên xuống dưới bắt đầu từ gốc của cây, tại mỗi nút quyết định, điều kiện sẽ được kiểm tra và mẫu sẽ được phân phối xuống các nút tiếp theo, cho đến khi đến một nút lá. Nhãn của nút lá sẽ chính là kết quả phân loại cuối cùng cho mẫu đó.

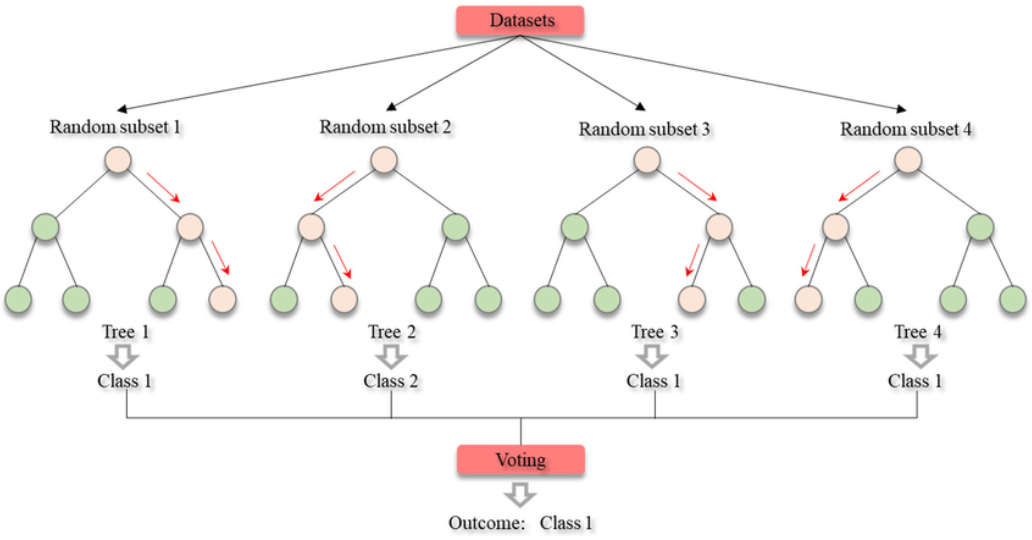
Trên thực tế, thuật toán đã được sử dụng trong rất nhiều công bố về phân loại tệp PDF độc hại: Davide Maiorca và cộng sự [7], Himanshu Pareek và cộng sự [8]. Ngoài ra một số công cụ phát hiện PDF độc hại như Hidost [13], Lux0R [2] hay PDF Malware Slayer [6] cũng sử dụng thuật toán này trong mô hình phân loại.

1.2.2 Random Forest

Random Forest là một thuật toán học kết hợp, bằng cách xây dựng rất nhiều các mô hình cây quyết định tạo thành rừng, mang lại một kết quả tốt hơn khi chỉ sử dụng một cây quyết định đơn lẻ. Mỗi cây quyết định trong rừng sẽ được học độc lập dựa trên một tập con ngẫu nhiên từ tập huấn luyện, và chỉ sử dụng một tập con các đặc trưng ngẫu nhiên của dữ liệu. Khi thực hiện phân loại, mẫu kiểm thử sẽ được phân loại song song trên các cây và trung bình cộng của các kết quả sẽ cho ra nhãn phân loại cuối cùng (Hình 1.8).

Trong thuật toán Cây quyết định, khi xây dựng cây với một độ sâu tùy ý, có thể gây nên việc mô hình khớp hoàn toàn với dữ liệu tập huấn luyện, nhưng lại mang lại kết quả kém trên các tập kiểm thử (hiện tượng overfitting). Các cây trong thuật toán Random Forest chỉ dùng một phần dữ liệu huấn luyện để học, do đó sẽ không gặp overfitting nhưng có thể gây ra underfitting - khi số tập dữ liệu không đủ hoặc số lượng đặc trưng được sử dụng quá ít, dẫn tới kết quả xấu trên cả tập huấn luyện lẫn tập kiểm thử. Tuy nhiên, kết quả của thuật toán Random Forest là sự tổng hợp từ tất cả các cây quyết định, nhờ đó, các cây sẽ bổ sung cho nhau và mang lại một kết quả phân loại tốt hơn.

Thuật toán Random Forest đã được sử dụng trong các công cụ phát hiện mã độc như PDFRate [12], PDF Malware Slayer [6] và LuxOR [2].



Hình 1.8: Quy trình phân loại dựa trên thuật toán Random Forest [18]

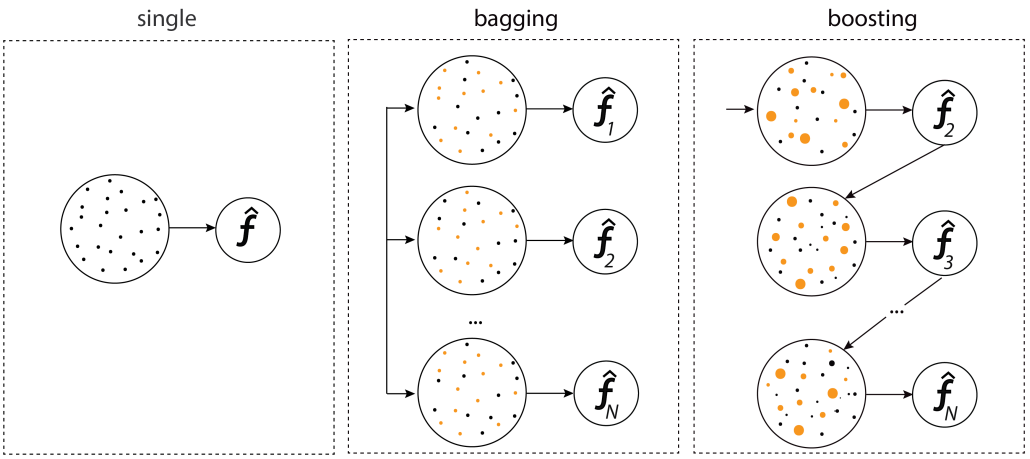
1.2.3 Kỹ thuật boosting và thuật toán LightGBM

Nhận thức được điểm yếu khi chỉ sử dụng một mô hình đơn lẻ như Cây quyết định, thuật toán Random Forest đưa ra một cải tiến mới khi kết hợp sử dụng mô hình trên nhiều phần dữ liệu khác nhau, huấn luyện độc lập và song song với nhau để đưa ra một kết quả tốt nhất. Đây là một ví dụ điển hình cho kỹ thuật bagging trong phương pháp học kết hợp. Ngoài ra, với phương pháp học kết hợp, một kỹ thuật được gọi là boosting cũng sẽ thực hiện xây dựng một chuỗi các mô hình trên những tập dữ liệu con khác nhau nhưng hoạt động học sẽ diễn ra một cách tuần tự. Theo đó, mô hình sau trong quá trình huấn luyện sẽ rút kinh nghiệm từ những lỗi của mô hình trước đó bằng cách cập nhật trọng số (Hình 1.9). Đầu ra của mô hình cuối cùng trong chuỗi sẽ là kết quả cuối cùng khi thực hiện phân loại.

Trong phạm vi của khóa luận này, tôi đề xuất sử dụng một thuật toán boosting được phát triển bởi Microsoft là LightGBM [3]. LightGBM viết tắt của Light Gradient Boosting Machine, là một mô hình xử lý thuật toán học kết hợp tăng cường. Mô hình này cũng sử dụng Cây quyết định để xây dựng các mô hình thành phần. Sau mỗi lần thực hiện huấn luyện trên một mô hình với một tập dữ liệu con, thuật toán sẽ cập nhật tham số của mô hình theo hướng giảm của đạo hàm hàm mất mát.

Với nhiều kỹ thuật và cơ chế phức tạp, thuật toán LightGBM mang lại một bộ phân loại mạnh mẽ, tốc độ tính toán được tối ưu, do đó cần ít thời

[3] `https://lightgbm.readthedocs.io/en/v3.3.2/`



Hình 1.9: So sánh các kỹ thuật được sử dụng trong mô hình sử dụng phương pháp học kết hợp

gian xử lý hơn so với các thuật toán cùng loại khác trên các tập huấn luyện có kích thước lớn [4].

1.3 Phát hiện tài liệu PDF độc hại

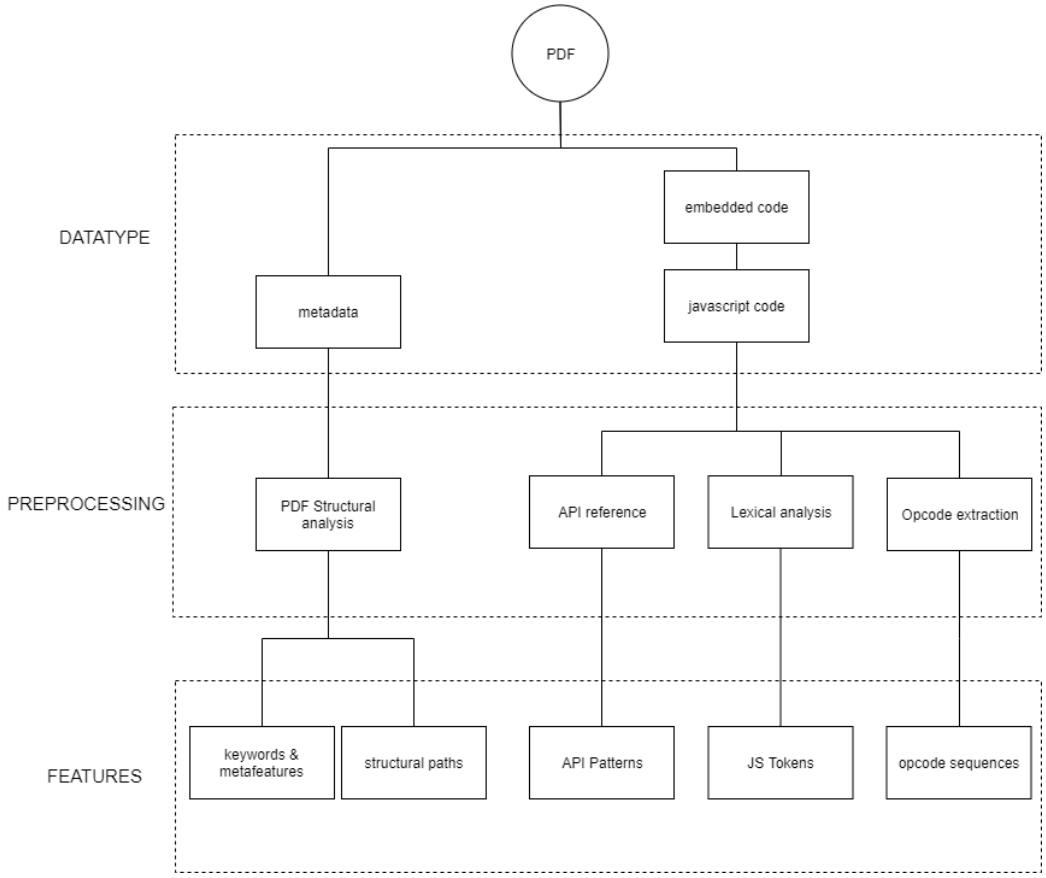
Phát hiện các tệp độc hại có rất nhiều cách tiếp cận, tuy nhiên, có hai giai đoạn chính phục vụ mục đích này

- Trích xuất đặc trưng
- Phân tích đặc trưng và đưa ra quyết định.

Trong giai đoạn trích xuất đặc trưng, các tệp PDF được xử lý và phân tích để trích xuất ra các đặc trưng cần thiết. Những đặc trưng này sẽ được thống kê và được đưa qua một bộ lọc để chọn ra những đặc trưng quan trọng trong giai đoạn Phân tích đặc trưng. Cuối cùng, một mô hình học máy sẽ được áp dụng để quyết định xem tệp đầu vào là sạch hay độc hại.

Mô hình 1.10 mô tả các đặc trưng được đề xuất để phát hiện tài liệu độc hại, và tổ chức chúng theo mô hình phân loại phân cấp. Mỗi cấp độ sẽ tương ứng với từng giai đoạn trong hoạt động trích xuất đặc trưng. Mô hình được giản lược từ mô hình được giới thiệu bởi Michele Elingiusti và cộng sự trong công bố năm 2018 [3].

[4] `https://lightgbm.readthedocs.io/en/v3.3.2/`



Hình 1.10: Phân loại các đặc trưng được sử dụng trong phát hiện tài liệu PDF độc hại

Loại dữ liệu

Ở cấp đầu tiên, đại diện cho loại dữ liệu được trích xuất từ tài liệu PDF

- Siêu dữ liệu (metadata) là tất cả những dữ liệu có thể được trích xuất từ dữ liệu "thô" của tệp PDF từ những mô tả, các mối liên hệ và cấu trúc bên trong tài liệu. Các siêu dữ liệu bao gồm các từ khóa xuất hiện trong tệp, số lượng các ký tự kết thúc tệp, tác giả, đường dẫn cấu trúc,...
- Embedded code gồm các đoạn mã hay các tệp được nhúng vào trong tài liệu PDF, có thể kể đến một số tệp thực thi hay một số mã shellcode, điển hình và xuất hiện nhiều nhất là các mã JavaScript.

Tiền xử lý

Tiền xử lý cho thấy các kỹ thuật xử lý được sử dụng ứng với từng loại dữ liệu đầu vào, phục vụ cho mục đích cụ thể là đưa ra những đặc trưng hữu ích trong việc phát hiện tài liệu độc hại. Kết quả của quá trình xử lý này là một tập các đặc trưng - được liệt kê ở cấp cuối cùng. Các đặc trưng này sẽ là đầu vào của các mô hình học máy khi thực hiện phân loại tệp độc hại.

- Phân tích cấu trúc của tệp PDF nhằm mục tiêu trích xuất các đặc trưng thống kê như từ khóa hay đường dẫn cấu trúc.
- Tham chiếu API mã JavaScript: Quá trình xử lý và nhận diện các API được sử dụng trong mã JavaScript, bằng phương pháp phân tích tĩnh hoặc phân tích động. Cách tiếp cận này đã được đề xuất trong LuxOr [2]. Đầu ra của quá trình này là các mẫu API được sử dụng nhiều trong các tài liệu độc hại hay các tài liệu lành tính.
- Phân tích từ vựng: các đoạn mã JavaScript thường bị xáo trộn và làm phức tạp hóa, điều này yêu cầu một dạng biểu diễn trừu tượng hóa để loại bỏ các chi tiết không cần thiết và cô lập những đoạn mã thực sự liên quan để dễ dàng trong việc nhận diện và phát hiện bất thường. Một quy trình phân tích từ vựng mã nguồn sẽ hỗ trợ tự động hóa quá trình trừu tượng hóa như vậy. Cách tiếp cận này đã được đề xuất bởi Vatamanu và cộng sự [16] sử dụng công cụ SpiderMonkey [5] để trích xuất ra các JavaScript token.
- Trích xuất opcode: một kỹ thuật tấn công được sử dụng trong các tài liệu PDF độc hại là xây dựng các đoạn mã shellcode trong thời gian thực thi bằng cách sao chép các chuỗi opcode ẩn giấu trong các biến. Do đó, một phương pháp phát hiện được đề xuất thực hiện phân tích động để xác định các biến có thể chứa các chuỗi opcode độc hại hoặc đáng ngờ. PDF Scrutinizer [11] xây dựng một phương pháp phỏng đoán đề xác định vị trí nào cần để phân tích. MDScan [14] chọn các chuỗi cần tìm shellcode bằng cách quan sát các chuỗi được tạo ra trong thời gian chạy, quét các vùng nhớ của các chuỗi mới được cấp phát.

[5] `https://spidermonkey.dev/`

Chương 2

Kết hợp đặc trưng thống kê và cấu trúc cây trong phát hiện PDF độc hại

Ở chương trước, tôi đã giới thiệu về quy trình phát hiện một tài liệu PDF độc hại, gồm hai phần chính là trích xuất đặc trưng và phân loại dữ liệu. Chương hai này sẽ tiến hành thử nghiệm, từ cài đặt công cụ trích xuất đặc trưng trên tập dữ liệu thu thập được đến xây dựng mô hình học máy mang lại độ chính xác cao khi phân loại một tệp sạch hay độc hại. Các đặc trưng được trích xuất đều dựa trên phân tích tĩnh tệp PDF, gồm hai loại đặc trưng thống kê và đường dẫn cấu trúc. Phần cuối của chương sử dụng mô hình học máy để đánh giá kết quả khi kết hợp các đặc trưng trên.

2.1 Tập dữ liệu

Tập dữ liệu bao gồm 22046 tệp PDF riêng biệt, với tỉ lệ tệp độc là 58%, tệp sạch là 42%, không gây sự chênh lệch quá lớn trên hai nhãn. Trong đó số lượng tài liệu có nhúng JavaScript chiếm 75% (Bảng 2.1). Toàn bộ tập dữ liệu là kết quả của quá trình thu thập và loại bỏ tệp trùng lặp từ một số nguồn như Contagio [9], Evasive-PDFMal2022 [1] và VirusTotal [2]. Với VirusTotal, thực hiện truy vấn để tìm kiếm các tệp PDF độc hại có nhúng JavaScript và được gán nhãn CVE có độ nguy hiểm cao. Một số CVE được thu thập trong các tài liệu PDF độc như CVE-2007-5659, CVE-2008-2992, CVE-2009-4324, CVE-2010-0188,...

[1] `https://www.unb.ca/cic/datasets/pdfmal-2022.html`

[2] `https://www.virustotal.com/`

Bảng 2.1: Thống kê số lượng tệp dữ liệu có chứa JavaScript.

	Chứa JavaScript	Không chứa mã JavaScript	Tổng
tệp độc hại	12183	584	12767
tệp lành tính	4326	4953	9279

Bảng 2.2: Phân bổ tập huấn luyện, kiểm chứng và tập kiểm thử

	Samples
Tập huấn luyện	15431
Tập kiểm chứng	4409
Tập kiểm thử	2205

Các tệp PDF sau khi được xử lý qua quá trình trích xuất đặc trưng, sẽ được phân chia thành các tập dữ liệu phục vụ quá trình huấn luyện và kiểm định. Với mô hình yêu cầu một tập validation phục vụ cho quá trình điều chỉnh tham số hay lựa chọn bộ đặc trưng tốt, sự phân chia sẽ theo tỉ lệ 7:2:1 với 70% là tập huấn luyện, 20% là tập kiểm chứng, 10% là tập kiểm thử (Bảng 2.2).

2.2 Trích xuất đặc trưng

2.2.1 Đặc trưng thống kê

Các đặc trưng thống kê là các đặc trưng được trích xuất từ quá trình phân tích cú pháp, cấu trúc của tệp PDF, mang ý nghĩa thống kê như đếm số lượng từ khóa xuất hiện trong tài liệu, phân tích tương quan giữa các giá trị, số lượng bộ lọc được sử dụng...

Trình đọc PDF sử dụng các từ khóa được nhúng trong tài liệu, theo sau bởi ký tự '/' để hiểu những hành động cần thực thi, do đó tập hợp những từ khóa có thể là các chỉ báo hiệu quả về hành vi của tài liệu. Một số từ khóa có thể có liên quan chặt chẽ tới một số lỗ hổng hoặc hành vi độc hại, điển hình như '/JS' hay '/JavaScript' chỉ ra có đoạn mã JavaScript được nhúng trong tài liệu, những đoạn mã này có độ nguy hiểm cao bởi xuất hiện trong hầu hết các cuộc tấn công mã độc; hay '/URI': cho phép truy cập một nguồn từ xa, bằng cách này, kẻ tấn công có thể chuyển hướng người dùng tới một trang web độc hại.

Tài liệu PDF hỗ trợ một số các bộ lọc phục vụ nén, mã hóa dữ liệu. Những kẻ tấn công có thể lợi dụng bộ lọc để ẩn các đoạn mã độc hại và che giấu hành vi. Do đó, việc xác định những bộ lọc nào được sử dụng, những bộ

lọc nào thường xuất hiện trong tệp độc hại hay tệp sạch sẽ mang lại những đặc trưng có giá trị trong việc phân loại tệp.

Ngoài ra đặc trưng thống kê cũng bao gồm các metafeatures - các đặc trưng phản ánh phần nào các thuộc tính của siêu dữ liệu, ví dụ như số lượng các từ viết hoa trong trường tác giả, hay tỉ lệ của số trang trên kích thước tài liệu.

Thông qua việc kết hợp sử dụng hai công cụ PeePDF [3] và PDFiD [4], tôi thu lại được tập các đặc trưng thống kê được lựa chọn trích xuất. Bộ đặc trưng này gồm có 67 đặc trưng (Bảng 2.3).

Bảng 2.3: Các đặc trưng thống kê

Tên đặc trưng	**Mô tả**
file_size	Kích thước tệp
pdfid_check_isPDF	Kiểm tra xem tệp có phải là tài liệu PDF hay không
is_linearized	Đặc trưng này dùng để kiểm tra tệp có là Linearized PDF hay không
is_encrypted	Kiểm tra xem tệp có bị mã hóa hay không
pdf_version	Phiên bản tài liệu PDF
num_updates	Số lần tệp PDF được cập nhật
num_objects	Số lượng đối tượng xuất hiện trong tệp
num_streams	Số lượng các đối tượng stream
peepdf_num_errors	Số lượng lỗi khi phân tích cú pháp tệp qua công cụ PeePDF
pdfid_error_occured	Kiểm tra tệp có bị lỗi khi phân tích qua công cụ PDFiD hay không
bad_pdf_header	Kiểm tra sự không hợp lệ của header tệp PDF
num_comments	Số lượng comment trong tệp
num_indirect_objects	Số lượng đối tượng tham chiếu trong tệp
num_compressed_objects	Số lượng các đối tượng được nén
num_encode_streams	Số lượng stream được mã hóa
num_decode_streams_error	Số lượng lỗi xuất hiện khi thực hiện giải mã những stream bị mã hóa
num_error_objects	Số lượng các đối tượng lỗi
num_object_streams	Số lượng các stream chứa đối tượng

[3] `https://github.com/jesparza/peepdf`

[4] `https://www.kali.org/tools/pdfid/`

num_xref_streams	Số lượng xref stream (từ phiên bản 1.5, tệp PDF cho phép bảng tham chiếu được lưu dưới dạng stream với từ khóa /ObjStm)
num_objects_contain_js	Số lượng đối tượng chứa đoạn mã JavaScript
num_%%EOF	Số lần xuất hiện của từ khóa đánh dấu sự kết thúc của tệp PDF
char_after_last_EOF	Số lượng ký tự sau từ khóa kết thúc cuối cùng trong tệp PDF
stream_entropy_less_than_2	Kiểm tra xem entropy của tệp có nhỏ hơn 2 hay không (entropy tính toán sự phân bổ ngẫu nhiên của các bytes trong chuỗi)
total_file_entropy	Giá trị entropy của toàn bộ tệp
stream_entropy	Giá trị entropy bên trong các đối tượng stream
non_stream_entropy	Giá trị entropy của các chuỗi bên ngoài stream
obj	Số lần xuất hiện của từ khóa 'obj', từ khóa xác định sự bắt đầu của một đối tượng
endobj	Số lần xuất hiện của từ khóa 'endobj', từ khóa xác định sự kết thúc của một đối tượng
stream	Số lần xuất hiện của từ khóa 'stream', từ khóa xác định sự bắt đầu của một stream
endstream	Số lần xuất hiện của từ khóa 'endstream', từ khóa xác định sự kết thúc của một stream
xref	Số lần xuất hiện của từ khóa 'xref', từ khóa xác định sự bắt đầu của bảng tham chiếu tới vị trí các đối tượng trong tệp PDF
trailer	Số lần xuất hiện của từ khóa 'trailer', từ khóa xác định trailer của tệp PDF
startxref	Số lần xuất hiện của từ khóa 'startxref', từ khóa xác định vị trí bắt đầu của bảng tham chiếu.
/Page	Số lần xuất hiện của từ khóa 'Page', xác định một trang của tệp PDF
/Encrypt	Số lần xuất hiện của từ khóa 'Encrypt'
/ObjStm	Số lần xuất hiện của từ khóa 'ObjStm'
/JS	Số lần xuất hiện của từ khóa 'JS', xác định đoạn mã JavaScript
/JavaScript	Số lần xuất hiện của từ khóa 'JavaScript', tương tự như từ khóa 'JS

/AA	Số lần xuất hiện của từ khóa 'AA', xác định các hành động bổ sung, các sự kiện kích hoạt một hành vi nào đó
/OpenAction	Số lần xuất hiện của từ khóa 'OpenAction', xác định hành động sẽ được thực thi ngay sau khi tệp PDF được mở
/AcroForm	Số lần xuất hiện của từ khóa 'AcroForm', xác định các phần tử biểu mẫu tương tác
/Names	Số lần xuất hiện của từ khóa 'Names'
/RichMedia	Số lần xuất hiện của từ khóa 'RichMedia', thường chứa embedded flash
/Launch	Số lần xuất hiện của từ khóa 'Launch', được sử dụng để mở một thực thể bên ngoài, có thể là một tệp thực thi.
/XFA	Số lần xuất hiện của từ khóa 'XFA'
/JBIG2Decode	Số lần xuất hiện của từ khóa 'JBIG2Decode'
/U3D	Số lần xuất hiện của từ khóa 'U3D'
/PRC	Số lần xuất hiện của từ khóa 'PRC'
/Flash	Số lần xuất hiện của từ khóa 'Flash'
/SubmitForm	Số lần xuất hiện của từ khóa 'SubmitForm'
/ImportData	Số lần xuất hiện của từ khóa 'ImportData'
Colors >2^24	Số lượng màu có kích thước lớn hơn 2^24 (hơn 3 bytes)
combination_js_and_action	Kiểm tra sự xuất hiện đồng thời của các đoạn mã JavaScript và các hành động khởi chạy
num_EmbeddedFile	Số lượng tệp được nhúng trong tài liệu PDF
num_URI	Số lần xuất hiện của từ khóa 'URI', đường dẫn tới một nguồn từ xa, có thể là một trang web
num_ASCIIHexDecode	Số lần sử dụng thuật toán giải mã ASCIIHexDecode
num_ASCII85Decode	Số lần sử dụng thuật toán giải mã ASCII85Decode
num_LZWDecode	Số lần sử dụng thuật toán giải mã LZWDecode
num_FlateDecode	Số lần sử dụng thuật toán giải mã FlateDecode
num_RunLengthDecode	Số lần sử dụng thuật toán giải mã RunLengthDecode

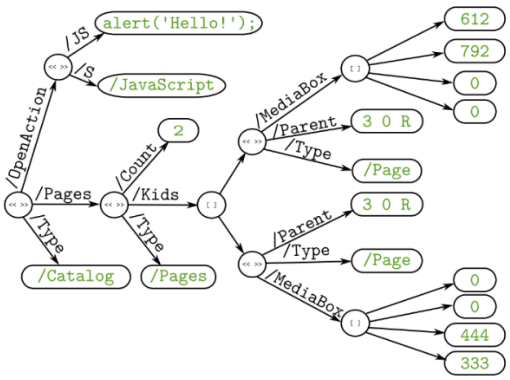
num_CCITTFaxDecode	Số lần sử dụng thuật toán giải mã CCITTFaxDecode
num_JBIG2Decode	Số lần sử dụng thuật toán giải mã JBIG2Decode
num_DCTDecode	Số lần sử dụng thuật toán giải mã DCTDecode
num_JPXDecode	Số lần sử dụng thuật toán giải mã JPXDecode
num_Crypt	Số lượng stream bị mã hóa
num_stream_filters	Số lượng bộ lọc được sử dụng
num_unknown_stream_filters	Số lượng bộ lọc được sử dụng nhưng không trong danh sách các bộ lọc được hỗ trợ của tài liệu PDF

2.2.2 Đặc trưng cấu trúc cây

Như đề cập ở mục 1.1.3, các đối tượng của tệp PDF được tổ chức phân cấp dạng cây. Do đó việc xem xét cách sắp xếp các đối tượng trên cấu trúc cây như vậy có thể tiết lộ những thông tin giá trị để phát hiện tài liệu độc hại. Phương pháp tiếp cận này được đề xuất bởi Nedim Srndic và Lavel Laskov [13]. Ở phương pháp này, mỗi đường dẫn cấu trúc tới lá trong cây sẽ đại diện cho một đặc trưng, tập hợp các đặc trưng sau đó sẽ được xử lý thông qua một kỹ thuật gọi là Hợp nhất đường dẫn cấu trúc để hợp nhất những đường dẫn tương tự nhau.

Đường dẫn cấu trúc

Đường dẫn cấu trúc trong cây cấu trúc PDF được định nghĩa là một chuỗi các cạnh bắt đầu từ nút gốc Catalog và kết thúc bởi một đối tượng đại diện cho nút lá. Ví dụ theo hình 2.1 , một đường dẫn cấu trúc bắt đầu bởi nút gốc đi qua cạnh /Pages và /Count để tới nút lá có giá trị là 2, được biểu diễn dưới dạng chuỗi '/Pages/Count'.



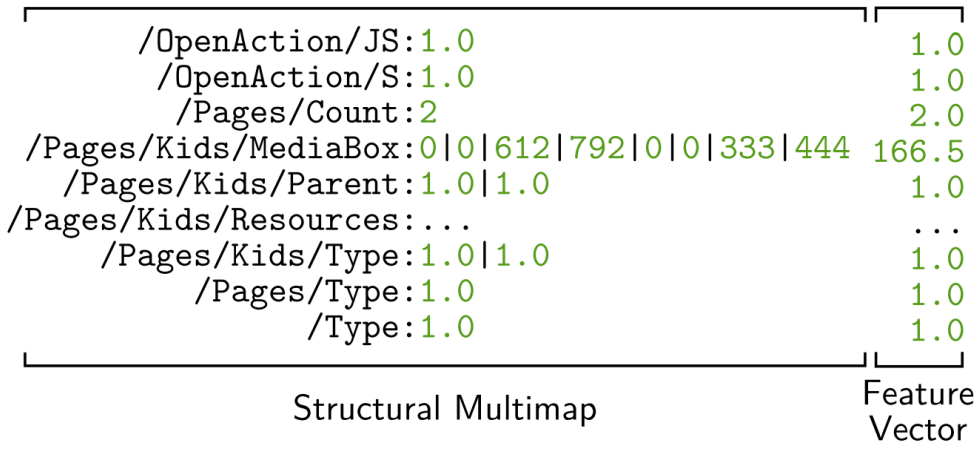
/Metadata
/Type
/Pages/Kids
/OpenAction/Contents
/StructTreeRoot/RoleMap
/Pages/Kids/Contents/Length
/OpenAction/D/Resources/ProcSet
/OpenAction/D
/Pages/Count
/PageLayout

Hình 2.1: Minh họa cấu trúc cây và đường dẫn cấu trúc của tệp PDF

Để tạo ra một đặc trưng, các đường dẫn sẽ trở thành tên, còn giá trị sẽ được xác định dựa trên giá trị của nút lá ứng với đường dẫn đó. Liên quan đến việc xử lý các dữ liệu ở nút lá mà không phải dạng số (dữ liệu dạng số bao gồm số nguyên, số thực, boolean), thì các dữ liệu kiểu chuỗi sẽ được thay thế bởi một giá trị cố định. Một mảng các số thì được thay thế bằng trung bình cộng của mảng đó. Với các đường dẫn xuất hiện nhiều lần, tham chiếu tới nhiều giá trị, thì sẽ hợp nhất thành một đường dẫn duy nhất có giá trị là một mảng các giá trị trên. Hình 2.2 và 2.3 mô phỏng lại cách thuật toán chuẩn hóa các giá trị về dạng số như vừa đề cập.

Keys	Values
/OpenAction/JS:	alert('Hello!');
/OpenAction/S:	/JavaScript
/Pages/Count:	2
/Pages/Kids/MediaBox:	0|0|612|792|0|0|333|444
/Pages/Kids/Parent:	3 0 R|3 0 R
/Pages/Kids/Resources:	...
/Pages/Kids/Type:	/Page|/Page
/Pages/Type:	/Pages
/Type:	/Catalog

Hình 2.2: Các cặp đường dẫn cấu trúc và giá trị chưa chuẩn hóa



Hình 2.3: Các cặp đường dẫn cấu trúc và giá trị sau khi được chuẩn hóa về dạng số

Hợp nhất đường dẫn cấu trúc

Một tệp PDF có cấu trúc tương đối phức tạp, do đó có thể trích xuất được một lượng lớn các đường dẫn và có nhiều đường dẫn có cấu trúc tương tự nhau. Do đó, một tập các quy tắc được xây dựng để hợp nhất các đường dẫn tương đương [ref]

Lựa chọn đặc trưng

Một số đường dẫn có thể xảy ra rất ít trong tập dữ liệu được trích xuất, việc giữ những đặc trưng này làm tăng kích thước của không gian đầu vào mà không cải thiện được độ chính xác của bộ phân loại. Do đó, việc lựa chọn đặc trưng cần được thực hiện để hạn chế tác động của những đặc trưng ít xuất hiện. Ở trong phạm vi cài đặt thí nghiệm sử dụng công cụ Hidost, lựa chọn đặc trưng dựa trên việc đặt ngưỡng cho sự phổ biến của đặc trưng trong tệp dữ liệu.

Công cụ Hidost

Công cụ Hidost [5] là công cụ được phát triển dựa theo ý tưởng trên, thực hiện trích xuất các đặc trưng đường dẫn cấu trúc của một tệp PDF, ngoài ra còn hỗ trợ các tệp SWF. Đầu vào của công cụ là một tập các tệp PDF, đầu ra là một tập các tệp dữ liệu đã được trích xuất tương ứng với từng tệp

[5] `https://github.com/srndic/hidost`

đầu vào. Trong quá trình cài đặt, với số lượng mẫu hơn 20000 tệp, tôi đã tiến hành đặt ngưỡng 900, tức các đặc trưng đường dẫn xuất hiện trong ít nhất 900 tệp sẽ được sử dụng trong tập đặc trưng cuối cùng. Kết quả sau khi thực thi, có 315 đặc trưng được trích xuất, phục vụ cho quá trình huấn luyện xây dựng mô hình học máy ở các phần sau. Do số lượng đặc trưng là tương đối lớn nên sẽ không được liệt kê trong khóa luận này, mà được liệt kê tại Các đặc trưng cấu trúc cây.

2.3 Phân loại và đánh giá kết quả

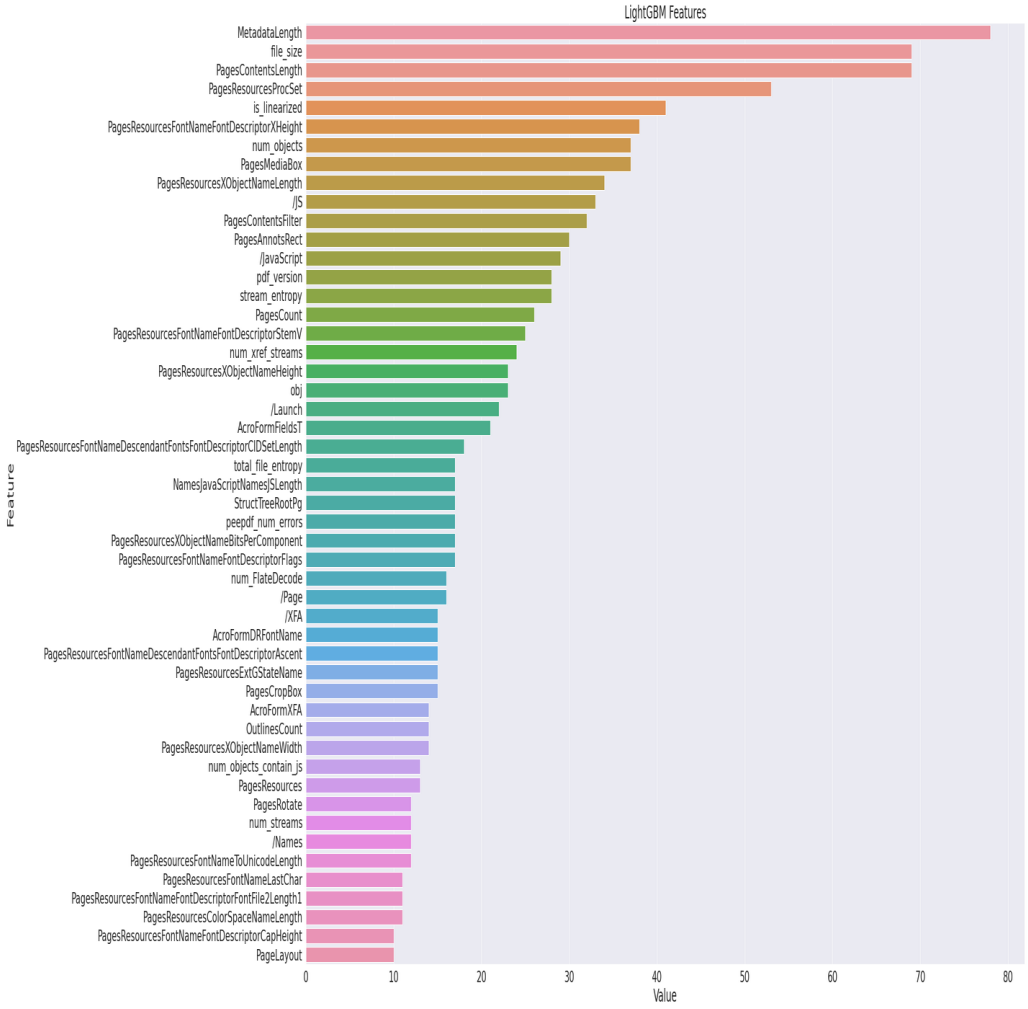
Như đã đề cập ở mục 1.3, một quy trình phát hiện tự động tệp PDF độc hại gồm trích xuất đặc trưng và xây dựng mô hình phân loại. Kết quả của quá trình thử nghiệm trích xuất đặc trưng trong mục 2.2 mang lại một tập các dữ liệu tệp đã được xử lý, là đầu vào cho quá trình phân loại này. Khi thực hiện phân loại, với mục tiêu xây dựng một mô hình học máy có thể mang lại kết quả dự đoán tốt nhất, các đặc trưng đã được xếp hạng, lựa chọn và đánh giá trên tập kiểm định.

Hai thuật toán học máy được cân nhắc đưa vào thử nghiệm gồm Random Forest và LightGBM. Ngoài ra, ở cả hai thuật toán này, tôi sử dụng thêm **optuna** [6] - một framework hỗ trợ điều chỉnh tự động các tham số cho mô hình học máy để mang lại kết quả tốt nhất mong muốn. Sau khi thử điều chỉnh tham số với 100 vòng lặp, nhận thấy LightGBM mang lại kết quả tốt hơn trên tập kiểm định với độ chính xác tốt nhất là 99,68%, trong khi RandomForest mang lại độ chính xác tốt nhất là 99.47%. Từ đây, thuật toán LightGBM sẽ được sử dụng trong tất cả các kết quả dưới đây.

2.3.1 Xếp hạng đặc trưng

Quá trình trích xuất sử dụng các công cụ PDFiD, PeePDF và Hidost mang lại tổng 382 đặc trưng, trong đó có 67 đặc trưng thống kê và 315 đặc trưng đường dẫn cấu trúc. Thông qua thuật toán LightGBM, mức độ quan trọng của các đặc trưng được đánh giá như hình 2.4

[6] `https://optuna.org`



Hình 2.4: Biểu đồ 50 đặc trưng có độ quan trọng cao nhất trong các đặc trưng thống kê và đường dẫn cấu trúc.

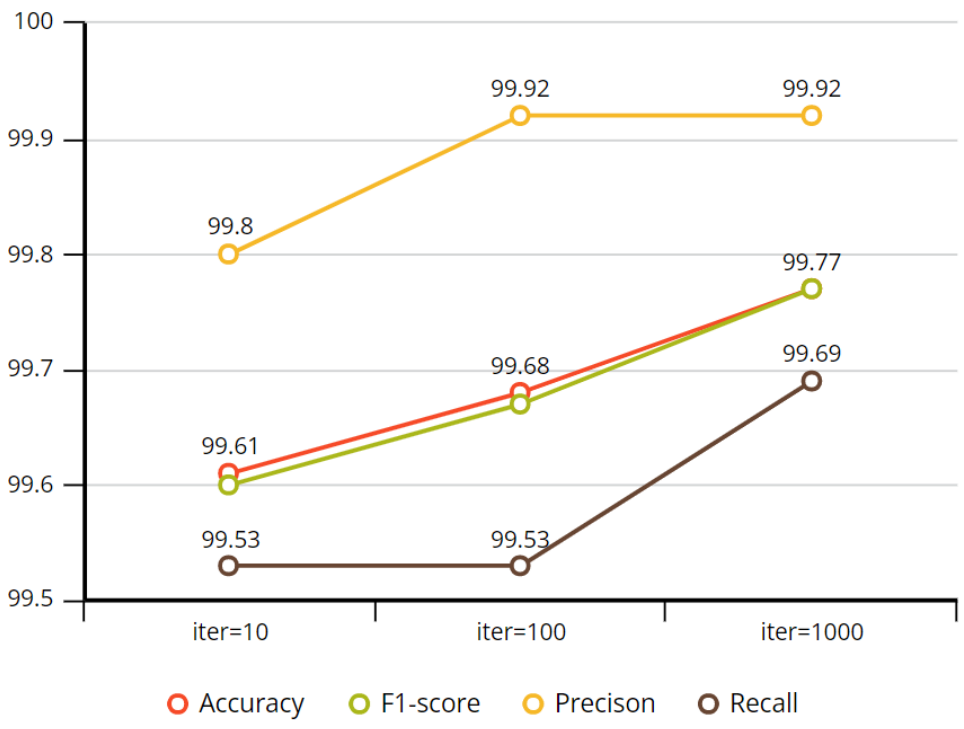
Quan sát kết quả xếp hạng độ quan trọng các đặc trưng, có thể thấy các đặc trưng liên quan đến kích thước dữ liệu, nội dung trong tệp PDF mang lại độ quan trọng cao nhất, ví dụ như Metadata/Length, Pages/Contents/Length hay file_size.

Ngoài ra một số đặc trưng liên quan đến đoạn mã JavaScript hay liên quan đến biểu mẫu tương tác AcroForm trong tài liệu PDF cũng mang lại sự hữu ích trong mô hình phân loại khi xuất hiện nhiều trong top 50 này, ví dụ /JS, /JavaScript, AcroFormDRFontName, AcroFormXFA, AcroFormFieldsT, NamesJavaScriptNameJSLength.

2.3.2 Điều chỉnh tham số và số lượng đặc trưng

Như đã giới thiệu, ở thí nghiệm này, tôi sử dụng optuna để tự động điều chỉnh tham số đầu vào cho thuật toán học máy LightGBM. Công cụ optuna sử dụng một kỹ thuật gọi là Tree-Parzen Estimator [1] đề chọn ra một tập các siêu tham số cho lần thử tiếp theo, dựa trên lịch sử của các lần điều chỉnh trước đó. Vậy nên, sau nhiều vòng lặp, công cụ này có thể mang lại một tập các tham số đầu vào tốt hơn cho thuật toán.

Qua quá trình lặp lại với nhiều lần điều chỉnh, tôi đã chọn được bộ tham số mang lại kết quả tốt nhất cho thuật toán. Dưới đây là biểu đồ so sánh kết quả kiểm định trên các mô hình LightGBM tốt nhất được ghi lại tại sau lần điều 10, 100 và 1000. Biểu đồ 2.5 đã cho thấy hiệu quả của việc điều chỉnh tham số trong mô hình học máy.

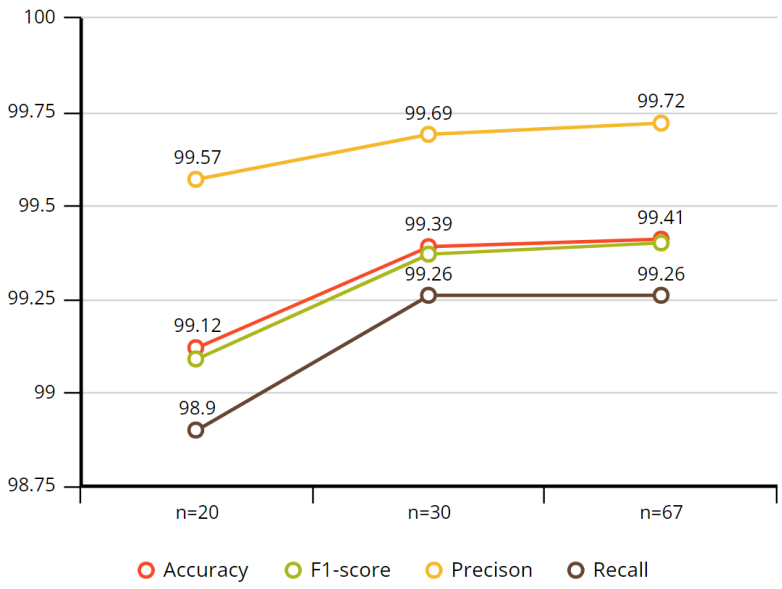


Hình 2.5: Biểu đồ so sánh kết quả thuật toán LightGBM qua các lần điều chỉnh trên tập kiểm chứng

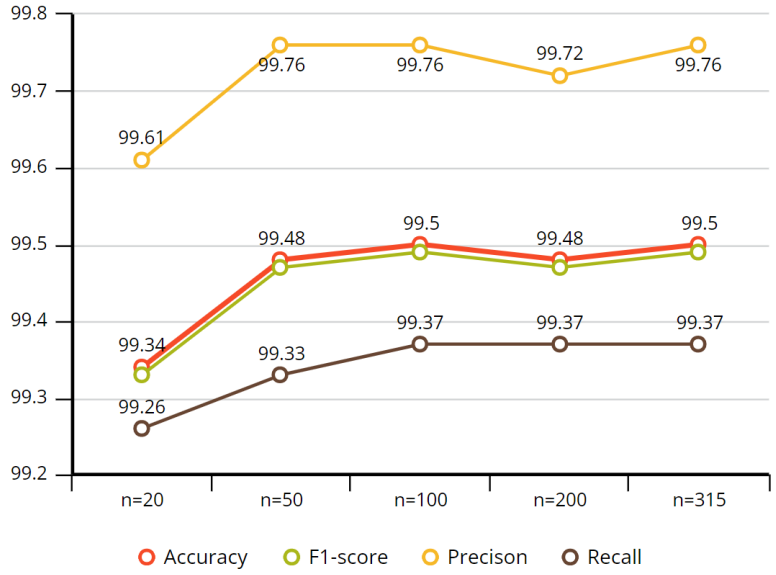
Bên cạnh việc điều chỉnh các tham số, tôi tiếp tục thử nghiệm thuật toán với việc điều chỉnh số lượng đặc trưng từ tập đặc trưng ban đầu:

- Hình 2.6: chỉ sử dụng các đặc trưng thống kê, và thực hiện thay đổi số lượng đặc trưng, kết quả cho thấy việc sử dụng toàn bộ tập đặc trưng thống kê mang lại kết quả tốt nhất với độ chính xác 99.41%
- Hình 2.7: chỉ sử dụng các đặc trưng đường dẫn cấu trúc, và thực hiện thay đổi số lượng đặc trưng, kết quả cho thấy việc sử dụng toàn bộ tập đặc trưng cấu trúc cây mang lại kết quả tốt nhất với độ chính xác 99.5%
- Hình 2.8: thay đổi số lượng đặc trưng trên toàn bộ đặc trưng đã trích xuất được (bao gồm cả đặc trưng thống kê và đặc trưng cấu trúc cây). Kết quả cũng mang tại độ chính xác tốt nhất là 99.77% khi sử dụng toàn bộ đặc trưng.

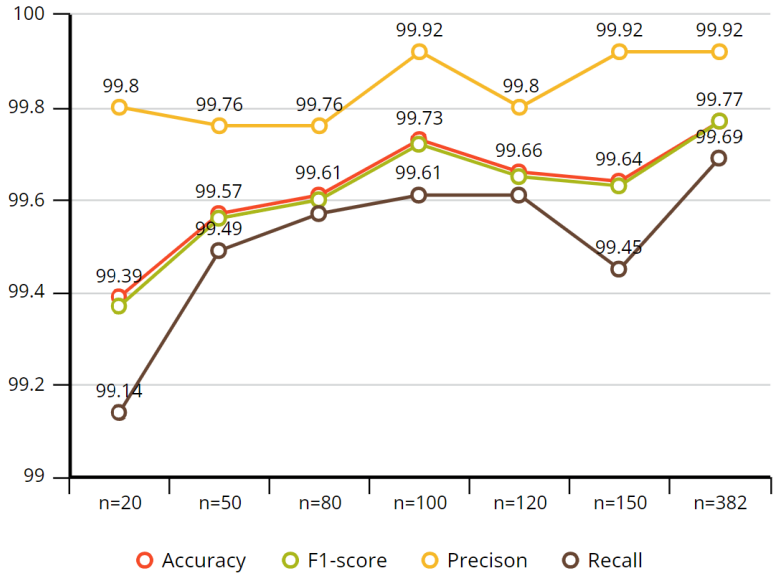
Việc sử dụng nhiều hơn các đặc trưng bằng thực nghiệm trên đã chứng minh là mang lại kết quả tốt hơn trên mô hình học máy LightGBM.



Hình 2.6: Biểu đồ so sánh mô hình khi sử dụng số lượng đặc trưng tăng dần từ tập đặc trưng thống kê.



Hình 2.7: Biểu đồ so sánh kết quả của mô hình khi sử dụng số lượng đặc trưng tăng dần từ tập đặc trưng cấu trúc cây

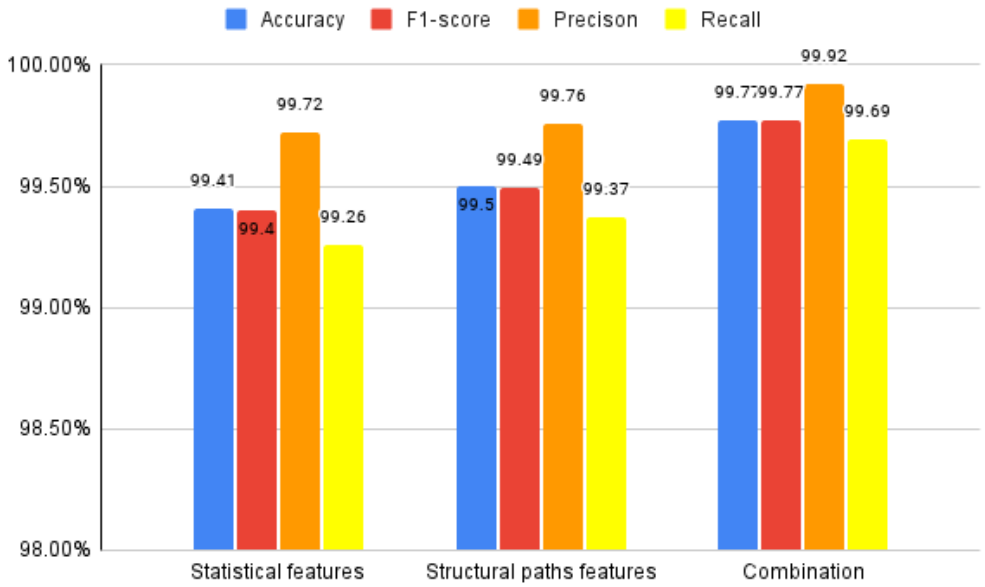


Hình 2.8: Biểu đồ so sánh kết quả của mô hình khi sử dụng số lượng đặc trưng tăng dần từ sự kết hợp các tập đặc trưng thống kê và đặc trưng cấu trúc cây.

2.3.3 Kết hợp các loại đặc trưng

Để đánh giá sự kết hợp của các đặc trưng thống kê và đường dẫn cấu trúc, thực hiện 3 thử nghiệm cho mô hình khi chỉ sử dụng tập đặc trưng thống kê, chỉ sử dụng tập đặc trưng cấu trúc cây và cuối cùng là sử dụng kết hợp hai loại đặc trưng này. Kết quả được biểu diễn trên biểu đồ 2.9, đã cho thấy một kết quả tốt nhất khi kết hợp các loại đặc trưng này với nhau.

Từ đó có thể kết luận, trong quy trình xác định một tài liệu PDF có độc hại hay không, việc phân tích và trích xuất nhiều loại đặc trưng nhất có thể, ở đây là các đặc trưng mang tính thống kê và các đặc trưng cấu trúc sẽ mang lại một mô hình phân loại đáng tin cậy hơn, thay vì chỉ sử dụng một số loại đặc trưng đơn lẻ.



Hình 2.9: Biểu đồ so sánh kết quả của mô hình khi sử dụng hai loại đặc trưng độc lập và khi kết hợp các đặc trưng thống kê và đặc trưng cấu trúc cây

Chương 3

Xây dựng mô hình phân loại Zero False Positive

3.1 Đặt vấn đề

Hiện nay, phương thức tấn công phổ biến nhất đối với các tệp PDF độc hại bắt nguồn từ việc nhúng các đoạn mã JavaScript - những đoạn mã mà sẽ được thực thi bởi ứng dụng đọc tệp PDF. Lợi dụng nhiều lỗ hổng bảo mật nghiêm trọng từ các trình đọc PDF, có thể kể đến nhiều nhất là Acrobat Reader, các kẻ tấn công đã sử dụng mã JavaScript để thực hiện khai thác các lỗ hổng đó, nhằm xâm nhập vào hệ thống máy tính nạn nhân và thực hiện những hành vi độc hại. Việc phân tích tĩnh và trích xuất các đặc trưng thực hiện ở chương 2 chỉ mang tính chất thống kê và kiểm tra sự tồn tại của JavaScript, chưa thực sự đi sâu vào khai thác các hành vi của mã. Bên cạnh đó, với sự phát triển của kỹ thuật làm rối, các đoạn mã JavaScript càng trở nên khó phát hiện hơn, hoặc được xây dựng trông như một đoạn mã lành tính, gây nhiễu cho các mô hình phân loại học máy. Có thể thấy các đặc trưng đã trích xuất chưa thực sự đem lại một kết quả tốt với các mã có nhúng JavaScript. Ở chương này, tôi đề xuất một phương pháp gồm hai giai đoạn để xử lý các tệp PDF có chứa JavaScript.

- Giai đoạn một thực hiện trích xuất đặc trưng cơ bản và đưa qua bộ phân loại Zero False Positive với mục tiêu của giai đoạn này là tạo nên một tường lửa nhạy bén với các tệp độc, đảm bảo loại bỏ được những tệp chắc chắn là độc, tức mô hình học máy không gây ra dương sai (FP).

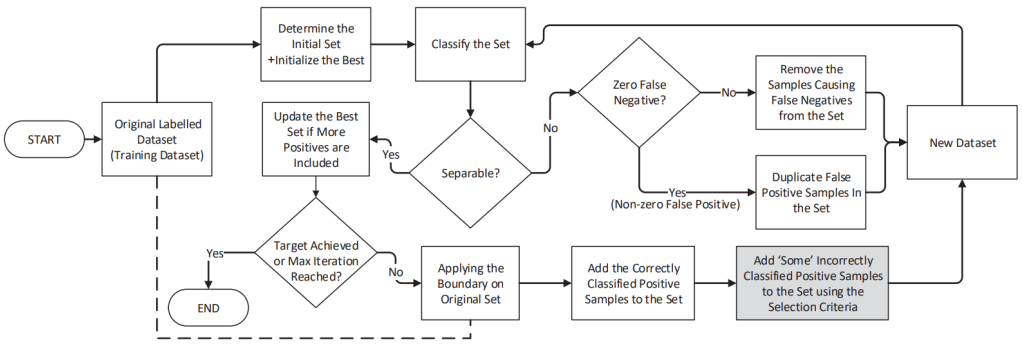
- Giai đoạn hai được đề xuất sẽ xử lý các tệp PDF được gán nhãn sạch, bằng các kỹ thuật xử lý nâng cao, đòi hỏi thời gian và chi phí phân

tích hơn để tìm ra chính xác tệp PDF độc hại nào còn sót lại.

3.2 Bộ phân loại Zero False Positive

Bộ phân loại Zero False Positive được đề xuất trong giai đoạn một để mang tới một kết quả phân loại không có gán nhãn dương sai. Ý tưởng xây dựng dựa trên mô hình được đề xuất trong công bố của Mohammad Sayad Haghighi và cộng sự [4].

3.2.1 Thuật toán Zero False Positive



Hình 3.1: Sơ đồ luồng hoạt động của thuật toán lặp Zero False Positive

Thuật toán là kỹ thuật phân loại lặp với mục tiêu tỷ lệ phát hiện sai cho lớp dương là bằng 0 (tức không có mẫu âm nào bị phát hiện thành mẫu dương), dựa trên thuật toán Tối ưu bầy đàn [17]. Từ một tập dữ liệu ban đầu, sẽ tạo ra nhiều các tập giản lược, được gọi là tập thành phần. Mỗi tập giản lược sẽ bao gồm tất cả các mẫu âm và một vài mẫu dương. Mỗi tập thành phần bắt đầu với một tập giản lược mà được trích xuất từ tập huấn luyện ban đầu T bằng cách loại bỏ một số mẫu dương để có thể trở nên tách được (tức FP và FN bằng 0). Ở trường hợp lý tưởng, tập khởi tạo này chỉ có 1 mẫu dương và tất cả là mẫu âm. Cho trước một bộ phân loại C, chúng ta có thể thực hiện phân loại trên tập T ban đầu, và sử dụng những mẫu dương được phân loại đúng để tạo ra các tập khởi tạo. Mỗi tập giản lược sẽ đi qua bộ phân loại C. Trong mỗi vòng lặp, nếu kết quả cho thấy sự tách được của tập (FP = 0 và FN = 0), tập sẽ được so sánh với kết quả tốt nhất trước đó của các tập thành phần khác, gọi là Best. Sự so sánh có thể là một hàm được định nghĩa trước ví dụ chứa bao nhiêu mẫu dương khi chúng ta muốn giảm số lượng FN. Nếu tập giản lược của tập thành phần

Algorithm 1 The Proposed Iterative Classification Technique with Zero False Positive (ʒ-Classifier)

Require: Training Set($\mathcal{T}$), Performance Target/Stop Criterion
Ensure: Zero False Positive Classification

Create initial separable particle sets $S_1, ...S_k$
Determine the $Best$ set (with the most positive samples)
while Stop Criterion/Performance Target is not met **do**
  for each S_i **do**
    Classify S_i with $\mathcal{C}$ and find the boundary $\mathcal{B}_i$
    if S_i is separable **then**
      Update the $Best$ if S_i is more fit
      Apply $\mathcal{B}_i$ to $\mathcal{T}$ & add the true positives to S_i
      Use the $Best$ and S_i knowledge to add k_i false negative samples to S_i
    else
      if there is false negatives **then**
        Remove false negative samples from S_i
      else
        Duplicate false positive samples in S_i
      end if
    end if
  end for
end while

Hình 3.2: Mã giả thuật toán lặp Zero False Positive

mang lại số điểm tốt hơn, Best sẽ được thay thế. Từ đó, biên mới của mô hình sẽ được dùng để phân loại lại tập dữ liệu ban đầu (như một tập kiểm thử). Theo đó, Các mẫu dương được phân loại đúng sẽ được thêm vào tập thành phần Si cho vòng lặp tiếp theo. Ngoài ra các mẫu dương bị phân loại sai (gây FN) sẽ được chọn lọc một phần (k mẫu) để thêm vào Si bằng cách sử dụng phương pháp Lựa chọn ngẫu nhiên dựa trên trọng số trên Best. Bởi những mẫu dương trong tập giản lược của Best sẽ có thể có trọng số, tức độ ưu tiên cao hơn. Ở một trường hợp khác, thông qua các vòng lặp, nếu một

Bảng 3.1: Kết quả của thuật toán phân loại Zero False Positive trên tập dữ liệu đã trích xuất

Iteration	TN	FP	FN	TP
0	7423	0	2993	7220
5	7423	0	361	9852
10	7423	0	255	9988
50	7423	0	133	10080
100	7423	0	107	10106
200	7423	0	105	10108
500	7423	0	100	10113
700	7423	0	97	10116
1000	7423	0	97	10116

tập thành phần trở nên không phân tách được tại một số điểm (ở đây là FP khác 0), thì sẽ được thông qua một quá trình cắt tỉa (pruning operation). Sự cắt tỉa này ban đầu sẽ được thực hiện bằng việc loại bỏ những mẫu dương mà gây nên phân loại âm sai (FN $>$ 0). Nếu cách này không mang lại hiệu quả (sẽ được kiểm tra trong vòng lặp tiếp theo) hoặc không mang lại FP = 0, trọng số của các mẫu âm gây nên FP sẽ được tăng lên - bằng cách nhân bản các mẫu âm này trong tập thành phần. Có thể thấy, tập giản lược của tập thành phần có thể không nhất thiết là một tập con của tập huấn luyện T ban đầu. Quá trình cắt tỉa sẽ tiếp tục qua các vòng lặp cho đến khi tập này trở nên phân tách lại một lần nữa. Toàn bộ quá trình thêm/ loại bỏ sẽ kết thúc khi hoặc đạt được mục tiêu FN (luôn muốn FN nhỏ nhất có thể) hoặc đã đạt số vòng lặp cho phép tối đa.

3.2.2 Thực nghiệm và đánh giá kết quả

Tiến hành thực nghiệm để kiểm định mô hình trên tập huấn luyện gồm các tệp PDF đã được trích xuất đặc trưng thống kê và đặc trưng cấu trúc trong chương 2. Kết quả cho thấy số lượng FP qua các vòng lặp đều được đảm bảo bằng 0 và số lượng FN giảm dần. Tuy nhiên, từ vòng lặp thứ 700 - 1000, kết quả FN không giảm, duy trì số lượng 97, với tỉ lệ FNR bằng 0.0095 (Bảng 3.1).

Ngoài ra, mô hình được tiến hành thử nghiệm trên một số tập dữ liệu khác. Bộ dữ liệu đặc trưng tệp PDF được trích xuất và công bố bởi đội ngũ phát triển Hidost [1] với một số lượng lớn các mẫu PDF độc hại được thu thập qua nhiều năm. Tôi đã liên hệ Nedim Srndic, một trong những tác giả của

[1] `https://github.com/srndic/hidost-reproduction`

Bảng 3.2: Kết quả thuật toán phân loại Zero False Positive trên tập dữ liệu Hidost (1)

Iteration	**TN**	**FP**	**FN**	**TP**
0	142796	0	1851	4406
10	142796	0	410	5847
50	142796	0	287	5970
100	142796	0	272	5985
120	142796	0	271	5986
160	142796	0	271	5986

Bảng 3.3: Kết quả thuật toán phân loại Zero False Positive trên tập dữ liệu Hidost (2)

iteration	**TN**	**FP**	**FN**	**TP**
0	132465	0	4919	2484
10	132465	0	539	6864
50	132465	0	317	7086
100	132465	0	305	7098
190	132465	0	305	7098

Hidost [13] để xin phép sử dụng bộ dữ liệu đặc trưng trong phạm vi khóa luận này. Bộ dữ liệu được phân thành nhiều tập, và dưới đây là kết quả của hai tập dữ liệu khi thử nghiệm trên mô hình phân loại Zero False Positive.

Với số lượng tập sạch lớn, chênh lệch so với số lượng tập độc, tập dữ liệu của Hidost yêu cầu thời gian chạy lớn khi thực hiện thuật toán phân loại Zero False Positive, và tỉ lệ FNR còn khá cao, 0.043 với tập thứ 1 và 0.041 ở tập thứ 2.

3.3 Phương hướng xử lý các tệp được gán nhãn sạch

Sau khi lọc các tệp PDF có chứa mã JavaScript qua mô hình phân loại Zero False Positive, các tệp được gán nhãn sạch sẽ có nguy cơ chứa các tệp độc hại. Từ đây các mẫu này sẽ tiếp tục được xử lý tại giai đoạn hai: thực hiện phân tích và trích xuất những đặc trưng liên quan tới JavaScript được nhúng trong tài liệu PDF. Từ đó, mục tiêu sẽ lọc được tất cả các tệp độc còn lại.

Dưới đây, tôi sẽ giới thiệu về một số phương hướng xử lý đoạn mã JavaScript, sau đó giới thiệu về một công cụ phục vụ mục tiêu của giai

đoạn này.

Có thể thấy, JavaScript là một ngôn ngữ lập trình được sử dụng trong rất nhiều ứng dụng khác nhau, bao gồm các trang web, trình đọc PDF,... Từ góc độ chức năng, JavaScript mang lại các giao diện thân thiện cho người dùng với các chức năng nâng cao. Từ góc độ bảo mật, JavaScript là một ngôn ngữ mạnh mẽ được sử dụng rộng rãi bởi các tội phạm mạng nhằm thực hiện các thủ đoạn khai thác độc hại. Tài liệu PDF độc hại sử dụng JavaScript thường được đặc trưng bởi các hành động sau:

- Giải mã: một quy trình giải mã sẽ được thực hiện để trích xuất mã khai thác, bởi những đoạn mã này thường được mã hóa bởi một thuật toán cụ thể và lưu trữ trong các đối tượng của tệp.

- Kiểm tra môi trường chạy: các đoạn mã khai thác có thể điều chỉnh hành vi theo môi trường thực thi. Điều này nhằm mục tiêu tập trung vào các lỗ hổng ảnh hưởng tới một phiên bản trình đọc PDF cụ thể, hoặc có thể nhằm né tránh việc phát hiện bằng phân tích động, ví dụ đoạn mã sẽ dừng thực thi nếu phát hiện đang trong môi trường debug (kỹ thuật antidebug rất hay được sử dụng trong các tài liệu độc hại).

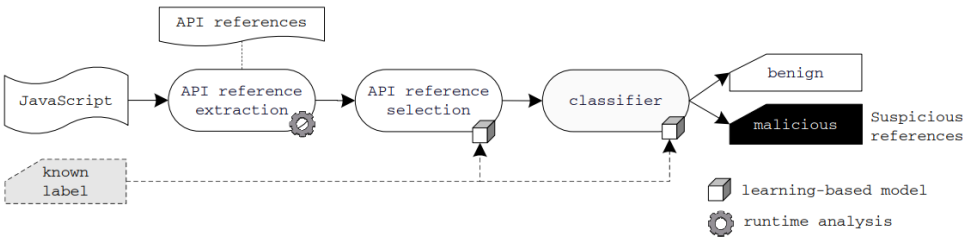
- Thực thi: khi các điều kiện tiên quyết về môi trường chạy được đáp ứng, các đoạn mã sẽ được thực thi. Các đoạn mã này có thể dựa vào một số lỗ hổng và kỹ thuật khai thác để tấn công các trình đọc và nắm kiểm soát toàn bộ hệ điều hành.

Nhằm phục vụ cho việc phân tích hành vi độc hại, đầu tiên, các đoạn mã JavaScript sẽ được trích xuất tĩnh hoặc động. Với phương pháp tĩnh, mã sẽ được trích xuất trực tiếp từ tệp, qua việc tìm vị trí của các đoạn mã JavaScript (thường được tìm thông qua các từ khóa "/JavaScript" hoặc "/JS"). Trong khi đó, ở phương pháp động, tệp sẽ được mở và lấy được chính xác đoạn mã nào sẽ thực thi (đặc biệt hữu ích trong việc chống lại kỹ thuật obfuscation). Việc trích xuất động sẽ yêu cầu một môi trường an toàn để thực thi và đòi hỏi nhiều tài nguyên hơn.

Các đoạn mã javascript được trích xuất theo đó sẽ được xử lý thông qua một số kỹ thuật như phân tích từ vựng, trích xuất bytecode hay tham chiếu các API nhằm đưa ra tập đặc trưng cuối cùng sử dụng trong các mô hình phân loại. Trong phạm vi khóa luận này, em tập trung giới thiệu về 2 kỹ thuật trích xuất các đặc trưng API JavaScript và kỹ thuật trích xuất đặc trưng bytecode

Đặc trưng API JavaScript

Trong một tài liệu PDF, ngoài sử dụng những API JavaScript của hệ thống, theo tiêu chuẩn ECMAScript như các hàm eval() - thực thi các lệnh JavaScript được mô tả trong chuỗi truyền vào hay hàm unescape() - thực hiện giải mã một chuỗi, thì các API JavaScript của các trình đọc PDF cũng được hỗ trợ. Những API này được tạo ra để giúp tương tác với trình đọc cụ thể. Trong công bố của mình, Davide Maiorca đã giới thiệu về một công cụ mạnh mẽ thực hiện phân tích động và trích xuất các đặc trưng API JavaScript tương tác với trình đọc Acrobat Reader. Hình 3.3 là mô hình kiến trúc của công cụ này - LuxOr [2].



Hình 3.3: Mô hình kiến trúc của LuxOr [2]

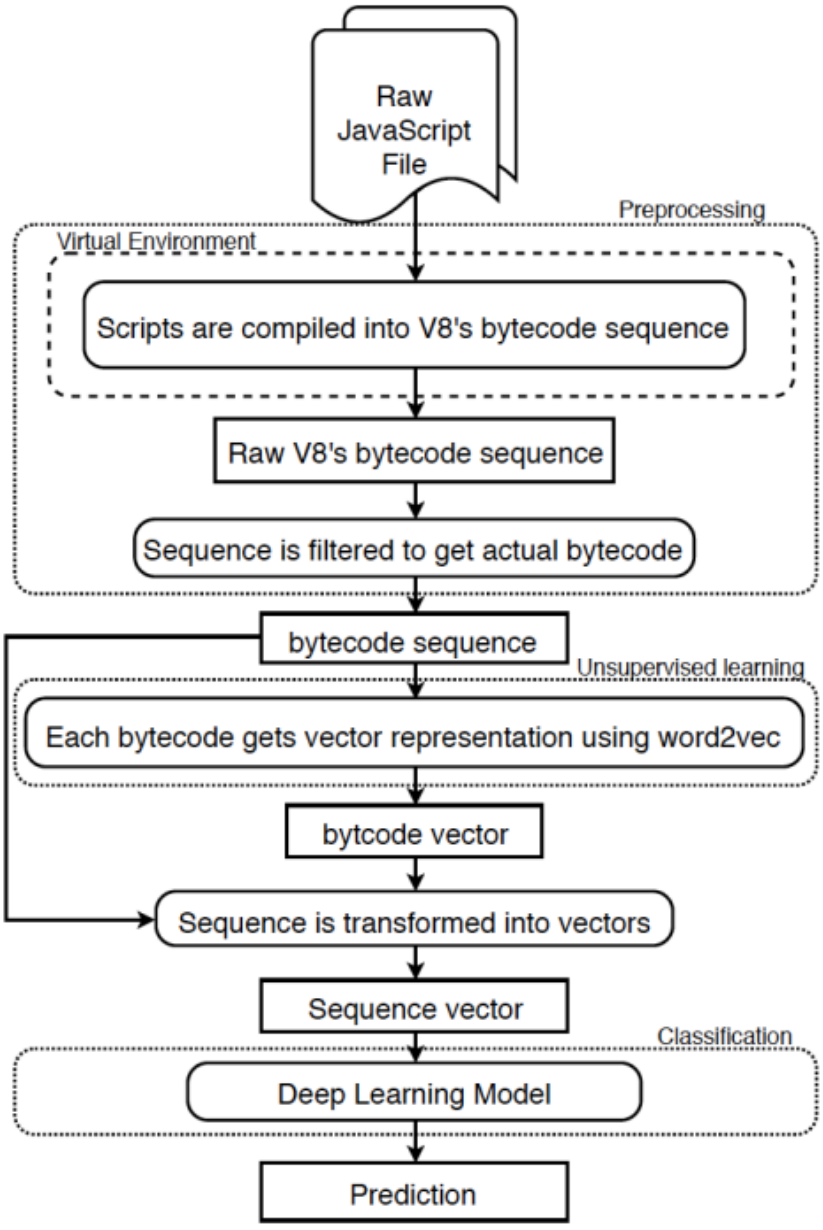
LuxOr sẽ thực hiện theo dõi tất cả các API JavaScript từ việc phân tích tĩnh và phân tích động. Sau đó, sẽ thông qua một quá trình lựa chọn API. Cuối cùng đưa ra một tập đặc trưng ứng với các API đã chọn để đưa vào quá trình phân loại. Hiện nay có các công cụ để trích xuất động API JavaScript như SpiderMokey hay Mozilla. Tuy nhiên những trình thông dịch như vậy nhận diện dựa trên tiêu chuẩn Js ECMA, không nhận diện được những API tương tác với trình đọc Acrobat Reader, ví dụ một số hàm như app.doc.syncAnnotScan(), app.plugIns.length, app.doc.getAnnots(). Do đó điểm nổi bật của LuxOr đã sử dụng công cụ PhoneyPdf được phát triển để giả lập và mô phỏng Adobe DOM, từ đó có thể thông dịch được các API Acrobat.

Kết quả thực nghiệm của LuxOr đã chứng minh rằng nó có hiệu quả trên rất nhiều các mẫu PDF chứa JavaScript, ngay cả với những CVE chứa những đoạn mã đã bị obfuscation. Có thể nói, công cụ này có thể được sử dụng phục vụ cho phát triển giai đoạn hai trong phương pháp phát hiện tài liệu PDF tôi đã đề xuất ở trên.

Đặc trưng chuỗi bytecode

Phương pháp tiếp cận trích xuất đặc trưng bytecode từ mã JavaScript sử dụng kỹ thuật phân tích động. Theo đó, một trình biên dịch mã JavaScript của Google là V8 Engine [2]được sử dụng để chuyển những mã JavaScript thành chuỗi bytecode - mã máy thực thi trong môi trường chạy thực tế. Hình 3.4 mô tả luồng hoạt động được đề xuất bởi M. F. Rozi và các cộng sự [10], gồm ba quá trình: tiền xử lý, học không giám sát và phân loại.

[2]`https://v8.dev/`



Hình 3.4: Mô hình luồng hoạt động của phương pháp trích xuất bytecode [10]

Quá trình tiền xử lý sử dụng cách tiếp cận trên, thực hiện biên dịch mã JavaScript nhúng trong tệp PDF để chuyển thành chuỗi bytecode trong một

môi trường ảo an toàn. Các chuỗi này sau đó được lọc và xử lý để có thể chuyển thành tập các đặc trưng bytecode cuối cùng. Những đặc trưng này là đầu vào cho quá trình học không giám sát sử dụng thuật toán học máy Word2vec để biểu diễn các đặc trưng chuỗi dưới dạng không gian vector. Ở giai đoạn cuối cùng, một kiến trúc mạng nơ-ron DPCNN [5] được sử dụng.

Quy trình trên đã được tiến hành thử nghiệm trên một tập dữ liệu với một số lượng lớn các tệp PDF có nhúng mã JavaScript cả lành tính lẫn độc hại, đem lại độ chính xác cao và tỉ lệ âm tính giả thấp.

Chương 4

Kết luận

4.1 Kết quả đạt được

Khóa luận đã trình bày những phương pháp phân tích, trích xuất đặc trưng và đề xuất các mô hình sử dụng trong phân loại tự động các tài liệu PDF độc hại.

Thứ nhất, khóa luận đã chỉ ra rằng, bằng cách trích xuất một tập hợp các đặc trưng trên phạm vi rộng, kết hợp nhiều loại đặc trưng khác nhau có thể tạo ra một mô hình phát hiện PDF độc hại mạnh mẽ và mang lại độ chính xác cao. Cụ thể, bằng việc sử dụng các công cụ PeePDF, PDFiD hay Hidost, những đặc trưng tĩnh đã được trích xuất bao gồm các đặc trưng có tính thống kê như từ khóa, metafeatures,... và các đặc trưng đường dẫn cấu trúc cây tài liệu PDF. Kết quả thí nghiệm đã chỉ ra rằng mô hình học máy mang lại kết quả tốt hơn khi tăng dần số lượng đặc trưng lựa chọn từ từng tập đặc trưng đã trích xuất. Bên cạnh đó, khi kết hợp hai tập đặc trưng thống kê và đặc trưng cấu trúc, mô hình đã cho thấy hiệu quả hơn khi chỉ sử dụng một loại đặc trưng, điều này được kiểm chứng bằng kết quả phân loại trên tập dữ liệu đã chuẩn bị.

Thứ hai, khóa luận đã đề xuất một phương pháp phát hiện tài liệu độc hại theo hai giai đoạn, với mục tiêu có thể xử lý tốt các tệp PDF có chứa những mã Javascript. Nhận thấy những tài liệu độc hại thường sử dụng nhiều kỹ thuật làm rối nâng cao, tinh vi nhằm che giấu những hành vi độc hại của các đoạn mã JavaScript và bắt chước tính bình thường của các tệp lành tính, thì chỉ sử dụng những đặc trưng thống kê và cấu trúc sẽ không thực sự hiệu quả, gây ra một số lượng âm tính giả cao. Từ đó, phương pháp chia hai giai đoạn được sử dụng để giải quyết vấn đề này. Ở giai đoạn một, khóa luận đã chỉ ra được một mô hình có thể mang lại tỷ lệ dương tính giả (FP) bằng 0, trong khi tỷ lệ âm tính giả (FN) được giữ ở mức tối thiểu trên tập huấn luyện. Mô

hình này được ứng dụng trong phạm vi khóa luận như một tường lửa, bước đầu lọc ra những tài liệu đảm bảo độc hại, trong khi những tài liệu được gán nhãn sạch sẽ tiếp tục được xử lý qua giai đoạn hai, thực hiện phân tích sâu hơn về các đoạn mã nhúng trong tài liệu. Mục tiêu này đã được kiểm chứng trên bộ dữ liệu với các đặc trưng thống kê và cấu trúc cây đã trích xuất ở thử nghiệm chương 2. Ngoài ra một số bộ dữ liệu đặc trưng đường dẫn cấu trúc cây được trích xuất trong công bố của Hidost [13] cũng được sử dụng để chứng minh tính đúng đắn của thuật toán Zero False Positive trên. Ở giai đoạn hai, khóa luận đã đề xuất các phương phương phát triển trong tương lai để xử lý các đoạn mã JavaScript bị làm rối, từ đó có thể hoàn toàn lọc được các tệp PDF độc hại còn sót lại.

4.2 Hạn chế và khó khăn

Trong quá trình cài đặt các công cụ trích xuất đặc trưng, tôi đã gặp những khó khăn:

- Các công cụ hỗ trợ trích xuất đặc trưng khá cũ, yêu cầu cài đặt môi trường và một số gói phụ thuộc (package dependencies) cụ thể, gây khó khăn trong việc tiến hành nhiệm vụ trích xuất đặc trưng. Ngoài ra nhiều tệp PDF có cấu trúc biến đổi, gây khó khăn cho một số trình phân tích cú pháp.
- Các đặc trưng API JavaScript đã được trích xuất nhưng số lượng ít và không mang lại kết quả khả quan nên không được đề cập tới trong thí nghiệm của khóa luận này.

Trong quá trình xây dựng mô hình và huấn luyện, với tập dữ liệu khá lớn, cùng với quá trình điều chỉnh tham số và chọn mô hình tốt nhất đòi hỏi một thời gian chạy dài. Bên cạnh đó, mô hình False Zero Positive đảm bảo không có dương tính giả trên tập huấn luyện, để đảm bảo kết quả này trên tập kiểm thử, cần phải xem xét ngưỡng dừng của tỉ lệ FNR ở mức không quá nhỏ hoặc cần có những cải tiến trong xây dựng mô hình.

4.3 Định hướng phát triển

Về định hướng phát triển đề tài trong tương lai, tôi đề xuất một số phương pháp phân tích và trích xuất được các đặc trưng JavaScript có giá trị để phục vụ phát hiện các tệp PDF độc hại.

- Sử dụng công cụ PhoneyPDF để trích xuất các API JavaScript tương tác với trình đọc PDF Acrobat Reader bằng cả phân tích tĩnh và phân tích động. Từ đó có thể có được một tập các đặc trưng gồm các API thể hiện hành vi của các đoạn mã javascript được nhúng trong tệp PDF [2].

- Sử dụng trình biên dịch JavaScript của Google là V8 để trích xuất các bytecode từ Javascript- dưới dạng mã máy, mục tiêu có thể chống lại sự ẩn giấu hành vi của các đoạn mã JavaScript bị làm rối. Rozi và các cộng sự [10] đã sử dụng phương pháp phân tích này để trích xuất ra được tập các đặc trưng là các chuỗi bytecode, sau đó sử dụng một mô hình học sâu DPCNN [5] để phân loại và phát hiện các tài liệu PDF độc hại.

- Một hướng phân tích tĩnh các đoạn mã JavaScript cũng hiệu quả được Vatamanu và cộng sự [16] đề xuất, sử dụng công cụ SpiderMonkey để trích xuất ra các đặc trưng JS Token - một dạng trừu tượng hóa mã JavaScript nhằm loại bỏ những chi tiết gây rối giúp dễ dàng nhận diện và phát hiện bất thường.

Phụ lục A

Chứng minh tính đúng đắn của mô hình Zero False Positive

A.1 Zero False Positive

Bài toán

Cho một tập dữ liệu với 2 nhãn và một bộ phân loại C, sau một số vòng lặp hữu hạn với thuật toán được đề xuất trên, có thể dẫn tới số lượng FP = 0.

Chứng minh

Cho tập dữ liệu $T(x_i; y_i)$ với $i = 1, \ldots, m$ và y thuộc $(-1; 1)$. Bộ phân loại C mục tiêu giảm thiểu hàm chi phí J được định nghĩa trước.

Chú ý rằng thuật toán sẽ tạo ra một tập giản lược từ tập T ban đầu, gồm tất cả các mẫu âm, do đó, nếu với đường biên Bi của bộ phân loại C giúp FP trên tập giản lược bằng 0 thì FP trên tập T ban đầu cũng sẽ bằng 0. Từ đó, chúng ta sẽ chứng minh FP sẽ có thể bằng 0 trên tập giản lược, gọi là tập S qua hữu hạn các vòng lặp.

Giả sử vòng lặp đầu tiên, bộ phân loại với đường biên $B(0)$ gây ra một số FN_s và FP_s trên tập S. Hàm chi phí của bộ phân loài là $J(FN_0, FP_0)$. Theo thuật toán Zero FP, những tập dương gây ra FN sẽ được loại bỏ khỏi tập giản lược S, thành $S(1)$ ($S(i)$ là tập S sau lần cập nhật thứ i). Khi đó, với biên $B(0)$, tập $S(1)$ sẽ có $FN_0 = 0$ và hàm chi phí trên tập $S(1)$ chỉ phụ thuộc vào FP, tức:

$J(FP_0) < J(FN_0, FP_0)$ hay $J(FP_0) = r_1 J(FN_0, FP_0)(0 \leq r_1 \leq 1)$

Tập được cập nhật $S(1)$ sau đó sẽ được phân loại lại với C với đường biên $B1$ mới, với hàm chi phí $J(FN_1, FP_1)$. Mục tiêu của hàm phân loại là

giảm hàm chi phí, do đó, hàm chi phí lúc này sẽ nhỏ hơn hàm chi phí trước khi phân loại.

$J(FN_1, FP_1) < J(FP_0)$ hay $J(FN_1, FP_1) = q_1 J(FP_0)(0 \leq q_1 \leq 1)$

Hiển nhiên nếu lúc này FP_1 bằng 0, chúng ta được điều phải chứng minh. Ngược lại, với $FP_1 > 0$, nếu $FN_1 > 0$ thì thuật toán tiếp tục thực hiện loại bỏ những mẫu dương gây FN_1, thành tập $S(2)$ với hàm chi phí trên $B(1)$ là

$J(FP_1) < J(FN_1, FP_1)$ hay $J(FP_1) = r_2 J(FN_1, FP_1)$

Tương tự như vậy, với thuật toán giản lược các mẫu dương bị sai (gây FN), tại vòng lặp thứ k, sau khi tìm ra $B(k)$ mới dựa trên phân loại $S(k)$ được giản lược từ vòng $k-1$, hàm chi phí là $J(FN_k, FP_k) = q_k J(FP_{k-1})$

Kết thúc vòng lặp này, nếu FP vẫn khác 0, thì sẽ thực hiện loại bỏ FN, trở thành

$J(FP_k) = r_k J(FN_k, FP_k)$

$\Rightarrow J(FP_k) = r_k q_k J(FP_{k-1}) = \ldots = T(r_i q_i) J(FP_0)$

Vì $T(r_i q_i) \to 0$. Do đó $lim_{k \to \infty} J(FP_k) = 0$ với biên Bk Từ đó khi với biên $B(k)$, tập S có $FP = 0$, thì tập ban đầu cũng có $FP = 0$

A.2 FN giảm dần

Bài toán

Chú ý rằng mục tiêu của thuật toán là $FP = 0$, nên chỉ khi $FP = 0$, chúng ta mới tập trung giảm FN. Do đó phần này sẽ chứng minh trong khi duy trì $FP = 0$, tập S phân tách được thì thuật toán sẽ giảm FN trên tập T ban đầu.

Chứng minh

Xét tập giản lược S, với $S(i)$ là lần cập nhật thứ i của tập S sau mỗi vòng lặp.

Từ điều kiện của giả thiết, giả sử $S(k)$ phân tách được với biên $B(k)$, tức $FP = FN = 0$. Biên này trên tập T ban đầu gây ra một lượng $TP_k > 0$ và $FN_k > 0$, thuật toán thực hiện thêm tất cả các mẫu dương được phân loại đúng (TP) và một vài mẫu dương trong FN_k mẫu dương bị phân loại sai (gây FN) vào tập $S(k)$, thành $S(k+1)$. Khi đó ở vòng lặp tiếp theo, tập $S(k+1)$ giữ nguyên số lượng mẫu âm, tăng số lượng mẫu dương. Do đó biên $B(k+1)$ sẽ cố gắng dịch chuyển về phía mẫu âm để hàm chi phí J được giảm, tức mô hình sẽ chú ý vào các mẫu dương bị phân loại sai (gây FN), cố gắng dịch chuyển để có thể phân loại đúng các mẫu này. Từ đó $FN_{k+1} \leq FN_k$, tức số lượng FN ở trên tập T có thể được giảm.

Tuy nhiên, theo như giả thiết, FN chỉ giảm khi FP được giữ về 0, ngay khi $FP > 0$, thuật toán sẽ thực hiện nhân bản tất cả mẫu âm hoặc lược bỏ các mẫu âm sai (FN), mục tiêu để tăng trọng số cho các mẫu âm, giảm lượng FP. Điều này có thể làm số lượng FN tăng lên. Do đó, giá trị của FN trong mô hình của Best là giá trị nhỏ nhất cục bộ.

Tài liệu tham khảo

[1] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in Neural Information Processing Systems* (Jan. 2011), pp. 2546–2554.

[2] Igino Corona et al. "Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References". In: vol. 2014. Nov. 2014. DOI: 10.1145/2666652.2666657.

[3] Michele Elingiusti, Leonardo Aniello, and Leonardo Querzoni. "PDF-Malware Detection: A Survey and Taxonomy of Current Techniques". In: Jan. 2018, pp. 169–191. ISBN: 978-3-319-73950-2. DOI: 10.1007/978-3-319-73951-9_9.

[4] Mohammad Sayad Haghighi, Faezeh Farivar, and Alireza Jolfaei. "A Machine Learning-based Approach to Build Zero False-Positive IPSs for Industrial IoT and CPS with a Case Study on Power Grids Security". In: (2020), pp. 1–1. ISSN: 0093-9994. DOI: 10.1109/tia.2020.3011397.

[5] Rie Johnson and Tong Zhang. "Deep Pyramid Convolutional Neural Networks for Text Categorization". In: Jan. 2017, pp. 562–570. DOI: 10.18653/v1/P17-1052.

[6] Davide Maiorca, Giorgio Giacinto, and Igino Corona. "A Pattern Recognition System for Malicious PDF Files Detection". In: vol. 7376. July 2012, pp. 510–524. ISBN: 9783642315367. DOI: 10.1007/978-3-642-31537-4_40.

[7] Davide Maiorca et al. "A Structural and Content-Based Approach for a Precise and Robust Detection of Malicious PDF Files". In: Feb. 2015. DOI: 10.5220/0005264400270036.

[8] Himanshu Pareek. "Entropy and n-gram analysis of malicious PDF documents". In: *International Journal of Engineering Research and Technology* (Feb. 2013).

[9] Mila Parkour. *16,800 clean and 11,960 malicious files for signature testing and research.* URL: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html.

[10] Muhammad Rozi, Sangwook Kim, and Seiichi Ozawa. "Deep Neural Networks for Malicious JavaScript Detection Using Bytecode Sequences". In: July 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207134.

[11] Florian Schmitt, Jan Gassen, and Elmar Gerhards-Padilla. "PDF Scrutinizer: Detecting JavaScript-based attacks in PDF documents". In: July 2012, pp. 104–111. ISBN: 978-1-4673-2323-9. DOI: 10.1109/PST.2012.6297926.

[12] Charles Smutz and Angelos Stavrou. "Malicious PDF detection using metadata and structural features". In: Dec. 2012, pp. 239–248. DOI: 10.1145/2420950.2420987.

[13] Nedim Srndic and Pavel Laskov. "Hidost: a static machine-learning-based detector of malicious files". In: *EURASIP J. Inf. Secur.* 2016 (2016), p. 22. DOI: 10.1186/s13635-016-0045-0.

[14] Zacharias Tzermias et al. "Combining static and dynamic analysis for the detection of malicious documents". In: *EUROSEC '11*. 2011.

[15] Caglar Ulucenk, Vijay Varadharajan, and Udaya Tupakula. "Techniques for Analysing PDF Malware". In: Dec. 2011, pp. 41–48. DOI: 10.1109/APSEC.2011.41.

[16] Cristina Vatamanu, Dragoş Gavriluţ, and Răzvan Benchea. "A practical approach on clustering malicious PDF documents". In: *Journal in Computer Virology* 8 (Nov. 2012). DOI: 10.1007/s11416-012-0166-z.

[17] Mohd Nadhir Ab Wahab, Samia Nefti-Meziani, and Adham Atyabi. "A Comprehensive Review of Swarm Optimization Algorithms". In: *PLOS ONE* 10.5 (May 2015). DOI: 10.1371/journal.pone.0122827.

[18] Contributors to Wikimedia projects. *Random forest.* June 2022. URL: https://en.wikipedia.org/wiki/Random_forest.