Sistem Automat de Clasificare Malware Android

Lucrare de Licență

Crețu Vlad-Sebastian

Coordonator: Dr. Ing. Aciu Răzvan

Universitatea Politehnica Timișoara
Facultatea de Automatică și Calculatoare
2024

Cuprins

1. Introducere

1.1 Context

În prezent, sistemul de operare Android domină piața globală a dispozitivelor mobile, având o cotă de piață de peste 70% din totalul dispozitivelor vândute[7]. Numărul aplicațiilor Android disponibile pe Google Play, cel mai popular magazin online pentru acest sistem, a ajuns la 3.718 milioane în 2023, cu o predicție de 4.200 milioane până în 2025[1]. Această creștere va duce la necesitatea unui sistem scalabil capabil să proceseze și să analizeze zilnic cantități enorme de aplicații, pentru a asigura securitatea și confidențialitatea utilizatorilor platformei.

1.2 Motivație

Studiile disponibile se concentrează pe îmbunătățirea acurateței algoritmilor de predicție automată a malware-ului, lăsând preprocesarea datelor pe un plan secundar. Puține cercetări abordează în mod exhaustiv dezvoltarea unui sistem cap-coadă care să fie atât eficient, cât și scalabil. În contextul unei creșteri a numărului de aplicații Android și a diversificării metodelor de atac, este esențial să se dezvolte soluții integrate care să includă atât preprocesarea datelor, cât și analiza și clasificarea acestora. Această lucrare își propune să umple acest gol, oferind un sistem complet care să poată preprocesa eficient datele, să le analizeze și să le clasifice automat pentru a evita atacurile malware.

1.3 Obiective

Lucrarea se concentrează pe dezvoltarea următoarelor obiective:

1. Dezvoltarea unui sistem automat de clasificare multi-label a fișierelor APK[11] folosind 3 componente principale:

- Modul de preprocesare a datelor realizat folosind limbajul de programare Rust[8], care se ocupă de extragerea permisiunilor și secvenței de cod din fișierele APK.
- Model de clasificare bazat pe algoritmi de învățare automată realizat folosind librăria Tensorflow[2], care va clasifica fișierele APK în una din 5 categorii: *Adware, Banking, Benign, Riskware, SMS*.
- REST API realizat folosind framework-ul axum[9] care va oferi o interfață programatică pentru a trimite fișiere APK spre clasificare și a primi rezultatele.

Vizualizarea predicțiilor se va face printr-o interfață web realizată folosind framework-ul React[10].

2. Afișarea rezultatelor experimentale și a statisticilor relevante pentru a evalua performanța sistemului.

2. Analiza stadiului actual în domeniul problemei

2.1 Clasificare malware

2.2 Metode actuale de detectare

3. Fundamentare teoretică

3.1 Sistemul Android

Android este un sistem de operare open source complet cu cod sursă personalizabil care poate fi portat pe aproape orice dispozitiv, și cu documentație publică disponibilă pentru toată lumea la *source.android.com* pentru dispozitive mobile, condus de Google[6]. Platforma Open System Android (AOSP) este un cod sursă Android disponibil public și modificabil. Oricine poate descărca și modifica AOSP pentru dispozitivul său. AOSP oferă o implementare completă și funcțională a platformei mobile Android care constituie următoarea stivă de software[4] (Figura 3.1):

1. **API-ul Android** - API de nivel înalt public pentru dezvoltarea aplicațiilor third-party

2. **Android Framework** - Un set de classe și interfețe Java precompilate care oferă funcționalități de bază pentru dezvoltarea aplicațiilor. O parte din framework este accesibilă prin Android API, pe când cealaltă este accesibilă doar prin API-ul sistemului.

3. **Aplicații Android** - Aplicații preinstalate sau third-party create folosind Android API.

4. **Aplicațiile Producătorului Dispozitivului** - Aplicație create folosind o combinație de API-ul Android, API-ul sistemului și accesul direct la implementarea Android Framework. Deoarece un producător de dispozitive ar putea accesa direct API-uri instabile din cadrul frameworkului, aceste aplicații trebuie să fie preinstalate pe dispozitiv și pot fi actualizate doar atunci când software-ul sistemului dispozitivului este actualizat.

5. **Aplicații Privilegiate** - Aplicații care au acces la API-ul sistemului și care sunt preinstalate pe dispozitiv.

6. **Servicii de Sistem** - Componnete modulare care oferă acces la hardware-ul dispozitivului.

7. **API-ul Sistemului** - API-ul care oferă acces la serviciile de sistem. Accesibile doar pentru producătorii originali de echipament (OEMs) și marcat cu *@SystemApi* în codul sursă.

8. **Android Runtime** - Un mediu de execuție Java care realizează executarea formatului **Dalvik Executable** (DEX) prin traducrea bytecode-ului aplicației în instrucțiuni specifice procesorului.

9. **Hardware Abstraction Layer (HAL)** - Un strat de abstractizare cu o interfață standard implementată de producătorii de hardware, care permite ca sistemul Android să fie agnostic în privința implementărilor driverelor low-level. Utilizarea unui HAL permite implementarea funcționalităților fără a afecta sau modifica sistemul high-level.

10. **Servicii de Sistem & Daemons** - Daemon-uri native cum ar fi: *init, health, logd, logd, storaged* și librării native cum ar fi: *libc, liblog, libutils, libbinder, libselinux* care interacționează direct cu kernelul și nu depind de HAL.

11. **Kernel Linux** - Partea centrală a sistemului de oprare care comunică cu hardware-ul dispozitivului. Kernel-ul AOSP este impărțit în module hardware-agnostice și module specifice pentru furnizor.

3.2 Aplicații

Aplicațiile Android pot fi scrise folosind Kotlin, limbajul de programare Java și limbajele C++. Instrumentele SDK Android compilează codul împreună cu orice fișiere de date și resurse într-un APK sau un Android App Bundle.[3]

Un pachet Android, care este un fișier de arhivă cu extensia *.apk*, conține fișierele unei aplicații Android necesare la rulare și este fișierul pe care dispozitivele cu Android îl folosesc pentru a instala aplicația.[3]

Un Android App Bundle, care este un fișier de arhivă cu extensia *.aab*, conține fișierele unui proiect de aplicație Android, inclusiv unele metadate suplimentare care nu sunt necesare la rulare. Un AAB este un format de publicare și nu poate fi instalat pe dispozitivele Android. Generarea și semnarea APK-ului sunt amânate pentru o etapă ulterioară.[3]

Fiecare aplicație Android funcționează într-un sandbox de securitate propriu, protejat de următoarele caracteristici de securitate ale Android[3]:

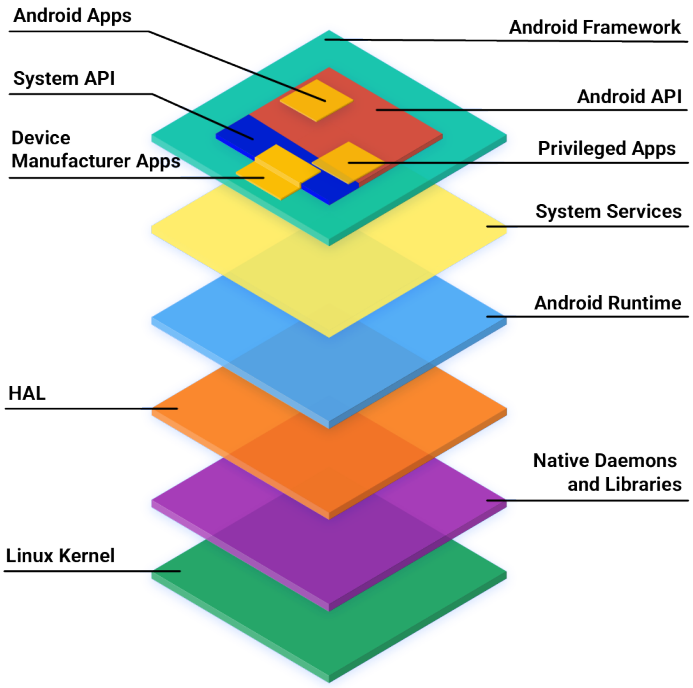

Figure 3.1: AOSP software stack architecture[4]

- Sistemul de operare Android este un sistem Linux multi-utilizator în care fiecare aplicație este un utilizator diferit.
- În mod implicit, sistemul atribuie fiecărei aplicații un ID unic de utilizator Linux, care este utilizat doar de sistem și este necunoscut aplicației. Sistemul setează permisiuni pentru toate fișierele dintr-o aplicație astfel încât doar ID-ul de utilizator atribuit acelei aplicații să le poată accesa.
- Fiecare proces are propria sa mașină virtuală (VM), astfel încât codul unei aplicații rulează izolat de alte aplicații.
- În mod implicit, fiecare aplicație rulează în propriul proces Linux. Sistemul Android pornește procesul atunci când oricare dintre componentele aplicației trebuie să fie executate și închide procesul atunci când nu mai este necesar sau când sistemul trebuie să recupereze memorie pentru alte aplicații.

Sistemul Android implementează principiul privilegiului minim (*principle of least privilege*). Adică, fiecare aplicație, în mod implicit, are acces doar la

componentele de care are nevoie pentru a-și îndeplini funcțiile și nimic mai mult. Acest lucru creează un mediu foarte sigur în care o aplicație nu poate accesa părți ale sistemului pentru care nu i s-au acordat permisiuni.[3]

Cu toate acestea, există modalități prin care o aplicație poate partaja date cu alte aplicații și prin care o aplicație poate accesa servicii de sistem[3]:

- Este posibil să se aranjeze ca două aplicații să partajeze același ID de utilizator Linux, caz în care acestea pot accesa fișierele uneia alteia. Pentru a conserva resursele sistemului, aplicațiile cu același ID de utilizator pot, de asemenea, să aranjeze să ruleze în același proces Linux și să partajeze aceeași mașină virtuală (VM). Aplicațiile trebuie, de asemenea, să fie semnate cu același certificat.

- O aplicație poate solicita permisiunea de a accesa datele dispozitivului, cum ar fi locația dispozitivului, camera și conexiunea Bluetooth. Utilizatorul trebuie să acorde explicit aceste permisiuni.

3.2.1 Permisiuni

Permisiunile aplicațiilor ajută la protejarea confidențialității utilizatorilor[5] prin restricționarea accesului la următoarele:

- Date restricționate, cum ar fi starea sistemului și informațiile de contact ale utilizatorilor

- Acțiuni restricționate, cum ar fi conectarea la un dispozitiv împerecheat și înregistrarea audio

Tipul fiecărei permisiuni indică domeniul datelor restricționate la care aplicația ta poate avea acces și domeniul acțiunilor restricționate pe care aplicația ta le poate efectua atunci când sistemul acordă permisiunea respectivei aplicații.[5] Android categorizează permisiunile în diferite tipuri:

- **Install-time** - Oferă aplicației acces limitat la date restricționate sau permit să efectueze acțiuni restricționate care afectează minim sistemul sau alte aplicații.[5]

  Android include două subtipuri de permisiuni la instalare:

  - **Normale** - Aceste permisiuni permit accesul la date și acțiuni care depășesc sandbox-ul aplicației tale, dar prezintă foarte puțin risc pentru confidențialitatea utilizatorului și funcționarea altor aplicații.[5]

  - **Semnatură (Signature)** - Sistemul acordă o permisiune de semnătură unei aplicații numai atunci când aplicația este semnată cu același certificat ca aplicația sau sistemul de operare care definește permisiunea.[5]

    Aplicațiile care implementează servicii privilegiate, cum ar fi serviciile de completare automată sau VPN, folosesc, de asemenea, permisiuni de semnătură, necesare pentru legarea serviciilor, astfel încât numai sistemul să se poată lega la servicii.[5]

- **Runtime** - Cunoscute sub numele de permisiuni periculoase, oferă aplicației tale acces suplimentar la date restricționate sau îi permit să efectueze acțiuni restricționate care afectează mai substanțial sistemul și alte aplicații.[5]

  Multe permisiuni de rulare accesează date private ale utilizatorilor, un tip special de date restricționate care include informații potențial sensibile. Exemple de date private ale utilizatorilor includ locația și informațiile de contact.[5]

- **Speciale** - Permisiunile speciale corespund unor operațiuni particulare ale aplicațiilor. Doar platforma și producătorii originali de echipamente (OEM) pot defini permisiuni speciale. În plus, platforma și producătorii originali de echipamente definesc de obicei permisiuni speciale atunci când doresc să protejeze accesul la acțiuni deosebit de puternice, cum ar fi afișarea unui UI peste alte aplicații.[5]

3.2.2 Componente

Componentele aplicației sunt blocurile esențiale ale unei aplicații Android, care reprezintă puncte de intrare prin care sistemul sau un utilizator poate accesa aplicația.[3]

Există patru tipuri de componente care pot fi folosite într-o aplicație Android:

- **Activități** - un punct de intrare pentru interacțiunea cu utilizatorul. Reprezintă o singură fereastră cu o interfață de utilizator.[3]

- **Servicii** - un punct de intrare general pentru menținerea aplicației în fundal din diverse motive. Este o componentă care rulează în fundal pentru a efectua operațiuni de lungă durată sau pentru a efectua sarcini pentru procesele la distanță. Un serviciu nu furnizează o interfață de utilizator.[3]

- **Receptoare de difuzare (Broadcast Receivers)** - o componentă care permite sistemului să livreze evenimente aplicației în afara unui flux normal de utilizator, astfel încât aplicația să poată răspunde anunțurilor de difuzare la nivel de sistem. Deoarece receptoarele de difuzare sunt un alt punct de intrare bine definit în aplicație, sistemul poate livra difuzări chiar și aplicațiilor care nu rulează în prezent.[3]

- **Furnizori de conținut (Content Providers)** - o componentă care gestionează un set partajat de date ale aplicației pe care le poți stoca în sistemul de fișiere, într-o bază de date SQLite, pe web sau în orice alt loc de stocare persistentă accesibil aplicației tale. Prin intermediul furnizorului de conținut, alte aplicații pot interoga sau modifica datele, dacă furnizorul de conținut permite acest lucru.[3]

3.2.3 Fișierul manifest

Înainte ca sistemul Android să poată porni o componentă a aplicației, sistemul trebuie să știe că acea componentă există citind fișierul manifest al aplicației, *AndroidManifest.xml*. Aplicația declară toate componentele sale în acest fișier, care se află la rădăcina directorului proiectului aplicației.[3]

Pe lângă aceasta, fișierul manifest răspunde de următoarele:

- Identifică orice permisiuni de utilizator de care are nevoie aplicația, cum ar fi accesul la internet sau accesul de citire la contactele utilizatorului.

- Declară nivelul minim al API-ului necesar aplicației, în funcție de API-urile pe care le folosește aplicația.

- Declară caracteristicile hardware și software utilizate sau necesare aplicației, cum ar fi o cameră, servicii Bluetooth sau un ecran multitouch.

- Declară bibliotecile API de care aplicația are nevoie pentru a fi legate (altele decât API-urile framework-ului Android).

3.3 Android Runtime și Dalvik

3.3.1 Bytecode-ul Dalvik

3.3.2 Formatul DEX

3.3.3 Instrucțiunile Dalvik Executable

4. Soluţia propusă şi metodologia de proiectare/dezvoltare

4.1 Arhitectura sistemului

4.2 Descrierea metodologiei

5. Implementare

5.1 Tehnologii utilizate

5.2 Descrierea implementării

6. Utilizare, rezultate experimentale

6.1 Descrierea procesului de testare

6.2 Rezultate experimentale

7. Concluzii, contribuţii şi direcţii de continuare a dezvoltării

7.1 Concluzii

7.2 Contribuții

7.3 Direcții viitoare

Bibliografie

[1] BankMyCell.com. How many apps in google play store? (2024). https://www.bankmycell.com/blog/number-of-google-play-store-apps/, 2023. Accesat: 18 mai 2024.

[2] Google Brain Google LLC. Tensorflow. https://www.tensorflow.org/, 2024. Accesat: 18 mai 2024.

[3] Open Handset Alliance Google LLC. Application fundamentals. https://developer.android.com/guide/components/fundamentals, 2024. Accesat: 19 mai 2024.

[4] Open Handset Alliance Google LLC. Android architecture. https://source.android.com/docs/core/architecture, 2024. Accesat: 19 mai 2024.

[5] Open Handset Alliance Google LLC. Permissions overview. https://developer.android.com/guide/topics/permissions/overview, 2024. Accesat: 18 mai 2024.

[6] Open Handset Alliance Google LLC. Android open source project. https://source.android.com, 2024. Accesat: 19 mai 2024.

[7] StatCounter. Mobile operating system market share worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide, 2024. Accesat: 18 mai 2024.

[8] The Rust Team. The rust programming language. https://www.rust-lang.org/, 2024. Accesat: 18 mai 2024.

[9] tokio rs. axum. https://github.com/tokio-rs/axum/, 2024. Accesat: 18 mai 2024.

[10] Jordan Walke. React. https://react.dev/, 2024. Accesat: 18 mai 2024.

[11] Wikipedia. apk (file format). https://en.wikipedia.org/wiki/Apk_(file_format), 2024. Accesat: 18 mai 2024.