ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

MÔN HỌC: CƠ CHẾ HOẠT ĐỘNG CỦA MÃ ĐỘC

BÁO CÁO ĐỒ ÁN CUỐI KÌ

Mitigating adversarial evasion attacks of ransomware using ensemble learning

Giảng viên hướng dẫn: Phan Thế Duy

NT230.O21.ATCL

Sinh viên thực hiện :

Nhóm G1

21522620 – Hồ Ngọc Thiện

21522483 - Chu Nguyễn Hoàng Phương

TP.HCM, Ngày 3 tháng 6 năm 2024

MỤC LỤC

BÁO CÁO CHI TIẾT

I. Tổng quan bài báo

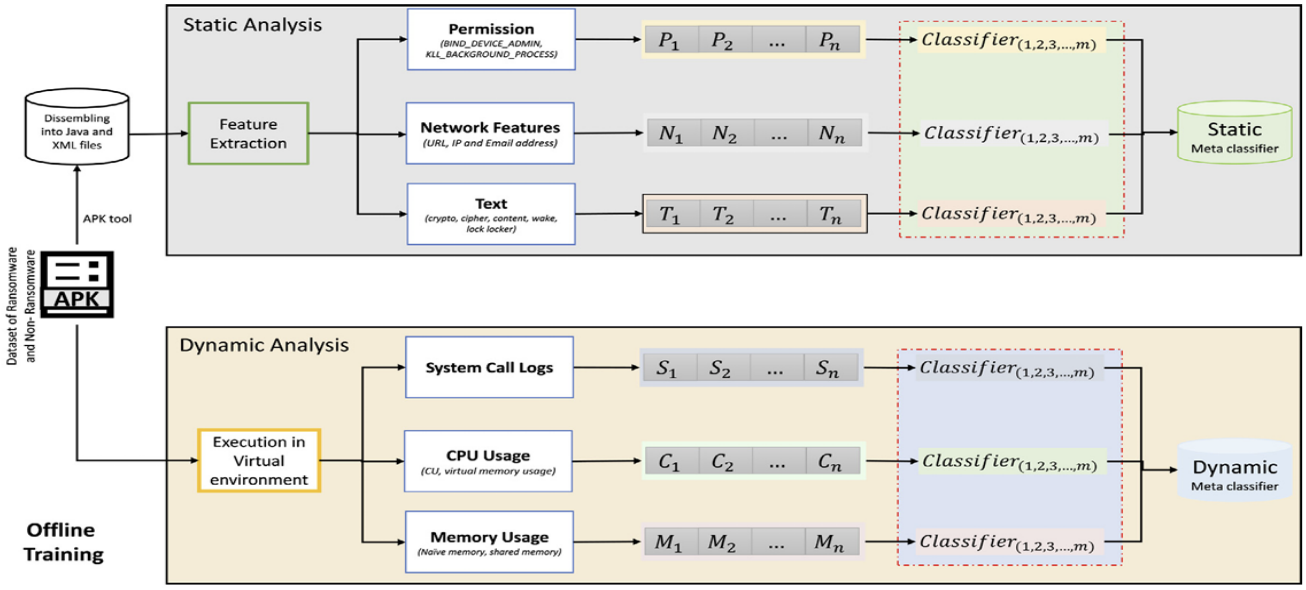
Ransomware tiếp tục gây ra mối đe dọa đáng kể đối với an ninh mạng bằng cách tống tiền người dùng bằng cách khóa thiết bị và dữ liệu cá nhân của họ. Những kẻ tấn công buộc phải trả tiền chuộc để khôi phục quyền truy cập vào các tệp cá nhân. Do sự tương đồng về cấu trúc, việc phát hiện ransomware và các ứng dụng lành tính trở nên dễ bị tấn công trốn tránh (evasion attack). Học tập kết hợp (ensemble learning) có thể cung cấp các biện pháp đối phó, trong khi những kẻ tấn công có thể sử dụng cùng một kỹ thuật để cải thiện hiệu quả của các cuộc tấn công tương tự của chúng. Điều này thúc đẩy việc điều tra xem liệu những phương pháp kết hợp riêng biệt có thể đạt được hiệu suất tốt hơn khi kết hợp với phương pháp dựa trên bỏ phiếu hay không. Nghiên cứu được đề cập trong bài báo này đề xuất một cách tiếp cận lai kiểm tra quyền, văn bản và các tính năng dựa trên mạng cả tĩnh và động bằng cách giám sát việc sử dụng bộ nhớ, nhật ký cuộc gọi hệ thống và sử dụng CPU. Các máy phân tích tập hợp trên các tính năng tĩnh và động được trích xuất từ các ứng dụng phần mềm độc hại Android (ransomware và không ransomware) sau đó được đào tạo theo mô hình được thiết kế. Kết quả thử nghiệm của bài báo cho thấy kỹ thuật phát hiện và phân loại tập thể được đề xuất có thể phân loại hành vi ransomware tĩnh và động không xác định để giảm thiểu các cuộc tấn công trốn tránh đối thủ (adversarial evasion attack).

Nghiên cứu này tập trung vào việc giảm thiểu các cuộc tấn công trốn tránh trong phát hiện ransomware Android, chẳng hạn như xáo trộn mã (obfuscation) được sử dụng để tránh việc bị phát hiện là phần mềm độc hại hoặc ransomware. Hầu hết các kỹ thuật ransomware hiện tại không sửa đổi vectơ tính năng đầu vào để phân tích (khi một khía cạnh được sử dụng để xáo trộn, nó sẽ thay đổi toàn bộ vectơ tính năng, khiến trình phân loại được đào tạo phân loại sai ransomware. Do đó, nghiên cứu này đề xuất một cơ chế phân tích dựa trên tổng hợp để phát hiện ransomware Android và giảm thiểu các cuộc tấn công trốn tránh. Các đặc điểm Android được sử dụng để phân tích ở đây không dễ sửa đổi và sử dụng cho các nỗ lực trốn tránh. Dựa trên các đặc tính này, tác giả đề xuất một kỹ thuật

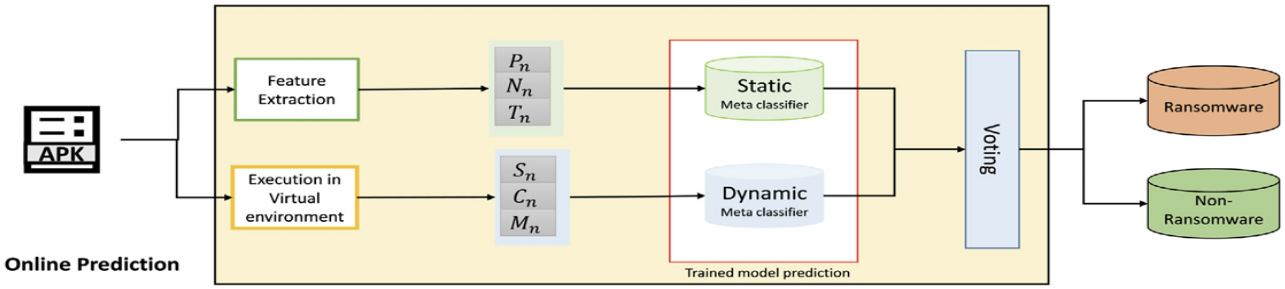
kết hợp hiệu quả của cả tính năng tĩnh (ví dụ: quyền, văn bản, tính năng dựa trên mạng, v.v.) và các tính năng động (như nhật ký cuộc gọi hệ thống, CPU và sử dụng bộ nhớ) để phát hiện phần mềm tống tiền Android bằng cách sử dụng mô hình máy học tổng hợp.

II. Sơ lược về phương pháp phân tích tổng hợp (ensemble analyze)

Mô hình của phương pháp phân tích tổng hợp sẽ bao gồm 2 phần: offline training và online prediction. Quá trình offline training bao gồm các công đoạn như trích xuất đặc trưng, chọn lựa đặc trưng, huấn luyện và kiểm thử hiệu suất của mô hình. Còn trong quá trình online prediction, bộ phân loại đã được huấn luyện trước đó sẽ được sử dụng để phân loại các ứng dụng độc hại và lành tính. Việc trích xuất đặc trưng dựa trên phân tích tĩnh và động, ở quá trình offline training, phân tích tĩnh sẽ được sử dụng để trích xuất quyền hạn, mạng và các văn bản chứa trong tập tin APK. Với phân tích



(a) Offline features extraction, training and ensemble method.



(b) Online prediction of the trrain method.

Fig. 1. The overall methodology of distinct ensemble analysis approach.

động, ứng dụng sẽ được chạy trong môi trường ảo được kiểm soát, sau đó các đặc trưng sẽ được trích xuất. Các thông tin của hệ thống như system call, tài nguyên CPU, memory được ghi lại, sau đó trích xuất và tiến hành phân tích.

1. Phân tích tĩnh

Dựa theo yêu cầu bài báo, tập dữ liệu được phân tích sẽ bao gồm các file APK, trong đó 50%

ransomware và 50% không phải ransomware). Android Package Kit (APK) là định dạng tệp mà Android sử dụng để phân phối và cài đặt ứng dụng. Nó chứa tất cả các thành phần như các tệp classes (.dex), tài nguyên và tệp manifest mà một ứng dụng cần để được cài đặt đúng cách trên thiết bị. Tệp manifest chứa các quyền (permission) và các chi tiết cấu hình khác của ứng dụng. Quá trình trích xuất đặc trưng bắt đầu bằng việc nắm bắt các tệp APK bằng cách sử dụng một script trích xuất đặc trưng. Chúng em đã viết một script Python để trích xuất quyền từ tệp manifest.xml, các đặc trưng văn bản và mạng (tức là các địa chỉ IP, địa chỉ email và URL) từ các tệp .dex. Script này giải mã các tệp APK, trích xuất các đặc trưng này và sau đó lưu chúng vào các tệp văn bản, sử dụng các tệp .txt này từ cả các ứng dụng ransomware và không phải ransomware để tạo ra các vector đặc trưng. Script tạo vector đặc trưng đọc các tệp .txt từ cả các ứng dụng ransomware và không phải ransomware. Nó tạo ra các vector đặc trưng của mỗi ứng dụng sau khi thu thập một tập dữ liệu của tất cả các đặc trưng để xác định các đặc trưng đặc trưng của mỗi ứng dụng và lưu tập dữ liệu này vào tệp đầu ra.

Tạo các chuỗi nhị phân cho mỗi file trong tập dữ liệu (đại diện cho vector đặc trưng). Tất cả các quyền được phát hiện riêng lẻ sau đó được sắp xếp dưới dạng một chuỗi các số 1 và 0. Một quyền cụ thể được biểu thị bằng số 1 và nếu không có quyền tương ứng sẽ được biểu thị bằng số 0 trong danh sách. Bit cuối cùng của vector đại diện cho loại ứng dụng (tức là ransomware hoặc không phải ransomware). Tất cả các quyền dư thừa đều bị loại bỏ khỏi tập dữ liệu, vì sự dư thừa có thể có tác động tiêu cực đến phân loại. Sau khi loại bỏ các quyền dư thừa, chúng em thu được số quyền duy nhất.

Cả 2 đặc trưng văn bản (text) và mạng (network) đều chứa các chuỗi; do đó, chúng ta tạo vector đặc trưng của chúng bằng cách sử dụng TF-IDF vectorizer. TF-IDF vectorizer chuyển đổi các đặc trưng văn bản thành các vector đặc trưng có thể được sử dụng làm đầu vào cho thuật toán phân loại. TF-IDF là một đại diện tuyệt vời và ngày càng hiệu quả cho các cơ chế phân loại dữ liệu văn bản. Tiếp theo, chúng ta sử dụng tất cả các vector đặc trưng tĩnh đã tạo để huấn luyện các bộ phân tích tĩnh dựa trên học máy. Thuật toán 1 mô tả quá trình trích xuất đặc trưng cho cả phân tích tĩnh và động của các tệp APK, chuyển đổi các đặc trưng đã trích xuất thành các vector đặc trưng và cơ chế phân loại được sử dụng để phát hiện ransomware Android. Tóm tắt các bước làm được mô tả như hình ở dưới đây

Algorithm 1 Feature extraction and classification detection

INPUT: APK_{File}.
OUTPUT: Malware or Ransomware.

for *all* $f \in F$ **do** ▷ F is APK folde
  $APK_{File} \leftarrow Open(file)$;
  $manifest_{File}, java_{File} \leftarrow APK\_Tool(APK_{File})$;
  if $manifest_{File} == androidmanifest.xml$ **then**
    permission$\leftarrow Get_Permission(androidmanifest.xml)$;
    for *all* $permission_{(i)} \in permission$ **do**
      if $Permission_{(list)}[i] == permission_{(i)}$ **then**
        $Vector_{(Permission)}[\,] \leftarrow 1$;
      end if
      $Vector_{(Permission)}[\,] \leftarrow 0$;
    end for
  end if
  $network_{vector} \leftarrow TF\_IDF(manifest_{File}, java_{File}, network_{File})$;
  $Text_{vector} \leftarrow TF\_IDF(manifest_{File}, java_{File}, Text_{File})$;
  $Dynamic_{vector} \leftarrow Virtual\_Environment(APK_{File})$;
  $Output_1 \leftarrow Classify(network_{vector} + Text_{vector} + Vector_{Permission})$;
  $Output_2 \leftarrow Classify(Dynamic_{vector})$;
  $Output \leftarrow XOR(Output_1, Output_2)$;
end for
Return *Output*.

2. Phân tích động

Phân tích động bao gồm việc chạy ứng dụng Android trên môi trường ảo để kiểm tra các hành vi của ứng dụng theo thời gian thực. Phân tích động được sử dụng để phát hiện các hành vi độc hại của ứng dụng mà ta không thể phát hiện chúng dựa vào phân tích tĩnh. Như có đề cập ở trên, mục đích của việc phân tích động là để trích xuất ra được các đặc trưng "động" (Lượng CPU sử dụng, thống kê system call, lượng memory sử dụng,…) , những thông tin này rất hữu ích cho việc phát hiện ransomware. Do đó, các dấu vết thực thi có chứa dữ liệu này phải được thu thập bằng cách chạy ứng dụng trong môi trường được kiểm soát. Những dấu vết này được ghi lại thủ công bằng cách chạy từng ứng dụng một trong 10 phút trong trình giả lập Android. Tuy nhiên, một số dấu vết ngắn hơn vì trình giả lập có những điểm yếu nhỏ. Tuy nhiên, thời gian thực hiện dài hơn mang lại cho chúng ta kết quả có ý nghĩa hơn. Trình giả lập Android Genymotion được chọn để phân tích động các ứng dụng phần mềm độc hại Android. Công cụ Genymotion được sử dụng vì nó là phần mềm mã nguồn mở và hỗ trợ Android Studio. Lý do sử dụng phần mềm giả lập Android thay vì thiết bị gốc là môi trường mô phỏng cung cấp nhiều dung lượng hơn để chạy một số lượng lớn các chương trình phần mềm độc hại trong một thời gian hợp lý. Phân tích động trên thiết bị miễn nhiễm với các kỹ thuật bỏ qua trình giả lập. Tuy nhiên, trong trường hợp ransomware Android, thiết bị vật lý không thể được đặt lại về trạng thái sạch. Ngược lại, trình giả lập có thể được khởi tạo lại sau khi phân tích từng ứng dụng. Tuy nhiên, việc chống mô phỏng có thể có một số tác động đến việc trích xuất tính năng động được thực hiện trên Genymotion. Do đó, trong mô hình đề xuất của chúng tôi, khi sử dụng các tính năng động (sử dụng bộ nhớ và CPU), chúng tôi giả định rằng ngay cả khi các ứng dụng được đóng gói lại, làm xáo trộn hoặc được trang bị các kỹ thuật để tránh lỗi phát hiện, chúng vẫn sẽ hiển thị các dấu vết hành

vi tương tự trong quá trình thực thi. Một trình phân tích tổng hợp được đào tạo về các đặc điểm này có thể phân biệt đầy đủ các ứng dụng ransomware Android với các ứng dụng phần mềm độc hại khác. Thiết bị ảo được khởi tạo lại mỗi lần trước khi một ứng dụng độc hại mới được thực thi để tránh sự can thiệp từ các ứng dụng đã thực thi trước đó, chẳng hạn như thay đổi cài đặt, thực thi các quy trình nền, thay đổi liên quan đến cấu hình hệ điều hành, v.v. Android Debug Bridge (ADB) được sử dụng để giám sát việc sử dụng bộ nhớ và CPU của các ứng dụng. ADB là một công cụ dòng lệnh cho phép PC giao tiếp với phiên bản giả lập hoặc thiết bị Android. Strace (một công cụ theo dõi cuộc gọi hệ thống) được sử dụng để thu thập các cuộc gọi hệ thống từ các ứng dụng. Để trích xuất tính năng động, các bước sau được thực hiện cho từng ứng dụng như được đề cập trong đoạn mã giả dưới đây.

Algorithm 2 Controlled environment feature extraction process.

INPUT: APK_{File}.
OUTPUT: Dynamic features.

$Start_{Device} \leftarrow AVM(genymotion)$;
for *all* $f \in F$ **do** ▷ F is APK folder
  $Package \leftarrow APK(f)$;
  $Events - Execution \leftarrow Apply(wipes, presses, touchscreens)$;
  $Memory \leftarrow ADB(meminfo)$;
  $CPU \leftarrow ADB(cpuinfo)$;
  $Processes \leftarrow PID()$;
  $System - call \leftarrow Command(strace --p\ pid)$;
  $Terminate\ (f,\ 10minutes)$;
  $Exit(f)$;
end for
$Feature\_set \leftarrow Package,\ Memory, CPU, Processes, System - call$;
Return $Feature\_set$.

III. Demo

1. Phân tích tĩnh

Công cụ được sử dụng cho việc phân tích tĩnh của nhóm là *Apktool 2.7.0*. Về ý tưởng, sau khi decompile file apk xong sẽ đi phân tích và trích xuất tất cả các đặc trưng ra từ các thư mục đã decompile, các đặc trưng về permission sẽ có trong file AndroidManifest.xml, còn các đặc trưng text và network sẽ sử dụng lệnh *grep* để tìm kiếm các từ khóa liên quan đến mã hóa hoặc khóa trong các tệp ở một thư mục đã được decompile bằng *Apktool*.

```
def decompile_apk(apk_path, output_directory):
    output_path = f"/home/kali/NT230/RansomwareAndroid/DecompileApk/{output_directory}"
    subprocess.run(["apktool", "d", apk_path, "-o" , output_path, "-f"], check=True)
```

Trích xuất các đặc trưng permission:

- Sử dụng thư viện *xml.etree.ElementTree* để làm việc với các tệp XML. Tạo một mảng (*permissions*) bao gồm tất cả các permission được tìm thấy.

- Phân tích cú pháp tệp manifest: Sử dụng *ET.parse(manifest_path)* để đọc và phân tích cú pháp của tệp XML.
- Tìm kiếm các quyền: sử dụng *findall('uses-permission')* để tìm tất cả các phần tử quyền trong tệp manifest.
- Trích xuất tên quyền: Lấy tên quyền từ thuộc tính *name* của mỗi phần tử và thêm vào danh sách quyền (*permissions*).

```
def extract_permissions(manifest_path):
    permissions = []
    try:
        manifest = ET.parse(manifest_path)
        root = manifest.getroot()
        for perm in root.findall('uses-permission'):
            permission_name = perm.get('{http://schemas.android.com/apk/res/android}name')
            permissions.append(permission_name)
    except Exception as e:
        print(f"Error extracting permissions from {manifest_path}: {e}")
    return permissions
```

Trích xuất các đặc trưng text:
- Phân tích và trích xuất các đặc trưng text bằng công cụ *grep*, tìm kiếm các văn bản liên quan đến các từ khóa như *crypto|Cipher|cipher|wake|lock|locker* trong tất cả các tệp trong thư mục đã decompile.
- Sau khi tìm tất cả các văn bản liên quan sẽ sử dụng qua hàm ***extract_important_info_text*** để lọc lại những thông tin cần thiết và bỏ qua những thông tin không cần thiết.

```
def extract_text_features(output_directory):
    try:
        output = subprocess.check_output(['grep', '-rE', 'crypto|Cipher|cipher|wake|lock|locker', output_directory])
        return extract_important_info_text(output.decode())  # Trả về chuỗi đầy đủ từ kết quả grep
    except Exception as e:
        print(f"Error extracting text features: {e}")
        return ""  # Trả về chuỗi rỗng nếu có lỗi

def extract_important_info_text(grep_output):
    important_info = []
    lines = grep_output.splitlines()
    for line in lines:
        # Sử dụng biểu thức chính quy để lọc ra phần bạn quan tâm
        match = re.search(r'(?<=:)(.*?);', line)
        if match:
            important_info.append(match.group())
    return important_info
```

Trích xuất các đặc trưng network:
- Sử dụng lệnh *grep* với tùy chọn *-rE* cho phép tìm kiếm đệ quy và sử dụng biểu thức chính quy để tìm các chuỗi mạng trong tất cả các tệp trong thư mục đầu ra.
- Biểu thức chính quy: Sử dụng regrex *r'https?://[^\s]+|[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+|[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'* để tìm các URL, địa chỉ IP và địa chỉ email.
- Sau đó ta tiếp tục xử lí đầu ra để lọc lại những thông tin cần thiết bằng hàm ***extract_important_info***.

```python
def extract_important_info(grep_output):
    important_info = []
    lines = grep_output.splitlines()
    seen = set()
    for line in lines:
        # Tìm các chuỗi URL, địa chỉ IP, email trong dòng
        matches = re.findall(r'https?://[^\s]+|[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+|[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', line)
        for match in matches:
            if match not in seen:
                seen.add(match)
                important_info.append((match))
    return important_info

def extract_network_features(output_directory):
    try:
        grep_command = [
            'grep', '-rE',
            'https?://|[0-9]+\\.[0-9]+\\.[0-9]+\\.[0-9]+|[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}',
            output_directory
        ]
        output = subprocess.run(grep_command, capture_output=True, text=True)
        important_info = extract_important_info(output.stdout)
        return important_info
    except Exception as e:
        print(f"Error extracting network features: {e}")
        return "Loi"
```

Tổng hợp, xử lí file apk và kết quả:

- Tổng hợp các hàm trên lại để tạo một script phân tích file apk, và kiểm tra kết quả.

```python
def process_apk(apk_path):
    try:
        apk_name = os.path.basename(apk_path)  # Lấy tên thư mục từ đường dẫn
        output_directory = apk_name + ".out"
        decompile_apk(apk_path, output_directory)
        manifest_path = os.path.join(output_directory, "AndroidManifest.xml")
        permissions = extract_permissions(manifest_path)
        network = extract_network_features(output_directory)
        text_features = extract_text_features(output_directory)
        return {
            'apk_name': apk_name,
            'permissions': permissions,
            'network': network,
            'text': text_features
        }
    except Exception as e:
        print(f"Error processing APK {apk_path}: {e}")
        return None
```

- Các kết quả được lưu vào file txt và có kết quả như sau:

+ **network:**

```
1 ['http://schemas.android.com/apk/res/android"', 'http://
  schemas.android.com/apk/res/com.rainbow.Thua">', 'http://
  schemas.android.com/apk/res/android">', 'http://schemas.android.com/apk/
  res/com.wwcscug.xiangqi">', 'dinowar@126.com']
2 ['http://schemas.android.com/apk/res/android">', 'http://
  schemas.android.com/apk/res/android"', 'http://www.umeng.com/
  check_config_update"', 'http://www.umeng.co/check_config_update"',
  'http://feedback.whalecloud.com"', 'http://feedback.whalecloud.com/
  mark_viewed"', 'http://feedback.whalecloud.com/feedback/reply"',
  'http://feedback.whalecloud.com/dev_replied"', 'http://
  feedback.whalecloud.com/feedback/feedbacks"', 'http://
  feedback.whalecloud.com/reply"', 'http://feedback.whalecloud.com/
  feedback/userinfo"', 'http://www.umeng.com/app_logs"', 'http://
  www.umeng.co/app_logs"', 'http://www.umeng.com/api/check_app_update"',
  'http://www.umeng.co/api/check_app_update"', '10.0.0.172', 'http://"',
  'http://app.wapx.cn/action/account/getinfo?"', 'http://app.wapx.cn/
  action/account/spend?"', 'http://ads.wapx.cn/action/user_info"',
  'http://app"', 'http://app.wapx.cn"', 'http://ads.wapx.cn"', 'http://
  app.wapx.cn/action/"', 'kingxiaoguang@gmail.com', 'http://app.wapx.cn/
  action/account/award?"', 'http://app.wapx.cn/action/connect/active?"',
  'http://app.wapx.cn/action/app/update?"', 'http://app.wapx.cn/action/
  account/offerlist?"', 'http://app.wapx.cn/action/account/ownslist?"',
  'http://ads.wapx.cn/action/"', 'http://app.wapx.cn/action/feedback/
  form"']
3 ['dev.kingbo@gmail.com', 'http://www.joyuejob.com/apps/MyJob.apk</
  string>', 'http://schemas.android.com/apk/res/android">', 'http://
```

+ text:

```
  WebSettings;']
13 ['    invoke-virtual {v0, v1}, Landroid/webkit/WebSettings;', '    new-
  instance v1, Ljavax/crypto/spec/SecretKeySpec;', '    invoke-direct {v1,
  v2, v3}, Ljavax/crypto/spec/SecretKeySpec;', '    invoke-static {v2},
  Ljavax/crypto/Cipher;', '    invoke-virtual {v2, v4, v1}, Ljavax/crypto/
  Cipher;', '    invoke-virtual {v2, v3}, Ljavax/crypto/Cipher;',
  '    .catch Ljavax/crypto/NoSuchPaddingException;', '    .catch Ljavax/
  crypto/IllegalBlockSizeException;', '    .catch Ljavax/crypto/
  BadPaddingException;', '    new-instance v1, Ljavax/crypto/spec/
  SecretKeySpec;', '    invoke-direct {v1, v2, v3}, Ljavax/crypto/spec/
  SecretKeySpec;', '    invoke-static {v2}, Ljavax/crypto/Cipher;', '
  invoke-virtual {v2, v4, v1}, Ljavax/crypto/Cipher;', '    invoke-virtual
  {v2, v3}, Ljavax/crypto/Cipher;', '    .catch Ljavax/crypto/
  NoSuchPaddingException;', '    .catch Ljavax/crypto/
  IllegalBlockSizeException;', '    .catch Ljavax/crypto/
  BadPaddingException;', '    invoke-virtual {v1, v3}, Landroid/webkit/
  WebSettings;', '    invoke-virtual {v1}, Landroid/os/StatFs;', '
  invoke-virtual {v1}, Landroid/os/StatFs;', '    invoke-virtual {v1},
  Landroid/os/StatFs;', '    invoke-virtual {v1}, Landroid/os/StatFs;',
  '.field private static final EXTRA_WAKE_LOCK_ID:Ljava/lang/String;',
  '.field private static final sPoolWorkQueue:Ljava/util/concurrent/
  BlockingQueue;', '    new-instance v0, Ljava/util/concurrent/
  LinkedBlockingQueue;', '    invoke-direct {v0, v1}, Ljava/util/
```

+ permisson:

```
1 android.permission.SYSTEM_ALERT_WINDOW,
  android.permission.ACCESS_NETWORK_STATE,
  android.permission.WRITE_EXTERNAL_STORAGE, android.permission.INTERNET,
  android.permission.READ_PHONE_STATE, android.permission.GET_TASKS,
  android.permission.INTERNET, android.permission.WAKE_LOCK,
  android.permission.READ_PHONE_STATE,
  android.permission.ACCESS_NETWORK_STATE,
  android.permission.WRITE_EXTERNAL_STORAGE,
  android.permission.ACCESS_WIFI_STATE,
  android.permission.SYSTEM_ALERT_WINDOW,
  com.android.launcher.permission.INSTALL_SHORTCUT,
  android.permission.GET_TASKS
2 android.permission.ACCESS_NETWORK_STATE, android.permission.INTERNET,
  android.permission.READ_PHONE_STATE, android.permission.READ_LOGS,
  android.permission.WRITE_EXTERNAL_STORAGE, android.permission.GET_TASKS,
  android.permission.ACCESS_WIFI_STATE,
  com.android.launcher.permission.INSTALL_SHORTCUT
3 android.permission.RESTART_PACKAGES,
  android.permission.ACCESS_WIFI_STATE, android.permission.INTERNET,
  android.permission.ACCESS_NETWORK_STATE,
  android.permission.READ_PHONE_STATE,
```

2. Phân tích động

Môi trường thực nghiệm: Genymotion 3.7.1, hệ điều hành Android 6.0.0, kích thước bộ nhớ là 2048 MB

Phân tích động sẽ bao gồm 2 bước:

+ Sử dụng Monkey Runner để tạo các thao tác trên ứng dụng, sau đó tiến hành thu thập các thông tin về system call

+ Chờ một khoảng thời gian là 10 phút, sau đó thu thập dữ liệu về sử dụng memory

Lưu ý: Nhóm cũng có thử trích xuất thông tin về CPU mà ứng dụng sử dụng bằng lệnh *cpuinfo*, tuy nhiên không thu lại được thông tin hữu ích.

Thu thập thông tin system call:

Đầu tiên, ta sử dụng lệnh strace để lắng nghe system call trên ứng dụng cần được theo dõi dựa trên PID, sau đó lưu lại report.

strace -p <PID>-c -o /path /report.csv

Sử dụng Monkey Runner để tạo các thao tác trên ứng dụng

adb shell monkey -p <package_name> -v 500 -s 42

Các thông tin về system call sẽ có dạng như sau:

% time	seconds	usecs/call	calls	errors	syscall
36.12	0.338569	181	1871	425	recvfrom
26.68	0.250134	11	23699		clock_gettime
11.19	0.104889	89	1172		epoll_pwait
8.13	0.076204	87	880		sendto
4.94	0.046277	49	951	4	ioctl
4.26	0.039906	30	1311		write
3.36	0.031481	46	688	47	futex
1.65	0.015468	36	429		writev
0.90	0.008449	8	1070		getuid32
0.83	0.007806	7	1102		read
0.51	0.004751	51	94		madvise
0.22	0.002056	12	166		mmap2
0.21	0.001979	19	103		fstat64
0.20	0.001836	42	44	4	fstatat64
0.17	0.001592	6	260		rt_sigprocmask
0.14	0.001305	9	148		timerfd_settime
0.12	0.001110	50	22		mprotect
0.12	0.001085	7	148		pread64
0.09	0.000808	40	20		clone
0.07	0.000652	2	297		gettimeofday
0.06	0.000537	8	71	4	openat
0.04	0.000331	4	92		close
0.02	0.000207	3	81		munmap
0.00	0.000000	0	13		dup
0.00	0.000000	0	21		prctl
0.00	0.000000	0	7		fcntl64
0.00	0.000000	0	8		epoll_ctl
0.00	0.000000	0	18		faccessat
0.00	0.000000	0	2	2	getsockopt
100.00	0.937432		34788	486	total

Thu thập thông tin memory:

Sau khi chờ đợi 10 phút, ta sử dụng lệnh *meminfo* để thu thập dữ liệu sử dụng memory của ứng dụng ta theo dõi

adb shell dumpsys meminfo <package_name>

Thông tin về memory sẽ có dạng như sau:

```
** MEMINFO in pid 7463 [com.ADASiteMap] **
                   Pss  Private  Private  SwapPss     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     5137     5080        0        0    12800    10632     2167
  Dalvik Heap     3582     3480        0        0     8844     5307     3537
Dalvik Other       446      444        0        0
       Stack        52       52        0        0
      Ashmem      7104     7104        0        0
   Other dev         6        0        4        0
    .so mmap      2314      136      128        0
    .apk mmap      552        0        0        0
    .ttf mmap      129        0        0        0
    .dex mmap      764        4      760        0
    .oat mmap     4222        0      136        0
    .art mmap     1355      944        0        0
   Other mmap     3078        4     1908        0
      Unknown      357      340        0        0
        TOTAL    29098    17588     2936        0    21644    15939     5704

App Summary
                       Pss(KB)
                        ------
           Java Heap:     4424
         Native Heap:     5080
                Code:     1164
               Stack:       52
            Graphics:        0
       Private Other:     9804
              System:     8574

               TOTAL:    29098       TOTAL SWAP PSS:        0
```

3. Vận dụng mô hình tổ hợp

Xử lí dữ liệu:

- Chuyển đổi các đặc trưng thành vector để đưa vào mô hình huấn luyện và thử nghiệm.

+ ***Các đặc trưng tĩnh:***

Permission:

- Đọc danh sách quyền từ tệp: Mỗi dòng của tệp chứa các quyền của một ứng dụng.
- Tạo danh sách quyền duy nhất (unique permission): Sử dụng regex để tìm các quyền trong danh sách quyền của cả ransomware và non-ransomware, sau đó loại bỏ các quyền trùng lặp để tạo danh sách quyền duy nhất.
- Tạo vector nhị phân: Với mỗi ứng dụng, tạo một vector nhị phân dựa trên danh sách quyền duy nhất và thông tin về việc ứng dụng đó có phải là ransomware hay không.

```python
# Hàm tạo vector nhị phân từ danh sách quyền của mỗi ứng dụng
def create_binary_permission_vector(permissions, unique_permissions, is_ransomware):
    num_unique_permissions = len(unique_permissions)
    binary_vector = np.zeros(num_unique_permissions + 1, dtSype=int)  # Tạo vector nhị phân với thêm một bit cuối cùng
    for perm in permissions:
        if perm in unique_permissions:
            index = unique_permissions.index(perm)  # Xác định chỉ mục của quyền trong danh sách duy nhất
            binary_vector[index] = 1  # Đánh dấu 1 cho quyền này trong vector
    binary_vector[-1] = 1 if is_ransomware else 0  # Đặt bit cuối cùng thành 1 nếu là ransomware, ngược lại là 0
    return binary_vector
```

- Kết hợp các vector nhị phân của cả ransomware và non-ransomware thành một ma trận.
- Xuất ra file CSV, ghi các vector nhị phân vào file CSV để sử dụng trong các bước phân tích và mô hình hóa tiếp theo

```
┌──(kali㉿kali)-[~/NT230/RansomwareAndroid/feature vectors]
└─$ python3 xem.py
     android.permission.FOREGROUND_SERVICE_DATA_SYNC  android.permission.GET_PACKAGE_SIZE  android.permission.USE_FULL_SCREEN_INTENT  ...  android.permission.MANAGE_ACCOUNTS  android.permission.SEND_SMS  Label
0                                                  0                                   0                                         0  ...                                   0                            0      1
1                                                  0                                   0                                         0  ...                                   0                            0      1
2                                                  0                                   0                                         0  ...                                   0                            0      1
3                                                  0                                   0                                         0  ...                                   0                            0      1
4                                                  0                                   0                                         0  ...                                   0                            0      1
..                                               ...                                 ...                                       ...  ...                                 ...                          ...    ...
204                                                0                                   0                                         0  ...                                   0                            0      0
205                                                0                                   0                                         0  ...                                   0                            0      0
206                                                0                                   0                                         0  ...                                   0                            0      0
207                                                0                                   0                                         0  ...                                   0                            0      0
208                                                0                                   0                                         0  ...                                   0                            0      0

[209 rows x 121 columns]
```

- Kết quả có được 121 unique permission và có tổng cộng 209 file apk.

Network và Text :

- Sử dụng TF-IDF vectorizer chuyển đổi các đặc trưng văn bản thành các vector đặc trưng.
- Đọc nội dung từ file sau khi đi phân tích tĩnh và tiến hành chuyển đổi sang vector bằng TF-IDF.
- Điều chỉnh thêm hàm *custom_tokenizer(text)* sử dụng regex để lọc lại các email, địa chỉ IP, URL và các từ khác từ văn bản đầu vào.

```python
def custom_tokenizer(text):
    email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    ip_pattern = r'\b(?:\d{1,3}\.){3}\d{1,3}\b'
    url_pattern = r'https?://\S+\b'
    pattern = re.compile(f'({email_pattern})|({ip_pattern})|({url_pattern})|\w+')
    tokens = pattern.findall(text)
    tokens = [token for sublist in tokens for token in sublist if token != '']
    return tokens
```

- Sử dụng ***TfidfVectorizer*** để tính toán TF-IDF cho dữ liệu văn bản và dư liệu mạng với tokenizer tùy chỉnh để tính toán TF-IDF.
- In ra danh sách các từ (vocabulary) và các giá trị IDF của chúng cho cả dữ liệu văn bản và mạng.

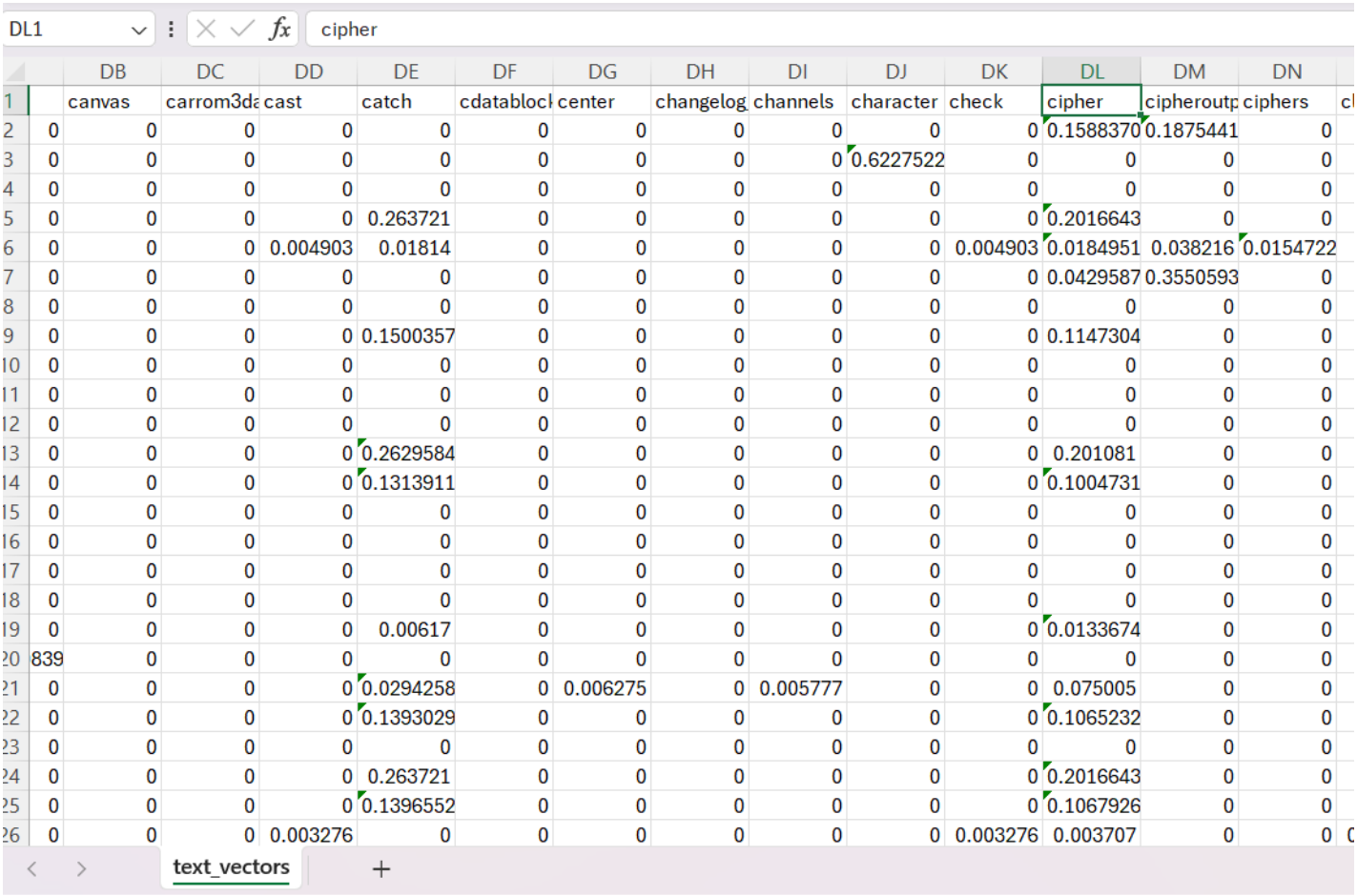
```
# Tính TF-IDF cho các đặc trưng text
vectorizer_text = TfidfVectorizer()
X_text = vectorizer_text.fit_transform(lines_text)
# Tính TF-IDF cho các đặc trưng network
vectorizer_network = TfidfVectorizer(tokenizer=custom_tokenizer)
X_network = vectorizer_network.fit_transform(lines_net)


print("TF-IDF Vocabulary (Text):", vectorizer_text.get_feature_names_out())
print("TF-IDF Matrix (Text) IDF_:", vectorizer_text.idf_)
print("TF-IDF Vocabulary (Network):", vectorizer_network.get_feature_names_out())
print("TF-IDF Matrix (Network) IDF_:", vectorizer_network.idf_)
```

- Chuyển đổi ma trận TF-IDF của văn bản và mạng thành DataFrame và xuất DataFrame thành các file CSV **vectors_text.csv** và **vectors_network.csv**.

Kết quả các vector network:

```
┌──(kali㉿kali)-[~/NT230/RansomwareAndroid/feature vectors]
└─$ python3 xem.py
0.0.0.0                 0.0
1.11.11.10              0.0
10.0.0.172              0.0
10.0.0.200              0.0
10.75.0.103             0.0
                       ...
uff1alymean@sina.cn     0.0
uff1aznzzzqzdd@163.com  0.0
xmmarmy007@gmail.com    0.0
xxx@chatroom.com        0.0
zhouzhanzhong@126.com   0.0
Name: 0, Length: 1396, dtype: float64
     0.0.0.0  1.11.11.10  10.0.0.172  ...  xmmarmy007@gmail.com  xxx@chatroom.com  zhouzhanzhong@126.com
0        0.0         0.0    0.000000  ...                   0.0               0.0                    0.0
1        0.0         0.0    0.084659  ...                   0.0               0.0                    0.0
2        0.0         0.0    0.121491  ...                   0.0               0.0                    0.0
3        0.0         0.0    0.000000  ...                   0.0               0.0                    0.0
4        0.0         0.0    0.046393  ...                   0.0               0.0                    0.0
..       ...         ...         ...  ...                   ...               ...                    ...
116      0.0         0.0    0.038233  ...                   0.0               0.0                    0.0
117      0.0         0.0    0.000000  ...                   0.0               0.0                    0.0
118      0.0         0.0    0.076754  ...                   0.0               0.0                    0.0
119      0.0         0.0    0.046915  ...                   0.0               0.0                    0.0
120      0.0         0.0    0.097973  ...                   0.0               0.0                    0.0

[121 rows x 1396 columns]
```

Kết quả các vector text:



DL1 | cipher

	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN
1		canvas	carrom3da	cast	catch	cdatabloc	center	changelog	channels	character	check	cipher	cipheroutp	ciphers
2	0	0	0	0	0	0	0	0	0	0	0	0.1588370	0.1875441	0
3	0	0	0	0	0	0	0	0	0	0.6227522	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.263721	0	0	0	0	0	0	0.2016643	0	0
6	0	0	0	0.004903	0.01814	0	0	0	0	0	0.004903	0.0184951	0.038216	0.0154722
7	0	0	0	0	0	0	0	0	0	0	0	0.0429587	0.3550593	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0.1500357	0	0	0	0	0	0	0.1147304	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0.2629584	0	0	0	0	0	0	0.201081	0	0
14	0	0	0	0	0.1313911	0	0	0	0	0	0	0.1004731	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0.00617	0	0	0	0	0	0	0.0133674	0	0
20	839	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0.0294258	0	0.006275	0	0.005777	0	0	0.075005	0	0
22	0	0	0	0	0.1393029	0	0	0	0	0	0	0.1065232	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0.263721	0	0	0	0	0	0	0.2016643	0	0
25	0	0	0	0	0.1396552	0	0	0	0	0	0	0.1067926	0	0
26	0	0	0	0.003276	0	0	0	0	0	0	0.003276	0.003707	0	0

text_vectors

+ Các đặc trưng động:

Các đặc trưng động nhóm chúng em phân tích sẽ bao gồm 2 phần là memory và systemcall

Memory:

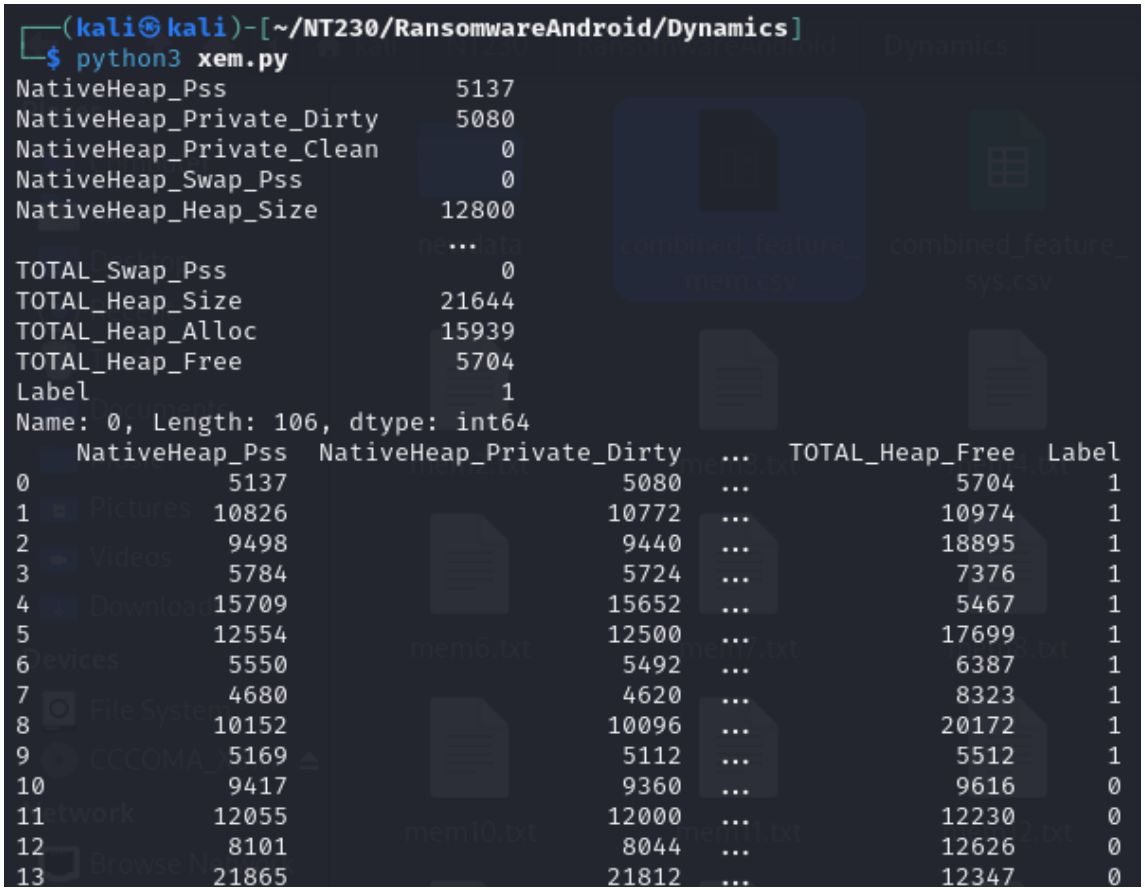
- Đọc và xử lý từng tệp văn bản sau khi đã trích xuất ra.
- Sử dụng ***pattern.findall(content)***: Tìm tất cả các kết quả cần thiết trong nội dung của tệp.
- Các nội dung ở đây bao gồm các thuật ngữ PSS, Private Dirty, Private Clean, Swap PSS, Heap Size, Heap Alloc và Heap Free liên quan đến việc đo lường và theo dõi sử dụng bộ nhớ trong hệ thống máy tính, đặc biệt là trong môi trường Android.
- Ví dụ file văn bản sau khi trích xuất đặc trưng động:

	Pss Total	Private Dirty	Private Clean	SwapPss Dirty	Heap Size	Heap Alloc	Heap Free
Native Heap	5137	5080	0	0	12800	10632	2167
Dalvik Heap	3582	3480	0	0	8844	5307	3537
Dalvik Other	446	444	0	0			
Stack	52	52	0	0			
Ashmem	7104	7104	0	0			
Other dev	6	0	4	0			
.so mmap	2314	136	128	0			
.apk mmap	552	0	0	0			
.ttf mmap	129	0	0	0			
.dex mmap	764	4	760	0			
.oat mmap	4222	0	136	0			
.art mmap	1355	944	0	0			
Other mmap	3078	4	1908	0			
Unknown	357	340	0	0			

-Xử lí dữ liệu:

```
    # Regular expression to match the lines of interest
    pattern = re.compile(r'(Native Heap|Dalvik Heap|Dalvik Other|Stack|Ashmem|Other dev|\.so mmap|\.apk
mmap|\.ttf mmap|\.dex mmap|\.oat mmap|\.art mmap|Other mmap|Unknown|TOTAL)\s+'r'(\d+)\s+(\d+)\s+(\d+)
\s+(\d+)\s*(\d*)\s*(\d*)\s*(\d*)')

    # Find all matches in the file content
    matches = pattern.findall(content)

    # Dictionary to hold data for this file
    csv_data = {}

    # Extract the data and handle missing values
    for entry in matches:
        name, pss, private_dirty, private_clean, swap_pss, heap_size, heap_alloc, heap_free = entry
        key_prefix = name.replace(" ", "").replace(".", "")
        csv_data[f'{key_prefix}_Pss'] = int(pss)
        csv_data[f'{key_prefix}_Private_Dirty'] = int(private_dirty)
        csv_data[f'{key_prefix}_Private_Clean'] = int(private_clean)
        csv_data[f'{key_prefix}_Swap_Pss'] = int(swap_pss)
        csv_data[f'{key_prefix}_Heap_Size'] = int(heap_size) if heap_size else 0
        csv_data[f'{key_prefix}_Heap_Alloc'] = int(heap_alloc) if heap_alloc else 0
        csv_data[f'{key_prefix}_Heap_Free'] = int(heap_free) if heap_free else 0

    # Append the data for this file to the list
    all_data.append(csv_data)
```

- Gán nhãn cho các tệp để phân biệt ransomware hay non-ransomware
- Kết quả:



```
┌──(kali㉿kali)-[~/NT230/RansomwareAndroid/Dynamics]
└─$ python3 xem.py
NativeHeap_Pss                5137
NativeHeap_Private_Dirty      5080
NativeHeap_Private_Clean         0
NativeHeap_Swap_Pss              0
NativeHeap_Heap_Size         12800
                               ...
TOTAL_Swap_Pss                   0
TOTAL_Heap_Size              21644
TOTAL_Heap_Alloc             15939
TOTAL_Heap_Free               5704
Label                            1
Name: 0, Length: 106, dtype: int64
    NativeHeap_Pss  NativeHeap_Private_Dirty  ...  TOTAL_Heap_Free  Label
0             5137                     5080  ...             5704      1
1            10826                    10772  ...            10974      1
2             9498                     9440  ...            18895      1
3             5784                     5724  ...             7376      1
4            15709                    15652  ...             5467      1
5            12554                    12500  ...            17699      1
6             5550                     5492  ...             6387      1
7             4680                     4620  ...             8323      1
8            10152                    10096  ...            20172      1
9             5169                     5112  ...             5512      1
10            9417                     9360  ...             9616      0
11           12055                    12000  ...            12230      0
12            8101                     8044  ...            12626      0
13           21865                    21812  ...            12347      0
```

System call:

- Tương tự với memory, đọc và xử lý từng tệp văn bản sau khi đã trích xuất ra.
- Sử dụng ***pattern.findall(content***): Tìm tất cả các kết quả cần thiết trong nội dung của tệp.

```
pattern = r'(\d+(?:\.\d+)?)\s+(\d+(?:\.\d+)?)\s+(\d+)\s+(\d+)\s+(\d*)\s*(\S+)'

matches = re.findall(pattern, content)
print(matches)
csv_data = {}


# Extract the data and handle missing values
for entry in matches:
    time, seconds, usecs_call, calls, errors, name = entry
    key_prefix = name.replace(" ", "").replace(".", "")
    csv_data[f'{key_prefix}_time'] = float(time)
    csv_data[f'{key_prefix}_seconds'] = float(seconds)
    csv_data[f'{key_prefix}_usecs_call'] = float(usecs_call)
    csv_data[f'{key_prefix}_calls'] = float(calls)
    csv_data[f'{key_prefix}_error'] = float(errors) if errors else 0


all_data.append(csv_data)
```

- Giải thích ***pattern.findall(content):*** lấy tất cả các thông tin bao gồm số và chữ từng dòng và sắp xếp lại, các thông tin cách nhau bằng khoảng trống
- Ví dụ file văn bản sau khi trích xuất đặc trưng động:

```
 1 % time     seconds  usecs/call     calls    errors syscall
 2 ------ ----------- ----------- --------- --------- ----------------
 3  36.12    0.338569         181      1871       425 recvfrom
 4  26.68    0.250134          11     23699           clock_gettime
 5  11.19    0.104889          89      1172           epoll_pwait
 6   8.13    0.076204          87       880           sendto
 7   4.94    0.046277          49       951         4 ioctl
 8   4.26    0.039906          30      1311           write
 9   3.36    0.031481          46       688        47 futex
10   1.65    0.015468          36       429           writev
11   0.90    0.008449           8      1070           getuid32
12   0.83    0.007806           7      1102           read
13   0.51    0.004751          51        94           madvise
14   0.22    0.002056          12       166           mmap2
```

- Gán nhãn và xuất file csv.
- Kết quả:

```
┌──(kali㉿kali)-[~/NT230/RansomwareAndroid/Dynamics]
└─$ python3 xem.py
recvfrom_time                36.120000
recvfrom_seconds              0.338569
recvfrom_usecs_call         181.000000
recvfrom_calls             1871.000000
recvfrom_error              425.000000
                               ...
rt_sigreturn_seconds              NaN
rt_sigreturn_usecs_call           NaN
rt_sigreturn_calls                NaN
rt_sigreturn_error                NaN
Label                        1.000000
Name: 0, Length: 206, dtype: float64
    recvfrom_time  recvfrom_seconds  ...  rt_sigreturn_error  Label
0           36.12          0.338569  ...                 NaN      1
1           41.23          0.529471  ...                 NaN      1
2           38.12          0.051206  ...                 NaN      1
3           50.05          2.624311  ...                 NaN      1
4            1.79          0.033055  ...                 NaN      1
5             NaN               NaN  ...                 NaN      1
6           59.94          0.300580  ...                 NaN      1
7           62.72          1.034727  ...                 NaN      1
8             NaN               NaN  ...                 NaN      1
9           40.49          0.280009  ...                 NaN      1
10           1.04          0.004925  ...                 NaN      0
11           0.33          0.001091  ...                 NaN      0
12           7.03          0.027479  ...                 NaN      0
```

Sử dụng phối hợp các mô hình ensemble:

- Các mô hình ensemble:
  - Ensemble 1 : C45 Decision tree, Random forest.
  - Ensemble 2 : Logistic regression, C45, SVM with SMO.
  - Ensemble 3 : Random forest, SVM with SMO.
  - Ensemble 4 : SVM with SMO, Logistic regression, Random forest.
  - Ensemble 5 : SVM with SMO, Logistic regression, AdaBoost with SVM base.
  - Ensemble 6 : Logistic regression, GaussianNB, Random forest, C45, SVM with SMO.
  - Ensemble 7 : SVM with SMO, logistic regression, KNeighbors, AdaBoost with SVM base.
- Thông số đánh giá hiệu suất từng mô hình ensemble:

	Static Feature				**Dynamic Feature**			
Model/Metrics	ACC	Precision	Recall	F1	ACC	Precision	Recall	F1
Ensemble 1	0.91	0.87	0.85	0.86	0.99	0.99	0.98	0.99
Ensemble 2	0.91	0.95	0.93	0.94	0.99	0.99	0.99	0.99
Ensemble 3	0.93	0.99	0.92	0.95	0.99	0.99	0.99	0.99
Ensemble 4	0.98	0.94	0.99	0.96	0.99	0.99	0.97	0.98
Ensemble 5	0.94	0.95	0.96	0.95	0.99	0.99	0.99	0.99
Ensemble 6	0.91	0.99	0.95	0.97	0.99	0.99	0.99	0.99
Ensemble 7	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Việc đưa ra kết quả nhận diện cuối cùng sẽ tiến hành bằng cách voting thông phép toán OR cho kết quả của đặc trưng tĩnh và đặc trưng động, sau đó đưa ra kết quả cuối cùng là ransom hay non-ransom. Thông số đánh giá kết quả cuối cùng biểu thị bởi bảng dưới đây

Model/Metrics	**ACC**	**Precision**	**Recall**	**F1**
Ensemble 1	0.95	0.96	0.92	0.94
Ensemble 2	0.94	0.98	0.99	0.98
Ensemble 3	0.94	0.99	0.99	0.99
Ensemble 4	0.97	0.94	0.97	0.95
Ensemble 5	0.98	0.97	0.96	0.96
Ensemble 6	0.97	0.95	0.98	0.96
Ensemble 7	0.99	0.97	0.98	0.97