Masarykova univerzita
Fakulta informatiky

Detekcia prebalených aplikácií v OS Android

Diplomová práca

Martin Styk

Brno, jeseň 2017

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Martin Styk

Vedúci práce: Ing. Mgr. et Mgr. Zdeněk Říha, Ph.D.

Poďakovanie

Rád by som sa poďakoval vedúcemu práce Ing. Mgr. et Mgr. Zdeňkovi Říhovi, Ph.D. za venovaný čas, ochotu a cenné pripomienky, ktoré mi pomohli pri tvorbe tejto práce.

Zhrnutie

Cieľom tejto diplomovej práce je vytvoriť mobilnú aplikáciu pre platformu Android, ktorá získava metadáta o nainštalovaných aplikačných balíčkoch. Aplikácia na základe týchto dát detekuje potenciálne škodlivé prebalené aplikácie.
Teoretická časť práce sa zaoberá Android aplikáciami. Práca predstavuje metódu prebaľovania APK súborov a existujúce spôsoby detekcie takto modifikovaných aplikácií.

Kľúčové slová

APK súbor, Android, malvér, prebalené aplikácie, analýza aplikácií, AndroidManifest.xml

Obsah

1 Úvod

Android je globálne najrozšírenejší mobilný operačný systém. Tento systém, pôvodne určený pre chytré mobilné telefóny, sa dnes vďaka širokému spektru poskytovaných služieb a prispôsobiteľnosti rozšíril na viacero nových platforiem, ako napríklad zábavné systémy automobilov, softvér televízií, notebooky alebo inteligentné hodinky. Jedným z hlavných dôvodov popularity Androidu je dobrá dostupnosť aplikácií. Vďaka veľkému množstvu aplikácií má každý užívateľ možnosť prispôsobiť softvérový obsah zariadenia svojim potrebám. Z pohľadu užívateľa je kľúčová funkcionalita poskytovaná aplikáciou. Aplikácie však obsahujú aj množstvo metadát, ktoré nám môžu o ich fungovaní veľa prezradiť. Mobilné aplikácie často pristupujú k citlivým dátam užívateľov. Preto je dôležité zaistiť bezpečnosť na všetkých úrovniach. Systém Android dlhodobo bojuje s bezpečnostným rizikom spojeným s distribúciou aplikácií. Táto zraniteľnosť spočíva v modifikácii existujúcej aplikácie tak, aby vykonávala činnosť ktorá je škodlivá pre užívateľa alebo pôvodného vývojára aplikácie. Takto modifikované aplikácie sú často označované ako prebalené.

Táto práca sa zaoberá získavaním metadát o aplikáciách na systéme Android a ich následnom využití pri detekcii potenciálne škodlivých prebalených aplikácií.

Teoretická časť práce má za úlohu oboznámiť čitateľa s hlavnými aspektami Android aplikácií so zameraním na bezpečnosť a spôsob ich distribúcie. Práca predstavuje metódu prebaľovania inštalačných APK súborov spolu s vybranými spôsobmi detekcie takýchto aplikácií.

Cieľom praktickej časti tejto práce je vytvoriť mobilnú aplikáciu slúžiacu na analýzu aplikácií pre operačný systém Android. Táto aplikácia by mala získavať detailné metainformácie o nainštalovaných aplikačných balíčkoch. Tieto informácie by mali obsahovať údaje o zabezpečení a digitálnom podpise, rôznych komponentoch aplikácie a bezpečnostných povoleniach vyžadovaných aplikáciou. Vytvorená aplikácia by mala tieto informácie sprostredkovávať užívateľom v prehľadnej a zrozumiteľnej forme. Cieľom je publikovať mobilnú aplikáciu v obchode *Google Play*.

Mobilná aplikácia nebude analyzovať nainštalované aplikácie výhradne za účelom prezentácie metadát užívateľovi. Zámerom tejto

práce je využiť metadáta získané analýzou aplikácií za účelom detekcie prebalených kópií. Cieľom je návrhnúť a implementovať metódu detekcie prebalených aplikácií využívajúcu podobnosť ich metadát. Metóda detekcie prebalených aplikácií by sa mala opierať o zistenia predchádzajúcich výskumov, ktoré ďalej rozširuje a posúva z vedeckej roviny do praxe. Táto metóda bude sprostredkovaná koncovým užívateľom. Na základe reálne dosiahnutých výsledkov a spätnej väzby od užívateľov budú zhodnotené klady a zápory navrhnutej metódy.

Očakávaným výsledkom práce je kompletný systém na analýzu aplikácií a detekciu prebalených kópií, pozostávajúci z mobilnej aplikácie a serveru s databázou, ktorá obsahuje dáta o veľkom počte aplikácií.

2 Aplikácie pre platformu Android

2.1 Android Application Package

Aplikácie pre platformu Android sú vyvíjané predovšetkým v jazyku Java, ich vývoj je však možný aj v jazykoch C++, Kotlin, Python, alebo prostredníctvom technológie Xamarin, ktorá využíva jazyk C#. Výstupom kompilácie a zabalenia Android aplikácie je súbor typu *Android Application Package*. Tento súbor slúži ako inštalačný balíček, ktorý je pomocou distribučných kanálov distribuovaný k cieľovým užívateľom.

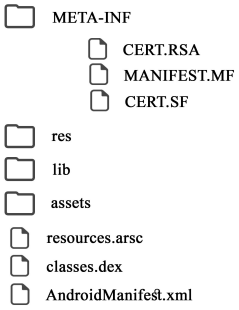
Android Application Package, skrátene APK, je formát archívnych súborov. APK súbory sú asociované s MIME typom *application/vnd.android.package-archive* a príponou *.apk* [1].

Formát APK súborov rozširuje JAR formát. Obidva spomínané formáty vychádzajú z archívneho formátu ZIP. Vďaka tejto vlastnosti je možné obsah archívu extrahovať pomocou štandardných nástrojov pre prácu so súbormi vo formáte ZIP.

APK súbory vznikajú ako výstup kompletnej kompilácie a zabalenia aplikácie. APK archív obsahuje súbory potrebné pre spustenie aplikácie [2].

2.1.1 Štruktúra

APK súbory dodržiavajú presne stanovenú vnútornú štruktúru. Táto štruktúra je znázornená na obrázku 2.1.



Obr. 2.1: Typická štruktúra APK súboru

V APK súbore nájdeme nasledujúce súbory a adresáre:

- *AndroidManifest.xml* – súbor obsahujúci meta informácie o aplikácii. Pomocou tohto súboru oznamuje aplikácia operačnému systému svoju identitu a požiadavky. Android Manifest sa v APK balíčku nachádza vo formáte skompilovaného binárneho XML. Súbor AndroidManifest.xml obsahuje okrem iného informácie o nasledujúcich vlastnostiach aplikácie:
  - meno balíka aplikácie slúžiace ako unikátny identifikátor aplikácie,
  - najnižšiu verziu Android API na ktorej je aplikácia spustiteľná,
  - popisuje základné komponenty aplikácie, obsahuje informácie o aktivitách, službách (service), poskytovateľoch obsahu (content provider), prijímačoch (broadcast receiver) a triedach, ktoré ich v rámci aplikácie implementujú,
  - deklaruje povolenia vyžadované aplikáciou na prístup k zabezpečeným častiam Android API,
  - definuje povolenia vyžadované od iných aplikácií pri interakcii s danou aplikáciou [3].

- *Classes.dex* – súbor obsahujúci spustiteľný kód aplikácie. Súbor typu DEX (Dalvik Executable) obsahuje operačné kódy a inštrukcie špecifické pre behové prostredie Android Runtime a virtuálny stroj Dalvik Virtual Machine (pre verzie Android 4.4.4 a staršie) [4].

- *Resources.arsc* – obsahuje informácie o zdrojových súboroch aplikácie. Tento súbor určuje vzťah medzi zdrojovými súbormi a ich identifikátormi, pomocou ktorých sú súbory referencované v zdrojovom kóde aplikácie.

- *Assets* – adresár obsahujúci neskomprimované zdrojové súbory. Na rozdiel od zdrojových súborov z priečinku res, tieto zdroje nie sú referencované identifikátorom, ale prístup k nim je umožnený pomocou API triedy *AssetManager*.

- *Lib* – v tomto adresári sa nachádzajú skompilované knižnice určené pre konkrétnu architektúru procesora. Medzi podporované architektúry patrí ARMv7 a novšie, x86, x86_64.

- *Res* – adresár obsahujúci zdroje aplikácie. Obsah tohto adresára je tvorený predovšetkým multimediálnymi súbormi ako napríklad obrázky a ikony, ale aj súbormi vo formáte XML ktoré definujú užívateľské rozhranie, farby, lokalizované texty alebo štýl aplikácie. Súbory umiestnené v tomto adresári sú v zdrojovom kóde referencované pomocou unikátnych číselných identifikátorov, ktoré sú vygenerované počas kompilácie a nachádzajú sa v triede *R.java*. Obsah priečinku je ďalej logicky členený do viacerých podpriečinkov. Android podporuje lokalizáciu a rôzne verzie zdrojových súborov, ktoré použije na základe údajov o zariadení na ktorom je aplikácia spustená [5].

- *META-INF* – v tomto adresári sa nachádzajú súbory zaručujúce digitálny podpis a integritu APK balíčka.

  - CERT.RSA – súbor obsahujúci certifikát podpisu aplikácie.
  - MANIFEST.MF – súbor obsahujúci hash každého súboru v APK archíve. Využíva hashovaciu funkciu *SHA-1*.
  - CERT.SF – súbor obsahujúci záznam o každom súbore v APK archíve. Záznam o jednom súbore obsahuje jeho názov a *SHA-1* hash záznamu o tomto súbore z MANIFEST.MF.

2.2 Distribúcia aplikácií

Aplikácie na platforme Android sú distribuované ako inštalačné APK balíčky. Podľa formy distribúcie a inštalácie môžeme aplikácie rozdeliť na dve základné skupiny:

- *Predinštalované (systémové) aplikácie* – tieto aplikácie sú distribuované priamo so systémom a sú nainštalované pri iniciálnom spustení systému,

- *Užívateľské aplikácie* – aplikácie distribuované vo forme balíčkov poskytovaných najčastejšie pomocou obchodov s aplikáciami.

Zoznam predinštalovaných aplikácií určuje výrobca zariadenia. Používateľ nemá možnosť systémové aplikácie odinštalovať. Do kategórie predinštalovaných systémových aplikácií patria aplikácie umožňujúce ovládanie základného vybavenia zariadenia ako napríklad aplikácia telefón alebo fotoaparát. Do tejto kategórie patria aj predinštalované služby od spoločnosti Google.

Užívateľské aplikácie slúžia na prispôsobenie systému pre potreby konkrétneho užívateľa. O inštalácii a odstránení užívateľských aplikácií rozhoduje používateľ. Na pohodlné prehliadanie a inštaláciu aplikácií slúžia obchody (app stores). Pomocou centralizovaných obchodov s aplikáciami môžu softvéroví vývojári jednoducho distribuovať svoju aplikáciu k veľkému počtu užívateľov. Najrozšírenejším a najznámejším obchodom s aplikáciami pre platformu Android je oficiálny obchod *Google Play Store*.

Okrem oficiálneho obchodu sú aplikácie distribuované aj pomocou alternatívnych distribučných kanálov. Medzi takéto kanály zaraďujeme alternatívne obchody s aplikáciami ako napríklad *Amazon Appstore*. Častým distribučným kanálom sú aj obchody výrobcov mobilných zariadení alebo obchody mobilných operátorov. Existuje viacero služieb tretích strán, ktoré sprostredkovávajú prístup k oficiálnym aplikáciám z týchto obchodov. APK súbory sú distribuované aj pomocou warezových portálov na zdieľanie súborov alebo underground obchodov.

Z dôvodu udržania bezpečného a spoľahlivého fungovania platformy Android je dôležitá bezpečnosť a kvalita distribuovaných aplikácií. Výskum porovnávajúci kvalitu aplikácií na neoficiálnych distribučných platformách ukázal, že tieto distribučné kanály obsahujú 5 až 13% aplikácií, ktoré sú klonom oficiálnych aplikácií distribuovaných pomocou *Google Play Store* [6].

2.3 Inštalácia aplikácií

Platforma Android poskytuje služby na inštaláciu, aktualizáciu a odinštalovanie aplikácií.
Základnú funkcionalitu poskytujú nasledujúce služby.

- *Package Installer* – implementuje funkcionalitu inštalácie, aktualizácie a odinštalovania aplikácií,

- *Package Manager Service* – poskytuje API pre správu nainštalovaných softvérových balíčkov,
- *Daemon Installd* – vytvára a spravuje adresáre potrebné pre nainštalovanie aplikácie.

Inštalácia aplikácie pozostáva z overenia aplikácie a extrakcie dát z APK balíčka. Aplikácia je overená na základe digitálneho podpisu a metadát zo súboru *AndroidManifest.xml*. Systémové aplikácie sú rozbalené do adresára */system/app*. Užívateľské aplikácie sú nainštalované v adresári */data/app*. Systém Android uchováva v spomínaných adresároch kompletné APK súbory [7].

2.4 Bezpečnosť

Bezpečnosť a integrita distribuovaných APK balíčkov je dôležitým aspektom zabezpečenia celého systému Android.

2.4.1 Podpis aplikácie

Každý APK balíček musí byť digitálne podpísaný. Systém Android zabráni inštalácii digitálne nepodpísaných aplikácií. Súbory súvisiace s podpisom APK balíčka sa nachádzajú v priečinku *META-INF*. Počas digitálneho podpisu aplikačného balíčka sa vygenerujú *SHA-1* alebo *SHA-256* hashe súborov obsiahnutých v APK archíve. Záznamy o súboroch a ich hashoch sú uložené v súboroch *META-INF/MANIFEST.MF* a *META-INF/CERT.SF*. Celý APK balíček je následne podpísaný pomocou utility *Jarsigner*.

Android podporuje dve schémy digitálneho podpisu aplikácie. Schéma verzie 1 je založená na podpise štandardného Java balíčka. Táto schéma nechráni niektoré súbory špecifické pre APK balíčky, ako napríklad metadáta o aplikácii. Nechránené súbory musia byť za účelom overenia dekomprimované počas procesu overenia, čo zvyšuje časovú a pamäťovú náročnosť overenia APK balíčka. Od verzie Android 7.0 (verzia API 24) je podporovaná schéma podpisu vo verzii 2, ktorá je špecializovaná na podpis APK balíčka a eliminuje nedostatky predchádzajúcej verzie. V súčasnosti sú aplikácie zvyčajne podpísané oboma spôsobmi [8].

Kontrola integrity

Hashe jednotlivých súborov v APK balíčku vygenerované počas procesu podpisu, slúžia na zaručenie integrity. Počas inštalácie aplikácie sa v rámci procesu overenia aplikácie kontrolujú hashe jednotlivých súborov. V prípade zmeny súborov v aplikácii bez jej opätovného podpisu systém zabráni inštalácii. Pri zmene obsahu musí byť APK balíček podpísaný znova.

Identifikácia autora

Podpis balíčka slúži na identifikáciu autora aplikácie. V prípade, že je viacero aplikácií podpísaných jedným kľúčom, pochádzajú tieto aplikácie od jedného vydavateľa. Všetky verzie jednej aplikácie musia byť podpísané rovnakým certifikátom. Unikátnym identifikátorom aplikácie je meno hlavného balíka aplikácie (package name). Android považuje aplikácie s rovnakým menom balíka za rôzne verzie jednej aplikácie a pri aktualizácii nahrádza staršiu verzie novšou. Všetky aplikácie s rovnakým menom balíka musia byť preto podpísané rovnakým certifikátom, čo zaručuje, že pochádzajú od rovnakého vydavateľa. V prípade, že sú podpísané rôznymi certifikátmi, Android ich inštaláciu zamietne.

Správa podpisových kľúčov

Platforma Android povoľuje používanie tzv. *self-signed* certifikátov. *Self-signed* certifikáty sú podpísané identitou ktorú identifikujú.

Z pohľadu vývojára je správa podpisového kľúča veľmi dôležitá. Vývojári aplikácií majú pri správe podpisového kľúča dve možnosti. Môžu používať manuálnu správu, alebo službu *Google Play App Signing*.

- *Manuálna správa kľúča* – v prípade použitia manuálnej správy kľúča je aplikácia podpísaná vývojárom a nahraná do obchodu s aplikáciami. Odtiaľ je ďalej distribuovaná k užívateľom. V prípade straty podpisového kľúča nie je vývojár aplikácie schopný vydávať nové verzie aplikácie pod rovnakým názvom balíka.

- *Google Play App Signing* – pri použití bezpečnej správy podpisových kľúčov *Google Play App Signing* je kľúč používaný na podpis

balíčka spravovaný službou *Google Play*. Vývojár podpíše aplikáciu svojim kľúčom používaným na komunikáciu so službou *Google Play*, ktorá aplikáciu na novo podpíše podpisovým kľúčom. Aplikácia distribuovaná k užívateľom je teda podpísaná kľúčom, ktorý je manažovaný službou *Google Play* [9].

2.4.2 Známe bezpečnostné riziká

Existuje viacero známych zraniteľností súvisiacich s distribúciou aplikácií pre platformu Android. Najviac rozšíreným problémom je prebaľovanie APK balíčkov. Pri tomto procese je aplikácia rozbalená a niektoré časti aplikácie sú upravené. Detailný popis prebaľovania aplikácií a rizík s tým spojených sa nachádza v kapitole 3.

Metóda prebaľovania upraví aplikáciu a nanovo ju podpíše iným podpisovým kľúčom. Ďalšie známe metódy útokov využívajú predovšetkým zraniteľnosti pri overovacom procese APK balíčku a nemodifikujú originálny podpis.

Zraniteľnosť pri inštalácií nekompletného APK súboru využíva fakt, že pri overovaní hashov súborov sa ignorujú záznamy, ktoré v balíku neexistujú. Súbory je teda možné z APK archívu odstrániť bez jeho opätovného podpísania. Modifikovanú aplikáciu je možné distribuovať ako originálnu verziu. Aplikácia po odstránení súborov nebude pracovať správne a môže dostať celé zariadenie do nepoužiteľného stavu [10].

Pri verifikačnom procese sa nekontrolujú súbory slúžiace na podpis balíčka. To umožňuje útočníkovi nahrať do adresára *META-INF* ľubovoľný súbor bez potreby opätovného podpísania. Útočník tak môže do aplikácií nahrať kusy kódu. Tento kód však nevykoná pôvodná aplikácia, ktorej zdrojové kódy zostali nepozmenené. Namiesto toho útočník distribuuje svoju vlastnú aplikáciu, ktorá pomocou classloaderu použije daný kód vložený do originálnej aplikácie [10].

3 Prebalené aplikácie

Prebaľovanie aplikácií je formou útoku na systému Android. Princíp útoku spočíva v modifikácii originálnej aplikácie a jej následnej redistribúcii. Počas modifikácie APK súboru môže útočník pridať škodlivú funkcionalitu a upraviť tak správanie aplikácie. Prebaľovanie aplikácií je jednou z najbežnejších techník distribúcie malvéru a škodlivého kódu na platforme Android.

Štandardný postup tvorby prebalených aplikácií z pohľadu útočníka:

1. Stiahnutie populárnej aplikácie,
2. Dekompilácia aplikácie pomocou nástrojov reverzného inžinierstva,
3. Modifikácia aplikácie pridaním škodlivého kódu,
4. Zabalenie aplikácie,
5. Podpísanie vlastným self-signed certifikátom,
6. Distribúcia aplikácie pomocou oficiálnych a neoficiálnych kanálov.

Tvorcovia prebalených aplikácií využívajú popularitu originálnych aplikácií za účelom propagácie malvéru. Cieľom prebaľovania APK súborov nie je vytvoriť novú odlišnú aplikáciu s využitím zdrojového kódu a vlastností existujúcej aplikácie. Namiesto toho je zámerom prebalených aplikácií vydávať sa za pôvodnú aplikáciu a pripomínať ju v maximálnej možnej miere. Modifikované verzie navonok pôsobia identicky ako originálne verzie. Z pohľadu užívateľa poskytuje modifikovaná aplikácia rovnakú funkcionalitu a užívateľské rozhranie ako originál.

Prebaľovanie aplikácií je populárne aj vďaka warezovým portálom slúžiacich na zdieľanie nelegálneho obsahu. Warezové portály ponúkajú bezplatne aj aplikácie, ktoré sú v službe *Google Play* dostupné iba za poplatok, čo výrazne prispieva k šíreniu takýchto aplikácií. Prebalené aplikácie však nie sú distribuované výhradne pomocou neoficiálnych kanálov, ale aj prostredníctvom *Google Play*.

Prebalené aplikácie sú škodlivé nie len pre koncových užívateľov, ale aj vývojárov, vlastníkov duševných práv a prevádzkovateľov obchodov s aplikáciami [11].

3.1 Spôsob modifikácie aplikácií

Za účelom modifikácie aplikácie upraví útočník inštalačný APK balíček. Modifikácia už existujúcej nainštalovanej aplikácie nie je možná. Spravidla je modifikovaný len jej inštalačný balíček a škodlivá aplikácia sa do zariadenia dostane inštaláciou takéhoto balíčku. APK súbory majú definovaný formát a štruktúru, ktoré sú popísané v kapitole 2.1.1. Vďaka ich formátu a štrukturovanému procesu tvorby je ich dekompilácia nenáročná.

Existuje viacero nástrojov reverzného inžinierstva, ktoré môžu byť použité pri tvorbe prebalených aplikácií. Najkomplexnejším nástrojom, ktorý je možné použiť na kompletné rozbalenie a následné zabalenie aplikácie je nástroj ApkTool [12]. Za účelom modifikácie zdrojového kódu je možné využiť bežne dostupné nástroje ako *dex2jar*, *jd-gui* a *smali* [13–15].

Z hľadiska modifikácie sú pri prebaľovaní zaujímavé nasledujúce súbory:

- *AndroidManifest.xml* – upravením tohto súboru je možné pridávať nové bezpečnostné povolenia. Súbor je v APK balíku dostupný ako skomprimované XML. Skomprimované (binárne) XML je možné konvertovať do editovatelnej formy pomocou nástroja *ApkTool* [12].

- *Classes.dex* – vykonanie zmien v zdrojovom kóde umožňuje komplentú modifikáciu funkcionality aplikácie. Tento súbor obsahujúci zdrojové kódy je možné editovať pomocou spomínaných nástrojov *dex2jar*, *jd-gui* a *smali*.

Zásadnou vecou, ktorá odlišuje prebalenú aplikáciu od originálu je digitálny podpis. Po editácii aplikácie a jej opätovnom zabalení je nutné aplikáciu digitálne podpísať. Za predpokladu, že privátny kľúč pôvodného vydavateľa neunikol a nie je k dispozícii útočníkovi, podpis originálnej a prebalenej aplikácie sa bude líšiť.

3.2 Časté typy modifikácie

Modifikácia zdrojových kódov aplikácie predstavuje vážne ohrozenie bezpečnosti. Nasledujúce body popisujú najčastejšie typy modifikácie zdrojových kódov.

Modifikácia reklamných modulov

Väčšina aplikácií je užívateľom dostupná zadarmo. Zdroj príjmu pre vývojárov predstavujú reklamné moduly integrované do aplikácie. Najčastejšou modifikáciou vykonávanou pri prebaľovaní aplikácií je nahradenie pôvodných reklamných modulov. Výnosy z reklamy sú tak presmerované z účtu pôvodného vývojára na účet útočníka. Najjednoduchšou úpravou je využitie existujúcich reklamných služieb a presmerovanie zisku na iný účet pomocou zmeny API kľúča. Takmer 65 % prebalených aplikácií obsahuje modifikované reklamné knižnice [16].

Zneužitie prémiových služieb

Niektoré prebalené aplikácie obsahujú platené služby, ktoré nie sú súčasťou originálnej aplikácie. Najčastejším spôsobom zneužitia prémiových služieb je odosielanie textových správ na spoplatnené prémiové čísla. Prebalená aplikácia vyžaduje pri inštalácii povolenie na odosielanie správ, ktoré ďalej odosiela bez používateľovho vedomia. Takto upravená aplikácia môže zaregistrovať vlastnú službu, ktorá odfiltruje SMS správy od telekomunikačného poskytovateľa a zabráni tak upozorneniam o využití platených prémiových služieb. Známym príkladom takto napadnutej aplikácie je prehliadač QQ Browser [16].

Prístup k citlivým dátam

Aplikácie môžu byť upravené za účelom zbierania informácií ako telefónne čísla kontaktov, emailové adresy, história prehliadača alebo zoznam nainštalovaných aplikácií. Túto činnosť vykonávajú služby na pozadí. Zozbierané dáta sú odosielané na vzdialené webové servre.

Inštalácia malvéru

Prebalené aplikácie môžu na pozadí spustiť sťahovanie a inštaláciu ďalšieho malvéru.

Vzdialený prístup a ovládanie zariadenia

Za účelom získania kontroly nad zariadením môže aplikácia nadviazať spojenie so vzdialeným serverom, ktorý je pod kontrolou útočníka. Server posiela aplikácii príkazy a prijíma od nej odpovede. Príkladom takéhoto správania je prebalená verzia aplikácie Stupid Birds, ktorá obsahuje vložený kód, pomocou ktorého sa pripojí na URL adresu z ktorej prijíma príkazy. Na základe týchto príkazov dokáže aplikácia na pozadí stiahnúť iné aplikácie, a umiestniť odkazy na ich inštaláciu priamo na plochu zariadenia [16].

Vloženie viacerých DEX súborov

Po zmenách aplikácie prezentovaných v predchádzajúcich bodoch je nutné APK balíček nanovo podpísať. Vo verzii Android 4.x existuje zraniteľnosť známa ako Master Key vulnerability. Ide o jednu z najzávažnejších zraniteľností pri overovaní APK balíčku počas jeho inštalácie. V prípade, že APK balíček obsahuje dva súbory s rovnakým názvom, pri overovaní hashov súborov sa použije prvý záznam o takomto súbore, avšak Android použije pri inštalácii druhý súbor. To umožňuje upraviť aplikáciu bez potreby opätovného podpísania [17]. Tento fakt je možné využiť na vloženie nového súboru so zdrojovým kódom, ktorý kompletne nahradí pôvodnú logiku aplikácie.

3.3 Výskyt prebalených aplikácií

Pre platformu Android existujú milióny aplikácií. Počet aplikácií dostupných v oficiálnom obchode *Google Play* prekročil v septembri 2017 3 milióny [18]. Množstvo aplikácií je dostupných prostredníctvom neoficiálnych distribučných kanálov. Výsledky predchádzajúcich výskumov ukazujú, že prebalené aplikácie predstavujú nezanedbateľnú časť dostupných aplikácií.

Analýzou 23 000 aplikácií zo šiestich alternatívnych obchodov v rámci práce *Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces* sa zistilo, že pomer prebalených aplikácií sa v alternatívnych obchodoch pohybuje medzi 5 až 13 % [19]. Počet aplikácií, ktoré sú nie len prebalené, ale pridávajú do originálnych aplikácií škodlivé správanie je taktiež významný. Wu Zhou a kolektív na základe analýzy vzorky 85 000 aplikácií identifikovali, že počet takýchto aplikácií sa pohybuje medzi 0,97 až 2,7 % všetkých aplikácií [20]. Až 86 % aplikácií obsahujúcich malvér vzniklo prebalením originálnej aplikácie a vložením škodlivého kódu [21].

Alternatívne distribučné platformy, ktoré často neposkytujú formu kontroly obsahu, obsahujú väčší počet prebalených APK súborov ako oficiálne zdroje. Proti prebaľovaniu aplikácií však nie je dostatočne zabezpečený ani oficiálny obchod [22].

4 Známe metódy detekcie prebalených aplikácií

Šírenie prebalených aplikácií predstavuje významnú hrozbu pre celý systém Android. Z hľadiska bezpečnosti systému je v súčasnosti detekcia prebalených aplikácií jednou z najdôležitejších tém. Téme prebalených APK súborov sa vo svojich prácach venuje viacero výskumných tímov. Bolo navrhnutých a implementovaných viacero spôsobov detekcie takýchto aplikácií.

Detekcia prebalených APK balíčkov vychádza z nasledujúcich pozorovaní:

- Prebalená aplikácia zachováva funkcionalitu pôvodnej aplikácie,
- Prebalená aplikácia zachováva vzhľad pôvodnej aplikácie,
- Prebalená a pôvodná aplikácia sú podpísané rôznymi entitami.

Algoritmy detekcie prebalených aplikácií pozostávajú zvyčajne z dvoch základných krokov:

- Extrakcia vlastností aplikácií,
- Párové porovnanie aplikácií na základe extrahovaných vlastností.

Základným rozdielom medzi existujúcimi metódami detekcie prebalených APK súborov sú vlastnosti aplikácií použité pri detekcii malvérových duplikátov. Väčšia časť existujúcich algoritmov využíva podobnosť zdrojových kódov a inštrukcií. Existuje niekoľko metód, ktoré na detekciu prebalených aplikácií využívajú podobnosť multimediálnych súborov obsiahnutých v APK balíčkoch.

4.1 DroidMOSS

Metóda *DroidMOSS* je založená na podobnosti zdrojového kódu originálu a prebalenej kópie. Táto metóda sa zameriava na podobnosť

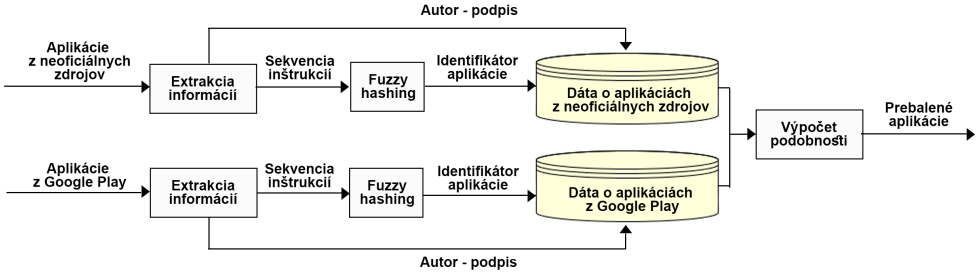
aplikačného Java bytekódu a nezaoberá sa natívnym kódom. Počas prebaľovania je jednoduchšie modifikovať Java kód ako natívny kód a len malá časť aplikácií (približne 5 %) používa natívny kód.

DroidMOSS pozostáva z troch kľúčových krokov. Prvým krokom je extrakcia aplikačných inštrukcií a získanie informácií o vydavateľovi aplikácie v podobe certifikátu. Tieto dva atribúty charakterizujú a odlišujú aplikácie. Za účelom extrakcie aplikačného kódu je použitý nástroj *smali/baksmali*, pomocou ktorého je súbor *classes.dex* dekompilovaný do Dalvik bytekódu [15]. Počas procesu prebaľovania môže byť použitá obfuskácia kódu. Počas obfuskácie sú premenované názvy balíkov, tried, metód a premenných. *DroidMOSS* sa s obfuskáciou kódu vysporiadava pomocou ignorovania operandov. Pri tvorbe identifikátora aplikácie sú zohľadnené len operačné inštrukcie. Intuitívne môžeme tento prístup vysvetliť tak, že počas prebaľovania je jednoduché upraviť kód premenovaním premenných, oveľa náročnejšie je zmeniť kód tak, aby používal odlišné inštrukcie. Informácie o autorovi aplikácie pochádzajú zo súborov v adresári *META-INF*. *DroidMOSS* vygeneruje identifikátor autora pomocou zahashovania verejného kľuča, mena vývojára a jeho kontaktných informácií.

Druhý krok spočíva vo vygenerovaní unikátneho identifikátoru každej aplikácie. Hlavným dôvodom pre tvorbu identifikátoru je komplexnosť a veľký počet inštrukcií v jednej aplikácii. Dĺžka identifikátoru je výrazne kratšia ako dáta o inštrukciách aplikácie, čo umožňuje efektívnejšie porovnávanie aplikácií. Vytvorenie identifikátoru pomocou hashovania celého kódu aplikácie môže byť efektívne použité na overenie úplnej zhody dvoch aplikácií. Tento postup však nie je možné aplikovať pri určení podobnosti aplikácií. Z tohto dôvodu využíva *DroidMOSS* špeciálnu hashovaciu techniku *fuzzy hashing* [23]. Namiesto vytvorenia identifikátoru celej sekvencie inštrukcií je táto sekvencia najskôr rozdelená na kratšie časti. Následne je každá z týchto častí hashovaná osobitne. Na vyhodnotenie podobnosti aplikácií sú použité identifikátory (hashe) týchto kratších sekvencií.

Posledným krokom je párové porovnanie aplikácií. Aplikácie sú rozdelené do dvoch skupín. Jedna skupina obsahuje aplikácie z *Google Play Store*, druhá aplikácie z neoficiálnych zdrojov. *DroidMOSS* využíva silný predpoklad, že aplikácie pochádzajúce z *Google Play Store* nie sú prebalené. Aplikácie z rozdielnych skupín sú vzájomne

porovnané. Na vyhodnotenie podobnosti je použitý algoritmus navrhnutý pomocou techniky dynamického programovania. Podobnosť aplikácií je určená ako minimálna vzdialenosť medzi identifikátormi dvoch sekvencií inštrukcií. Vzdialenosť je reprezentovaná počtom úprav potrebných na pretvorenie identifikátora jednej sekvencie na identifikátor druhej sekvencie. Ak vypočítaná podobnosť presiahne definovanú hranicu a porovnávané aplikácie sú podpísané rôznymi vydavateľmi, aplikácia ktorá nepochádza z *Google Play Store* je označená za prebalenú. Použitá technika umožňuje efektívne lokalizovať zmeny vykonané v prebalených aplikáciách.



Obr. 4.1: Postup metódy DroidMOSS

Systém *DroidMOSS* bol aplikovaný na kolekciu 86 000 aplikácií. Pomocou tejto techniky sa ukázalo, že 5 až 13 % aplikácií distribuovaných pomocou alternatívnych zdrojov je prebalených [19].

4.2 ImageStruct

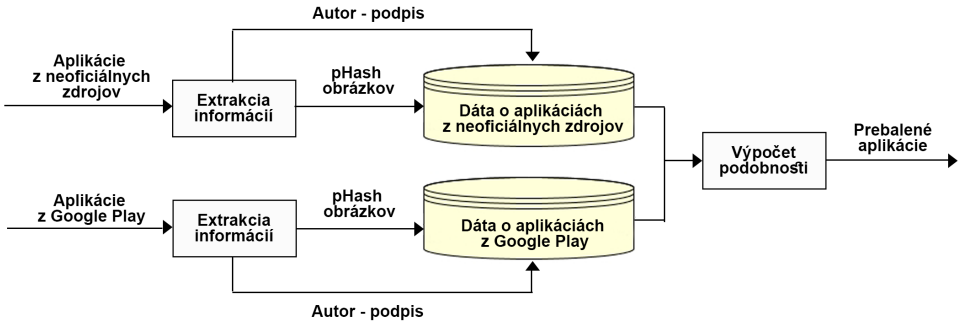
Metóda *ImageStruct* používa pri detekcii prebalených aplikácií podobnosť obrázkových súborov. Tento spôsob je založený na pozorovaní, že podobné aplikácie (rôzne verzie tej istej aplikácie alebo prebalené aplikácie) obsahujú podobné obrázky, zatiaľ čo obrázky v aplikáciách s rôznou funkcionalitou sú rozdielne.

ImageStruct získava z APK balíčka charakteristické dáta obrázkov. Na extrakciu obrázkových dát je použitý algoritmus *pHash*. Tento algoritmus používa diskrétnu kosínusovú transformáciu, pomocou ktorej z obrázka odstráni niektoré farebné frekvencie. To umožňuje

detekciu podobnosti aj pri jednoduchších úpravách, ako napríklad zmena rozlíšenia alebo vystrihnutie časti obrazu [24]. Na identifikáciu vydavateľa sú použité údaje z priečinka *META-INF*.

Extrahované dáta sú ukladané v databáze *Redis*, ktorá slúži ako rýchla cache pamäť. Podobnosť dvoch APK súborov je určená ako pomer spoločných obrázkov a všetkých obrázkov daných aplikácií.

ImageStruct porovnáva kolekciu aplikácií z *Google Play* s aplikáciami z alternatívnych obchodov, pričom využíva predpoklad, že *Google Play* neobsahuje prebalené aplikácie. Ak hodnota podobnosti prekročí stanovenú hranicu a aplikácie sú podpísané rôznymi autormi, aplikácia z alternatívneho zdroja je označená za prebalenú. Nastavenie hodnoty hranice nad ktorou sú aplikácie považované za duplikáty je z hľadiska presnosti a spoľahlivosti detekcie kľúčové. Autori uvádzajú, že pomocou viacerých experimentov určili túto hodnotu na $0,6$ [25].



Obr. 4.2: Postup metódy ImageStruct

Pomocou experimentov nad databázou 48 000 aplikácií z *Google Play* a alternatívnych zdrojov odhalila táto metóda, že výskyt prebalených aplikácií v alternatívnych zdrojoch sa pohybuje medzi $6,7\%$ až $14,5\ \%$. Pomer prebalených aplikácií detekovaných touto metódou je takmer rovnaký ako v prípade metódy *DroidMOSS*. Ako dôkaz použiteľnosti metód zameraných na podobnosť multimediálnych súborov bolo v rámci experimentov vykonané porovnanie hodnôt podobnosti určených metódou *ImageStruct* a metódou *AndroGuard*, ktorá využíva podobnosť zdrojového kódu. Výsledky ukazujú, že medzi týmito postupmi existuje silná pozitívna korelácia [25].

4.3 FSquaDRA

Základnou myšlienkou metódy *FSquaDRA* je, že aplikácia obsahuje okrem zdrojového kódu a obrázkov aj množstvo iných súborov, ktoré definujú jej identitu. Preto táto metóda porovnáva všetky súbory dvoch aplikácií. Na evaluáciu podobnosti dvoch APK súborov je využitý Jaccardov index, ktorý sa určí ako pomer súborov, ktoré sa nachádzajú v obidvoch balíčkoch, ku počtu súborov, ktoré sa nachádzajú v aspoň jednej z aplikácií.

Pri porovnaní súborov sú využité ich hashe vygenerované počas podpisu APK balíčka, ktoré sa nachádzajú v súbore *MANIFEST.MF*. Využitie existujúcich hashov zefektívňuje túto metódu. Rovnako ako pri ostatných metódach, aj *FSquaDRA* využíva údaje z adresára *META-INF* na identifikáciu vydavateľa aplikácie.

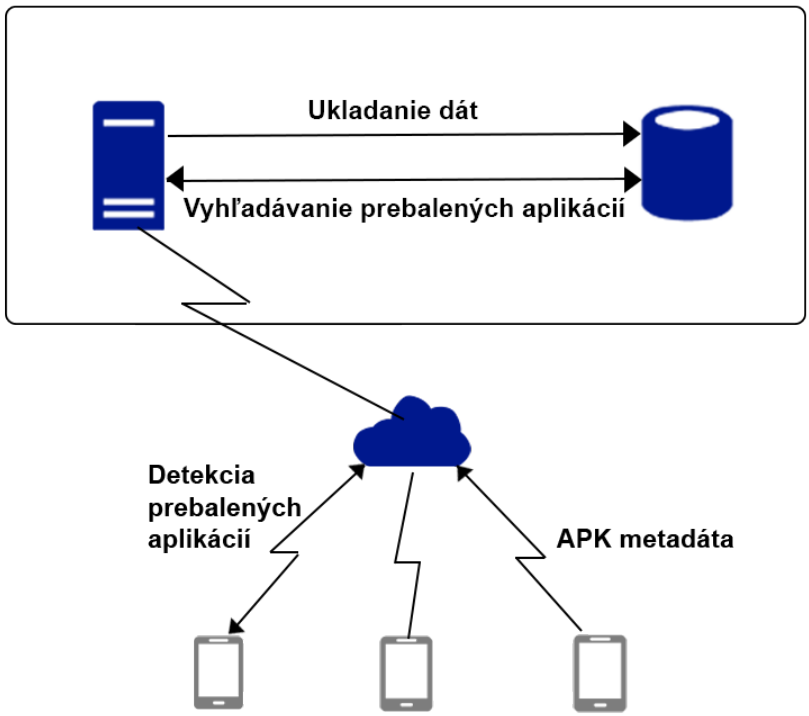
Narozdiel od metód *ImageStruct* a *DroidMOSS* boli v rámci experimentov porovnané všetky aplikácie medzi sebou. Hodnota Jaccardovho indexu, nad ktorou sú aplikácie považované za prebalené bola empiricky určená ako $0,7$. Výsledky získané pomocou tejto metódy korešpondujú s výsledkami získanými pomocou analýzy zdrojového kódu. V porovnaní s metódami založenými na analýze zdrojového kódu je tento spôsob oveľa rýchlejší. Autori uvádzajú, že za jednu minútu systém vykoná 1000 párových porovnaní, zatiaľ čo metóda porovnávania zdrojového kódu vyhodnotí za rovnaký čas podobnosť jedného páru aplikácií. Nevýhodou tejto metódy je, že neskúma obsah súborov a spolieha sa výhradne na ich hash. Pri prebaľovaní môže útočník mierne pozmeniť zdrojové súbory, čo má za následok znefunkčnenie detekcie [26].

Použiteľnosť tejto metódy bola ďalej skúmaná v práci *Evaluation of Resource-Based App Repackaging Detection in Android* [27]. V rámci tejto práce bola metóda vyhodnocovaná nad aplikáciami, ktoré obsahovali známe prebalené súbory, a teda bolo možné efektívne vyhodnotiť presnosť a úspešnosť tejto metódy. Práca potvrdila, že táto metóda je aj napriek nedostatkom dosahuje presnosť metód založených na analýze zdrojového kódu.

5 Systém Apk Analyzer

Primárnym cieľom tejto diplomovej práce je navrhnúť a implementovať mobilnú aplikáciu pre operačný systém Android, ktorá slúži na analýzu aplikačných balíčkov za účelom získania podrobných informácií o nainštalovaných aplikáciách. Na základe informácií získaných analýzou poskytuje táto mobilná aplikácia možnosť detekcie prebalených a škodlivých aplikácií.

Za účelom dosiahnutia stanovených cieľov je v rámci práce implementované softvérové riešenie pozostávajúce z mobilnej aplikácie, serverovej časti a centrálnej databázy. Obrázok 5.1 schematicky znázorňuje koncept vytvoreného systému.



Obr. 5.1: Návrh systému *Apk Analyzer*

5.1 Mobilná aplikácia

Základ systému tvorí mobilná aplikácia *Apk Analyzer*. Aplikácia vykonáva analýzu všetkých nainštalovaných aplikácií v Android zariadení a ponúka možnosť analýzy aplikácií pred ich samotnou inštaláciou. Aplikácia poskytuje užívateľovi detailné informácie o aplikáciách, vrátane možnosti zobrazenia štatistík o softvérovom obsahu jeho zariadenia a možnosti detekcie prebalených aplikácií. *Apk Analyzer* na pozadí odosiela informácie o aplikáciách na vzdialený server. Tieto informácie sú použité pri tvorbe databázy potrebnej pre detekciu prebalených aplikácií. Detailný popis vytvorenej aplikácie vrátane metód získavania informácií o aplikáciách obsahuje kapitola . .

5.2 Server

Detekcia prebalených aplikácií nie je implementovaná priamo v mobilnej aplikácii, ale na strane zdieľaného servera. Tento dizajn je potrebný z dôvodu implementácie navrhnutej metódy detekcie prebalených aplikácií. Informácie o spôsobe detekcie obsahuje kapitola 7. Navrhnutá metóda pracuje s rozsiahlou databázou dát o aplikačných balíčkoch. Takáto rozsiahla kolekcia dát nie je zdieľaná medzi mobilnými zariadeniami. Prístup k dátam zabezpečuje centrálny server.
Dôvody komunikácie mobilných zariadení a serveru:

- Mobilné zariadenia odosielajú metadáta o ich aplikáciách,
- Mobilné zariadenia odosielajú požiadavky na analýzu prebalených aplikácií,
- Mobilné zariadenia dostávajú informácie o výsledkoch analýzy prebalených aplikácií.

Server poskytuje administrátorom nasledujúce služby:

- Rozhranie na správu dát,
- Vyhľadávanie v databáze aplikácií,
- Zobrazenie štatistických údajov využitia služby,
- Zobrazenie výsledkov detekcie prebalených aplikácií spolu s údajmi vysvetľujúcimi rozhodnutia systému.

5.3 Databáza

Dáta odoslané mobilnými klientami na server sú ukladané v zdieľanom úložisku. Databáza obsahuje metadáta o všetkých aplikáciách odoslaných z mobilných zariadení. Tieto dáta obsahujú všetky kľúčové atribúty potrebné pre detekciu prebalených aplikácií. Spôsob ukladania dát predstavuje dôležitý aspekt pri detekcii prebalených APK balíčkov. Veľká časť operácií potrebných na detekciu prebalených aplikácií je implementovaná pomocou SQL príkazov priamo nad databázou.

6 Mobilná aplikácia

Ústredný bod systému Apk Analyzer tvorí mobilná aplikácia, ktorá analyzuje nainštalované aplikačné balíčky. Táto aplikácia poskytuje užívateľovi podrobné informácie o jeho aplikáciách a umožňuje mu kontrolu originality aplikácie. Vyvinutá aplikácia je od októbra 2017 dostupná v oficiálnom obchode *Google Play Store* [28]. Táto kapitola obsahuje popis vyvinutej aplikácie, spolu s technikami získavania informácií o aplikačných balíčkoch v operačnom systéme Android.

6.1 Ciele a požiadavky kladené na aplikáciu

6.1.1 Existujúce riešenia a ich nedostatky

Oficiálny obchod *Google Play Store* obsahuje viaceré aplikácie schopné analyzovať nainštalované aplikačné balíčky. V obchode sa nachádzajú aplikácie s názvom *Apk Analyzer*, *App Info* alebo *App Detective*. Najpopulárnejšie z týchto aplikácií dosahovali v januári 2018 viac ako 50 000 celkových inštalácií [29]. Pred začiatkom vývoja novej aplikácie bol vykonaný detailný rozbor existujúcich riešení, s dôrazom na identifikáciu nedostatkov, ktoré by mohli byť v novovyvinutej aplikácii odstránené.

Používateľské rozhranie

Väčšina aplikácií zameriavajúcich sa na analýzu softvérového obsahu zariadenia nedodržiava základné princípy dizajnu používateľského rozhrania. Pri niektorých aplikáciách je to spôsobené ich starším časom vydania a prispôsobením na staršie verzie systému. No aj veľká časť pravidelne aktualizovaných nových aplikácií sa nedrží doporučených dizajnových princípov, čo má za následok zhoršenie ovládateľnosti a užívateľského dojmu.

Limitované informácie

Veľká časť aplikácií, ktorých názov napovedá, že zobrazujú dáta o aplikačných balíčkoch sa zameriava na veľmi malú časť dostupných

informácií. Ako príklad môže slúžiť aplikácia *APK ANALYZER* od vydavateľa *z-project*, ktorá zobrazuje len súbor *AndroidManifest.xml* [30]. Aplikácia *App Info* zobrazuje informácie výhradne o vyžadovaných bezpečnostných povoleniach [30].

6.1.2 Funkčné požiadavky

Hlavnou motiváciou pri tvorbe bolo vytvorenie novej mobilnej aplikácie, ktorá má predpoklady presadiť sa medzi ostatnými podobnými riešeniami. Za týmto účelom je potrebné nie len vylepšiť nedostatky existujúcich aplikácií, ale aj poskytnúť nové funkcie, atraktívny dizajn a prehľadné užívateľské rozhranie. Funkčné požiadavky kladené na mobilnú aplikáciu vychádzajú z jej funkcie v celom navrhnutom systéme Apk Analyzer, kde mobilná aplikácia slúži ako zdroj aplikačných metadát a poskytuje užívateľské rozhranie.

Na základe analýzy podobných aplikácií boli na novú aplikáciu kladené nasledujúce požiadavky, ktoré sa podarilo úspešne zrealizovať.

Získanie podrobných dát o aplikáciách

Cieľom aplikácie je získať čo najpodrobnejšie dáta z rôznych častí aplikácie. Získané informácie obsahujú metadáta zo súboru *AndroidManifest.xml*, údaje o zdrojových súboroch, aplikačných komponentoch, povoleniach, a aj dáta o zdrojovom kóde aplikácie.

Analýza nenainštalovaných aplikácií

Žiadna z existujúcich aplikácií (podľa vedomia autora) v čase návrhu novej aplikácie neposkytovala analýzu aplikácií, ktoré nie sú nainštalované, ale sú prítomné v pamäti zariadenia ako inštalačné APK balíčky. Cieľom je implementovať túto funkcionalitu, ktorá poskytuje konkurenčnú výhodu oproti ostatným podobným aplikáciám.

Štatistiky o aplikáciách v zariadení

Na základe dát získaných analýzou, je možné prezentovať užívateľovi štatistické informácie o kolekcii aplikácií na jeho zariadení.

Zrozumiteľná interpretácia dát

Dáta získané pomocou analýzy je potrebné prehľadne zobraziť užívateľovi. Získané dáta obsahujú množstvo rôznych atribútov APK súboru. Cieľom je prezentovať tieto dáta tak, aby boli zrozumiteľné aj užívateľom, ktorí sa neorientujú v danej problematike. Každý atribút musí byť preto sprevádzaný krátkym textovým popisom, ktorý je dostupný užívateľovi.

Komunikácia so serverovou časťou systému

Mobilná aplikácia komunikuje so serverom, ktorý sprostredkováva funkcionalitu detekcie prebalených aplikácií. Mobilná aplikácia odosiela serveru metadáta o aplikáciách.

6.1.3 Nefunkčné požiadavky

Návrh aplikácie ovplyvňujú dve základné nefunkčné požiadavky.

Aplikácia nevyžaduje *root* zariadenia

V prípade rootu zariadenia má užívateľ právo použiť systémové prostriedky, ktoré sú obyčajnému užívateľovi zakázané. Tieto práva môžu byť využité napríklad na priamy prístup do inštalačných adresárov. Limitácia aplikácie na rootnuté zariadenia by výrazne znížila skupinu potenciálnych užívateľov. Preto musí aplikácia pracovať s oprávneniami štandardného užívateľa.

Výkonnosť

Analýza aplikácií musí zohľadňovať aspekt časovej náročnosti. Z pohľadu užívateľa je dôležité rýchle zobrazenie informácií o aplikácii.

6.2 Získavanie dát

Základná funkcionalita aplikácie spočíva v získavaní metadát o aplikáciách. Systém Android poskytuje viacero rozhraní, prostredníctvom ktorých je možné získavať dáta o aplikačných balíčkoch.

6.2.1 Package Manager

Trieda *PackageManager* poskytuje API pre získavanie rôznych dát o nainštalovaných aplikáciách. Aplikácia *Apk Analyzer* využíva metódu *getInstalledPackages()* na získanie zoznamu nainštalovaných aplikácií.

Pomocou triedy *PackageManager* je získaná veľká časť informácií, ktoré obsahuje metasúbor *AndroidManifest.xml*. Táto trieda poskytuje metódu *getPackageInfo()*, pomocou ktorej je možné získať informácie o nainštalovanom aplikačnom balíčku. Na získanie informácií o nenainštalovaných APK súboroch je možné použiť metódu *getPackageArchiveInfo()*, ktorá na rozdiel od predchádzajúcej metódy analyzuje aplikačný archív, ktorý nemusí byť nainštalovaný.

Pomocou metód *getPackageArchiveInfo()* a *getPackageInfo()* sú z aplikačného balíčka získané nasledujúce údaje:

- Základné metadáta aplikácie a informácie dostupné v súbore *AndroidManifest.xml*,
- Informácie o digitálnom podpise,
- Komponenty aplikácie,
- Definované a vyžadované bezpečnostné oprávnenia,
- Hardvérové vlastnosti zariadenia vyžadované aplikáciou.

6.2.2 Rekonštrukcia AndroidManifest.xml

Systém Android neposkytuje aplikačné programové rozhranie na prístup k súboru *AndroidManifest.xml*. Všetky dôležité informácie z tohto súboru sú dostupné pomocou API triedy *PackageManager*. Za účelom získania pôvodného manifest súboru používa aplikácia *Apk Analyzer* riešenie založené na prístupe k zdrojovým súborom. Trieda *PackageManager* poskytuje možnosť prístupu k niektorým zdrojovým súborom aplikácií. *AndroidManifest.xml* je možné nájsť medzi zvasset súbormi. Pomocou tohto prístupu však nie je možné súbor celý načítať, ale len vytvoriť tzv. pull parser, ktorý slúži na spracovanie *xml* súboru.

```
Resources apkResources = packageManager
                            .getResourcesForApplication(packageName);
XmlResourceParser parser = apkResources.getAssets()
                            .openXmlResourceParser("AndroidManifest.xml");
```

Kód 6.1: Prístup k súboru *AndroidManifest.xml* v nainštalovanej aplikácii

Algoritmus následne pomocou tohto objektu spracováva jednotlivé xml elementy a vytvára z nich textovú reprezentáciu tohto súboru.

Takto vytvorený xml súbor však nie je čitateľný. Hodnoty veľkej časti atribútov sa odkazujú na zdrojové súbory, ktoré sú skompilované v rámci súboru *resources.arsc*. Príkladom je xml element *application* v nasledujúcej ukážke kódu. Téma na ktorú sa tento element odkazuje je v aplikácii dostupná pod identifikátorom 78954341, a názov je dostupný ako textový zdroj číslo 4562348.

```
<application theme="@78954341" label="@4562348" >
```

Kód 6.2: Ukážka *AndroidManifest.xml* získaného pomocou parsera

Za účelom eliminovania tohto správania využíva *Apk Analyzer* prístup k zdrojovým súborom analyzovaných aplikácií. V týchto súboroch vyhľadá text korešpondujúci s daným identifikátorom a dosadí ho do vytvorenej textovej reprezentácie súboru *AndroidManifest.xml*.

```
<application theme="Theme.YouTube.Light" label="Youtube"/>
```

Kód 6.3: Ukážka *AndroidManifest.xml* po substitúcií identifikátorov zdrojových textov

6.2.3 Súbory v APK balíčku

Na získanie zoznamu všetkých súborov v aplikácii spolu s ich identifikátormi v podobe hashu je využitý súbor *MANIFEST.MF*, ktorý je

vytvorený v priebehu tvorby APK balíčka. Prístup k tomuto súboru je možný prostredníctvom štandardného Java API a triedy *JarFile*.

6.2.4 Zdrojový kód

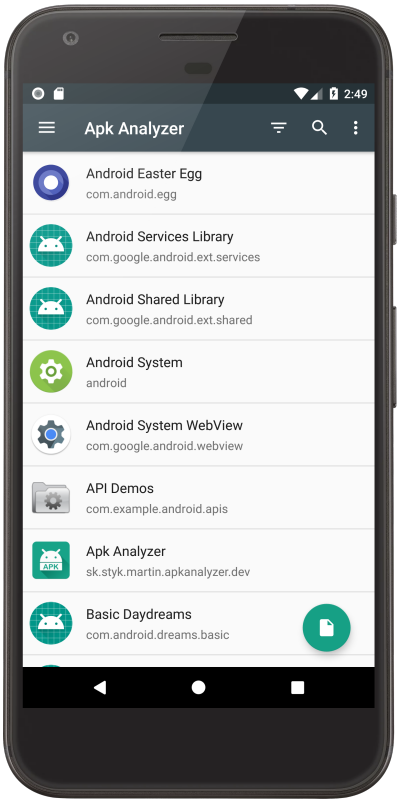
Dáta *DEX* súboru, ktorý obsahuje zdrojový kód, získava aplikácia pomocou triedy *dalvik.system.DexFile*. Táto trieda obsahuje operácie, pomocou ktorých je možné získať zoznam všetkých aplikačných tried, ktoré má daná aplikácia k dispozícií. Tento zoznam obsahuje triedy samotnej aplikácie, spolu so všetkými použitými knižnicami.

6.3 Aplikácia

Mobilná aplikácia bola implementovaná v jazyku Kotlin. Je dostupná v obchode *Google Play* pod názvom *Apk Analyzer*, a je unikátne identifikovaná pomocou mena balíka *sk.styk.martin.apkanalyzer* [28]. Aplikácia je dostupná pre všetky zariadenia s verziou systému novšou ako Android 4.0.3 (API level 15) a je prispôsobená na použitie na mobilných telefónoch a tabletoch. Nasledujúce sekcie opisujú hlavnú funkcionalitu aplikácie.

6.3.1 Zoznam nainštalovaných aplikácií

Základná obrazovka obsahuje zoznam všetkých nainštalovaných aplikačných balíčkov. Tento zoznam sa užívateľovi zobrazí po spustení aplikácie. V zozname aplikácií je možné vyhľadávať podľa lokalizovaného názvu aplikácie alebo mena balíčku. Taktiež je možné filtrovať aplikácie na základe ich pôvodu. Táto funkcia poskytuje užívateľom možnosť ľahko rozlíšiť medzi aplikáciami získanými z oficiálneho obchodu a z alternatívnych zdrojov. Kliknutím na položku zoznamu sa zobrazia detailné informácie o vybranej aplikácii. V prípade analýzy nenainštalovaného aplikačného balíčku vyberie užívateľ APK súbor z pamäte zariadenia pomocou tlačidla umiestneného v pravom dolnom rohu obrazovky.



Obr. 6.1: Zoznam nainštalovaných aplikácií

6.3.2 Analýza aplikácie

Táto obrazovka zobrazuje dáta získané analýzou aplikačného balíčku. Z dôvodu veľkého počtu atribútov získaných analýzou, sú tieto dáta zobrazené vo viacerých záložkách, medzi ktorými môže užívateľ prepínať.
Atribúty analyzovaných aplikácií sú rozdelené tematicky do nasledujúcich skupín:

- *Všeobecné informácie* – základné dáta o aplikácii, napríklad názov, verzia, zdroj alebo kompatibilné verzie systému Android.

- *Informácie o podpise APK balíčka* – dáta získané zo súborov v adresári META-INF. Atribúty v tejto kategórii obsahujú dobu platnosti certifikátu, názov vydavateľa, alebo algoritmus podpisu APK balíčka.

- *Informácie o zdrojových súboroch* – zastúpenie rôznych formátov a veľkostí zdrojových súborov. Tieto informácie je možné získať iba pokiaľ aplikácia nevyužíva obfuskáciu názvov súborov a adresárov.

- *Informácie komponentoch aplikácie* – dáta o aktivitách, službách, poskytovateľoch obsahu a prijímačoch definovaných aplikáciou. V prípade verejne dostupných aktivít poskytuje aplikácia možnosť ich priameho spustenia. Užívateľ tak môže priamo spustiť rôzne časti iných aplikácií.

- *Používané hardvérové vlastnosti* – zoznam vlastností definovaných v súbore *AndroidManifest.xml* v rámci elementu *feature*.

- *Vyžadované bezpečnostné povolenia* – zoznam povolení definovaných v súbore *AndroidManifest.xml* v rámci elementu *uses-permission*.

- *Definované bezpečnostné povolenia* – zoznam povolení definovaných v súbore *AndroidManifest.xml* v rámci elementu *defines-permission*.

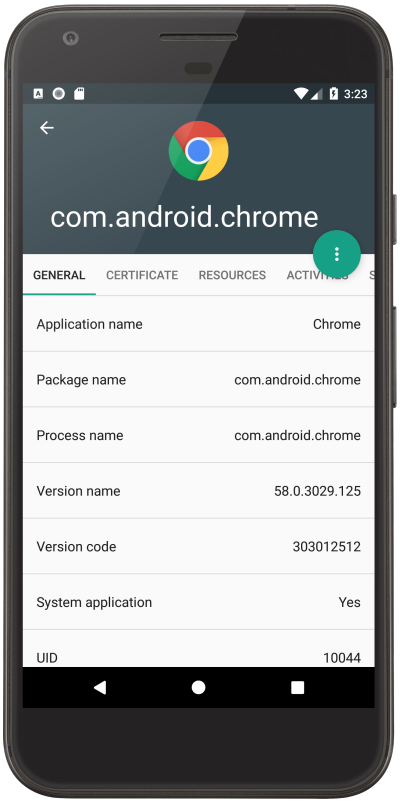
- *DEX súbor* – zoznam všetkých Java a Kotlin tried, ktoré sú zabalené v DEX súbore danej aplikácie.

Kompletný zoznam atribútov získaných pomocou analýzy obsahuje príloha A.

Po kliknutí na položku atribútu analyzovanej aplikácie sa zobrazí popis daného atribútu.

Táto obrazovka poskytuje užívateľovi prístup k ďalším informáciám a operáciám s analyzovanou aplikáciou. Medzi najdôležitejšie patrí možnosť zobrazenia súboru *AndroidManifest.xml* danej aplikácie. Manifest je možné uložiť do externej pamäte zariadenia. Prístup k súboru *AndroidManifest.xml* je možný len pri analýze nainštalovanej aplikácie.

Užívateľ môže exportovať APK súbor nainštalovanej aplikácie. Systém Android ukladá inštalačné súbory všetkých aplikácií do priečinka, ktorý nie je bežným užívateľom prístupný. Pomocou funkcie exportu APK súboru dokáže naša aplikácia skopírovať tento súbor do užívateľovi prístupnej časti úložného priestoru. Aplikácia taktiež poskytuje možnosť zdieľania APK súborov. *Apk Analyzer* sa nezameriava na získavanie dát o konkrétnej inštalácii aplikácie. Tieto dáta poskytuje systémová aplikácia, ktorá je súčasťou štandardných Android nastavení. Naša aplikácia poskytuje možnosť priameho prechodu na túto systémovú aplikáciu. Taktiež poskytuje možnosť zobrazenia záznamu o aplikácii v obchode *Google Play*.



Obr. 6.2: Detaily aplikácie

6.3.3 Analýza aplikácie pred inštaláciou z lokálneho APK súboru

Naša aplikácia poskytuje možnosť analýzy pred inštaláciou APK súborov, ktoré sú uložené v lokálnej pamäti zariadenia. Táto funkcionalita je užitočná v prípade stiahnutia aplikácie z webu a jej následnej inštalácie. Po výbere APK súboru pomocou prieskumníka súborov spustí Android inštaláciu prevzatej aplikácie. V prípade, že má používateľ nainštalovanú našu aplikáciu, nespustí Android inštaláciu automaticky po kliknutí na APK súbor. Namiesto toho zobrazí užívateľovi dialóg, v ktorom si môže vybrať medzi priamou inštaláciou a analýzou v aplikácii *Apk Analyzer*. Naša aplikácia zobrazí detaily APK balíčka a zároveň ponúkne užívateľovi možnosť nainštalovať danú aplikáciu.

6.3.4 Štatistické údaje o nainštalovaných aplikáciách

Aplikácia zobrazuje štatistické informácie získané analýzou všetkých nainštalovaných aplikácií. Primárnym účelom tejto funkcie je poskytnúť užívateľovi zaujímavý štatistický prehľad o kolekcii jeho aplikácií. Štatistické dáta prezentované pomocou grafov zahŕňajú:

- *Kompatibilné verzie systému Android* – minimálna a cieľová verzia Android API
- *Inštalačná politika aplikácie* – preferované umiestnenie aplikácie
- *Algoritmus podpisu balíčka*
- *Pôvod aplikácie* – rozlíšenie medzi predinštalovanými aplikáciami a aplikáciami z oficiálnych a neoficiálnych zdrojov.

Po kliknutí na položku grafu je zobrazený zoznam aplikácií, ktoré patria do danej kategórie.
Nasledujúce dáta sú prezentované pomocou štandardných štatistických ukazovateľov:

- Veľkosť aplikácií
- Počet komponent aplikácií
- Počet súborov
- Počet obrazoviek
- Počet obrázkov

Štatistické dáta o týchto atribútoch zahŕňajú aritmetický priemer, medián, maximálnu a minimálnu hodnotu, rozptyl a smerodajnú odchýlku.

6.3.5 Odosielanie aplikačných metadát na server

Mobilná aplikácia odosiela metadáta o všetkých nesystémových aplikáciách na server, kde sú tieto dáta použité za účelom detekcie prebalených aplikácií.

Informácie o nainštalovaných aplikáciách sú klasifikované ako citlivé údaje. Tieto dáta nie je možné zbierať bez povolenia od užívateľa. Z tohto dôvodu naša aplikácia počas prvého spustenia požiada užívateľa o súhlas s odosielaním metadát o jeho aplikáciách. Užívateľ má neskôr možnosť zmeniť svoje rozhodnutie v sekcii nastavenia.

V prípade, že užívateľ povolí odosielanie dát a zariadenie je pripojené na internet, aplikácia na pozadí analyzuje všetky nesystémové aplikácie a odosiela dáta na server systému *Apk Analyzer*. Tento proces sa spustí pri každom spustení aplikácie. Dáta sú odosielané v serializovanej podobe vo formáte JSON. Keďže tieto dáta obsahujú hashe všetkých súborov v aplikácii, veľkosť odosielaného súboru môže pri väčších aplikáciách dosiahnuť až 1 MB. Z dôvodu efektívneho využitia siete sú tieto dáta skomprimované pomocou algoritmu GZIP, ktorý mnohonásobne zmenší veľkosť prenášaných dát.

Server a centrálna databáza neakceptujú viacero záznamov o jednej verzii aplikácie odoslaných z jedného zariadenia. Na identifikáciu zariadenia slúži *Android Id*, ktoré je taktiež súčasťou odosielaných dát. Za účelom zamedzenia odosielania dát, ktoré už boli z daného zariadenia odoslané, využíva naša mobilná aplikácia lokálnu databázu, do ktorej ukladá meno balíčku a verziu všetkých aplikácií, ktoré boli úspešne odoslané na server. Aplikácia je analyzovaná a odosielaná len v prípade, že ju dané zariadenie ešte neodoslalo.

Komunikácia so serverom prebieha prostrednístvom HTTPS a aplikácia sa musí serveru autentizovať.

6.3.6 Detekcia prebalenej aplikácie

Detekcia prebalených aplikácií prebieha na strane serveru a mobilná aplikácia slúži ako rozhranie, prostredníctvom ktorého ma užívateľ možnosť požiadať o detekciu svojich aplikácií. Táto funkcia je dostupná z menu obrazovky zobrazujúcej detaily aplikačného balíčku. Výsledky detekcie prezentované užívateľovi obsahujú:

- Výsledok – aplikácia je/nie je prebalená, alebo systém nemá dostatok dát na vyhodnotenie,
- Počet identifikovaných nadmieru podobných aplikácií,

- Pomer aplikácií od vydavateľa, od ktorého pochádza užívateľova aplikácia a všetkých identifikovaných podobných aplikácií,
- Pomer aplikácií od vydavateľa, od ktorého pochádza väčšina podobných aplikácií a všetkých identifikovaných podobných aplikácií.

7 Metóda detekcie prebalených aplikácií Apk Analyzer

V rámci práce je navrhnutá metóda detekcie prebalených APK súborov. Navrhnutá metóda stavia na poznatkoch existujúcich výzkumov, a je založená predovšetkým na podobnosti zdrojových súborov obsiahnutých v aplikáciách. Základom pre novonavrhnutú metódu sú práce *FSquaDRA* a *ImageStruct*. Primárnym cieľom novej metódy je vytvoriť spôsob detekcie prebalených aplikácií, ktorý je pomocou mobilnej aplikácie prístupný pre použitie širokej škále užívateľov systému Android.

7.1 Motivácia a ciele

Medzi existujúcimi riešeniami bolo identifikovaných niekoľko možných vylepšení, ktoré návrh novej metódy zohľadňuje. Tieto vylepšenia súvisia predovšetkým so sprístupnením tejto metódy používateľom. Navrhnutá metóda sa zameriava na vylepšenie nasledujúcich oblastí:

- Určenie, ktorá aplikácia je originál a ktorá prebalená kópia,
- Detekcia prebalených aplikácií pochádzajúcich výhradne z alternatívnych obchodov,
- Kolaborácia užívateľov za účelom spresnenia detekčnej metódy,
- Dynamická aktualizácia dát potrebných na detekciu prebalených súborov.

Sprístupnenie užívateľom

Známe metódy detekcie prebalených APK súborov (vrátane metód predstavených v kapitole 4) vykonávajú tzv. offline analýzu dát. Tieto metódy pracujú nad existujúcou databázou súborov, nad ktorou vykonávajú operácie s cieľom identifikácie prebalených dvojíc. Offline metódy slúžia predovšetkým na vedecké účely, no môžu byť použité priamo obchodmi s aplikáciami. V súčasnosti však mnohé obchody

vrátane *Google Play* nie sú efektívne zabezpečené proti prebaleným aplikáciám [22]. Užívatelia sú nútení chrániť svoje Android zariadenia pomocou antivírových programov. Napriek tomu, že niektoré antivírové a zabezpečovacie aplikácie môžu detekovať hrozby v podobe prebalených aplikácií, v súčasnosti (podľa vedomia autora) nie je rozšírená žiadna metóda zameriavajúca sa priamo na detekciu prebalených APK súborov na Android zariadení.

Metóda navrhnutá v rámci projektu *Apk Analyzer* predstavuje pre užívateľov ďalšiu možnosť ako ochrániť svoje zariadenie pred škodlivými aplikáciami. Cieľom je sprostredkovať užívateľom mechanizmus detekcie prebalených aplikácií priamo z ich zariadenia. Tento spôsobo detekcie nebezpečných aplikácií môže byť použitý ako doplnok k existujúcim nástrojom.

Určenie originálnej aplikácie a využitie pri alternatívnych zdrojoch

Existujúce metódy detekcie prebalených aplikácií sú primárne zamerané na určenie vzťahu medzi dvomi aplikáciami. Takéto metódy dokážu detekovať nadmernú podobnosť dvojice aplikácií a rozhodnúť, či pochádzajú od rôznych autorov. Nástroje častokrát nedokážu určiť, ktorá z týchto aplikácií je originál, a ktorá prebalená kópia. Väčšina nástrojov využíva predpoklad, že aplikácia pochádzajúca z oficiálneho zdroja *Google Play Store* je pravá a aplikácie z alternatívnych zdrojov sú prebalené. Tento predpoklad však vo všeobecnosti platiť nemusí. Niektoré aplikácie sú distribuované výhradne pomocou alternatívnych kanálov. Existujúce metódy neumožňujú rozlíšenie medzi originálom a kópiou, v prípade, že obe porovnávané aplikácie pochádzajú z alternatívnych zdrojov. Keďže našim základným cieľom je ponúknuť detekciu prebalených aplikácií koncovému užívateľovi, je výsledok detekcie bez rozhodnutia ktorá aplikácia je originál a ktorá prebalená kópia nedostačujúci.

Cieľom je poskytnúť užívateľovi odpoveď, ktorá ho informuje či je jeho aplikácia podľa systému *Apk Analyzer* prebalená. Zámerom metódy *Apk Analyzer* je využiť heuristiku počtu výskytov jednotlivých verzií aplikácií s prirodzeným predpokladom, že na Android zariadeniach sa nachádza väčší počet originálnych aplikácií, ako ich prebalených kópií.

Dynamickosť a kolaborácia

Predstavené offline metódy pracujú nad statickou kolekciou aplikácií. Metódy využívané priamo v obchodoch s aplikáciami pracujú nad kolekciou aplikácií v danom obchode. Za účelom vytvorenia metódy vychádzajúcej z *FSquaDRA* a *ImageStruct* je nutné vytvoriť databázu aplikácií. Táto databáza však musí reflektovať zmeny v aktuálne dostupných aplikáciách, pretože nové verzie a nové aplikácie sú vydávané dennodenne.

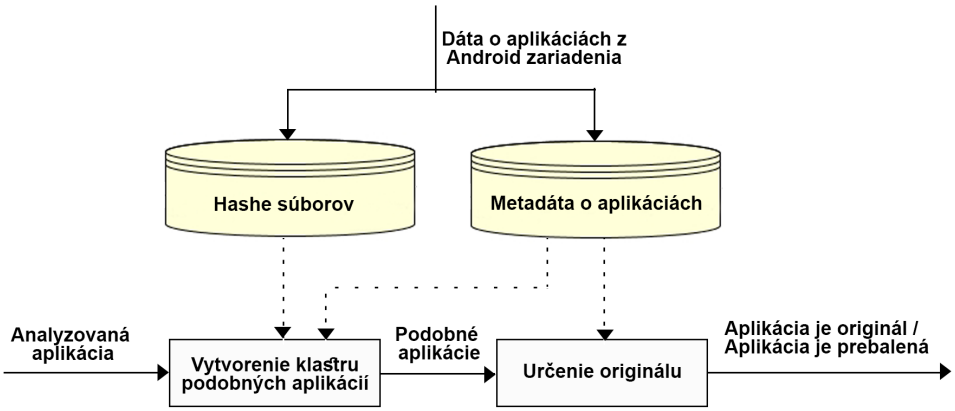
Cieľom systému *Apk Analyzer* je vytvorenie dynamickej databázy dát o aplikáciách, na základe ktorej môže byť uskutočnená detekcia prebalených súborov. Za týmto účelom je zámerom systému *Apk Analyzer* zapojiť do tvorby databázy existujúcich aplikácií koncových užívatelov. Mobilná aplikácia, ktorá sprostredkováva detekciu prebalených aplikácií odošle dáta o aplikáciách nainštalovaných na danom zariadení do centrálnej databázy systému *Apk Analyzer*. Tento kolaboratívny spôsob zaručí neustálu aktualizáciu databázy. Čím viacej užívateľov je zapojených, tým väčšiu databázu sa podarí vytvoriť.

7.2 Návrh metódy

Návrh kompletného systému *Apk Analyzer*, ktorý pozostáva z mobilného klienta a serverovej časti obsahuje kapitola 5 . Táto kapitola sa zameriava predovšetkým na samotnú detekčnú metódu a tvorbu databázy, ktoré sa odohrávajú na strane serveru.

7.2.1 Základný princíp detekcie prebalených aplikácií

Mobilná aplikácia *Apk Analyzer* odosiela dáta o aplikáciách nainštalovaných na jednotlivých Android zariadeniach na server, ktorý ich ukladá do databázy. Pri detekcii sa použijú dáta uložené v databáze. K analyzovanej aplikácii sa vyhľadajú podobné aplikácie. Podobnosť je určená ako pomer obrázkov, ktoré dve aplikácie zdieľajú. Z kolekcie aplikácií, ktoré zdieľajú veľkú časť obrázkov, a teda sú považované za rôzne verzie tej istej aplikácie je vybraný najčastejšie sa opakujúci digitálny podpis, ako identifikátor originálnej verzie. Ak je analyzovaná aplikácia podpísaná týmto kľúčom, je označená za dôveryhodnú. V opačnom prípade systém upozorní užívateľa na prebalenú aplikáciu.



Obr. 7.1: Postup metódy Apk Analyzer

7.2.2 Databáza aplikácií

Základom detekcie prebalených aplikácií je databáza metainformácií o aplikáciách. Dáta o aplikáciách pochádzajú z analýzy, ktorá je vykonaná na Android zariadeniach. Detailné informácie týkajúce sa analýzy aplikácií pomocou mobilnej aplikácie *Apk Analyzer* sa nachádzajú v kapitole 6. Mobilná aplikácia odosiela metadáta o každej nesystémovej aplikácii na server.
Tento prístup poskytuje nasledujúce výhody:

- *Rozloženie výpočtovej záťaže* – výpočtová záťaž spojená s analýzou a reverzným inžinierstvom APK súborov je distribuovaná medzi mnohé Android zariadenia. Každé zariadenie analyzuje svoje nainštalované aplikácie. Hromadná centralizovaná analýza veľkého množstva aplikácií je výpočtovo a časovo náročná činnosť. Server nevykonáva dekompiláciu aplikácie, ale len ukladá dáta, ktoré mu odošle Android zariadenie.

- *Dynamický vývoj databázy* – klientska aplikácia odosiela na server všetky nesystémové aplikácie, vrátane aktualizovaných verzií.

To zabezpečuje, že databáza aplikácií sa dynamicky vyvíja a obsahuje aj najnovšie aplikácie. Z dôvodu udržovania aktuálnosti databázy nie je potrebný žiadny zásah administrátora.

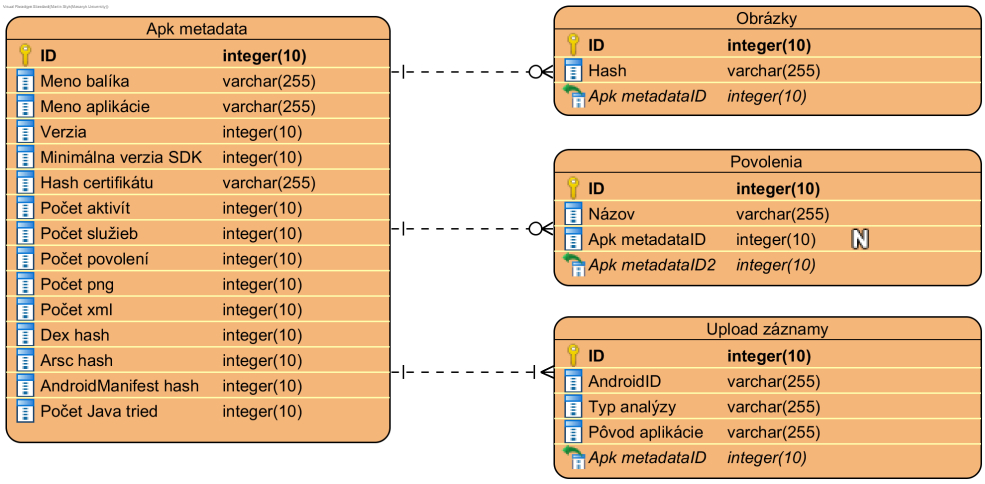
Odosielané dáta o nainštalovaných aplikáciách sú podmnožinou dát získaných analýzou APK súboru. Tieto dáta obsahujú základné metadáta ako napríklad meno balíka aplikácie, veľkosť APK súboru, hash certifikátu alebo počty jednotlivých komponent aplikácie. Okrem toho sú odosielané aj hashe všetkých rastrových obrázkových súborov obsiahnutých v APK balíčku a taktiež bezpečnostné povolenia využívané aplikáciou. Kompletný zoznam všetkých uchovávaných atribútov aplikácie obsahuje príloha B.

Z dôvodu efektivity vyhľadávania v databáze je pre každú aplikáciu určený jej unikátny identifikátor. V prípade, že majú dve aplikácie tento identifikátor rovnaký, ide o úplne rovnakú aplikáciu a v databáze ju server ukladá len raz.
Na tvorbu tohto identifikátoru sú použité nasledovné atribúty APK balíčka:

- *meno balíka aplikácie* – z pohľadu systému Android identifikuje aplikáciu. Pokiaľ majú dve aplikácie rovnaké meno balíka, jedná sa o rôzne verzie tej istej aplikácie,
- *hash certifikátu aplikácie* – unikátne identifikuje autora aplikácie,
- *hash súboru Classes.dex* – reprezentácia zdrojového kódu aplikácie,
- *hash súboru Resources.arsc* – reprezentácia skompilovaných zdrojových súborov aplikácie,
- *hash súboru AndroidManifest.xml* – reprezentácia hlavného metasúboru,
- *kombinovaný hash všetkých obrázkov* – identifikuje kolekciu obrázkov aplikácie,
- *počet obrázkových súborov*,
- *počet xml súborov*.

V prípade, že majú dve aplikácie rovnaký identifikátor, je zaručené, že pochádzajú od jedného vydavateľa, ich zdrojový kód, skompilované zdrojové súbory, metadáta a všetky obrázky sú identické.

Dôležitým aspektom pri detekcii prebalených aplikácií je počet výskytov jednotlivých verzií aplikácií. Túto metriku systém využíva pri určení, ktoré aplikácie sú originály a ktoré prebalené imitácie. Preto je dôležité identifikovať, z akého zariadenia dáta o aplikácii pochádzajú. Za týmto účelom je použitý unikátny identifikátor zariadenia *AndroidID*. Server si ku každej aplikácii pamätá všetky zariadenia, z ktorých bola daná aplikácia nahraná, a taktiež uchováva informácie o pôvode aplikácie.



Obr. 7.2: Dátový model ukladajúci informácie o aplikáciách

Návrh dátového modelu zodpovedá štruktúre ukladaných dát. Vzhľadom k požiadavkám kladeným na systém Apk Analyzer je použité riešenie prezentované na ERD diagrame 7.2. Pri tvorbe dátového modelu je potrebné zohľadniť požadovanú výkonnosť aplikácie. Za účelom optimalizácie výkonnosti nemá dátový model normalizovanú dátovú schému. Niektoré atribúty v tabuľke *Apk metadata* sú duplicitné, a môžu byť získané pomocou iných tabuliek. Príkladom takéhoto atribútu je počet obrázkov v danej aplikácii. Tento údaj je možné zistiť dotazom na tabuľku *Obrázky*, no z dôvodu optimalizácie výkonnosti je uložený aj ako atribút tabuľky *Apk metadata*. Ďalším porušením

normalizácie je vzťah medzi tabuľkou *Apk metadata* a *Obrázky*, respektíve *Apk metadata* a *Povolenia*. V systéme môže nastať situácia, že jeden súbor, prípadne jedno povolenie je zdieľané medzi viacerými aplikáciami. V tom prípade by mal byť medzi týmito entitnými množinami použitý vzťah $N : N$. Takýto prístup by vyžadoval použitie spojovacej tabuľky, čo by malo negatívny vplyv na výkonnosť. Nad databázou vykonáva systém *Apk Analyzer* iba operácie dotazovania a vkladania nových záznamov. Negatívne dôsledky nenormalizovanej databázovej schémy by spôsobili problémy predovšetkým pri operáciách *update* a *delete*. Medzi negatíva zvoleného prístupu patrí zvýšená veľkosť ukladaných dát. Tieto nedostatky sú však vyvážené vyššou výkonnosťou. Vloženie jedného záznamu o APK súbore do existujúcej databázy trvá približne 100 ms. Pri použití normalizovanej schémy zaberie táto operácia monohonásobne viac času.

7.2.3 Detekcia prebalených aplikácií

Základným obmedzením kladeným na metódu detekcie prebalených APK súborov je využitie dát, ktoré sú získané analýzou vykonávanou klientskou aplikáciou priamo na Android zariadení. S tým je spojených viacero limitácií. Metódy využívajúce analýzu zdrojového kódu aplikácií sú výpočtovo náročné. Extrakcia dát zo súboru *classes.dex* a vytvorenie identifikátora aplikácie na základe týchto dát predstavuje netriviálnu výpočtovú záťaž [6]. Viaceré výzkumy ukázali, že metódy detekcie prebalených aplikácií založené na podobnosti zdrojových súborov dosahujú podobné výsledky ako metódy analýzy zdrojového kódu [25–27]. Spôsob implementovaný v rámci systému *Apk Analyzer* využíva zhodu obrázkových súborov. Tento princíp kombinuje metódy *FSquaDRA* a *ImageStruct* (viď kapitola 4).

Na rozdiel od metódy *ImageStruct* naša metóda neextrahuje z obrázkových súborov dáta, ale využíva hash celého súboru. Porovnávanie je teda založené na presnej binárnej zhode. Získanie charakteristických informácií obrázka podobne ako v metóde *ImageStruct* nie je možné, pretože klientska aplikácia nemá priamy prístup k obrázkovým súborom iných aplikácií. Preto sa naša metóda spolieha na predpoklad, že pri prebaľovaní aplikácií sa obrázkové súbory často nemenia. V porovnaní s metódou *FSquaDRA* nepoužíva *Apk Analyzer* pri porovnaní všetky súbory, ale len rastrové obrázky.

Samotný algoritmus detekcie prebalených súborov dostane na vstup aplikáciu, o ktorej má rozhodnúť, či je prebalená alebo nie. Algoritmus sa skladá z viacerých krokov.

Detekcia klastru podobných aplikácií

V prvom kroku algoritmus nájde všetky aplikácie, ktoré spĺňajú definované kritériá zhody aplikácií. Tieto kritériá pozostávajú z podobnosti metadát a zhody obrázkových súborov. Podobnosť metadát je použitá za účelom zefektívnenia vyhľadávania podobných aplikácií. Algoritmus sa rozhoduje na základe metadát, ktoré nemôžu byť pri prebaľovaní ľahko zmenené, respektíve z výsledkov predchádzajúcich výskumov vyplýva, že ich útočníci často nemenia.
Algoritmus používa nasledujúce atribúty:

- počet *xml* súborov v aplikácii
- počet obrázkových súborov v aplikácii
- celkový počet súborov
- počet Java tried v aplikácii

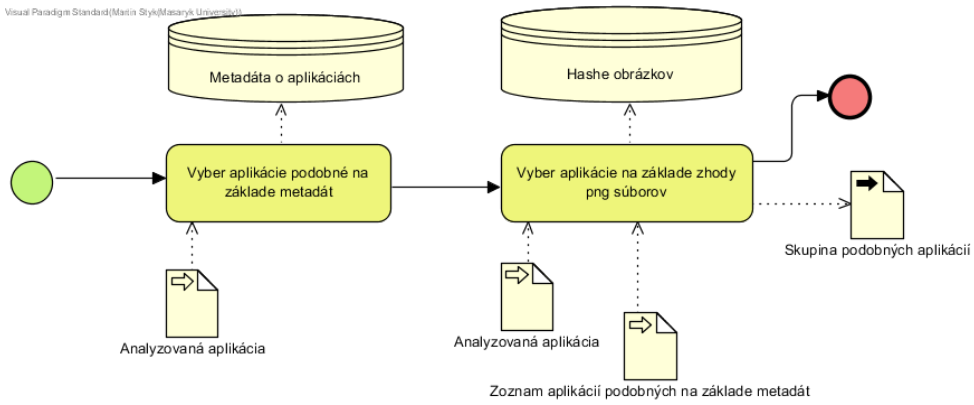
Všetky z týchto atribútov sú číselné. Ako podobné aplikácie sú na základe týchto atribútov označené také aplikácie, ktorých hodnoty týchto atribútov sú v rozmedzí ± 20 % od hodnôt analyzovanej aplikácie. Hodnota 20 % bola určená na základe experimentov nad databázou APK súborov. Tieto pokusy ukázali ukázali, že zmena tohto parametru na vyššie hodnoty nemá zásadný vplyv na výsledky detekcie. Pri nastavení povoleného rozmedzia na 50 % stúpol počet detekovaných aplikácií o 3%.

Aplikácie, ktorých metadáta sú vyhodnotené ako podobné, sú následne porovnané s aplikáciou ktorej detekcia prebieha. Toto porovnanie je založené na zhode obrázkových súborov v týchto APK balíčkoch. Koeficient podobnosti je určený pomocou *Jaccardovej metódy*.

$$similarity(app_1, app_2) = \frac{|app_1 images \cap app_2 images|}{app_1 images}$$

Hodnota Jaccardovho koeficientu, nad ktorou sú aplikácie považované za zhodné je odvodená z metódy *FSquaDRA* a naväzujúcich prác. Algoritmus APK balíčky ako rôzne verzie jednej aplikácie, ak je Jaccardov index podobnosti vyšší ako 0,7.

Výstupom prvej časti algoritmu je skupina aplikácií, ktoré algoritmus klasifikuje ako nadmieru podobné. Takéto aplikácie považujeme za rôzne druhy jednej aplikácie.



Obr. 7.3: Postrup tvorby klastru podobných aplikácií

Identifikácia originálu

V predchádzajúcom kroku identifikuje algoritmus nadmieru podobné aplikácie. Aplikácia, ktorej detekcia prebieha je s aplikáciami z tejto skupiny zvyčajne v jednom z nasledujúcich vzťahov:

- rôzne verzie jednej aplikácie
- upravené / prebalené aplikácie
- iné (false positive)

V takejto skupine sa môžu vyskytovať aj aplikácie, ktoré nie sú inou verziou alebo kópiou aplikácie, ktorej detekcia prebieha. Popis možných chýb a nedostatkou tejto metódy obsahuje sekcia 7.4.

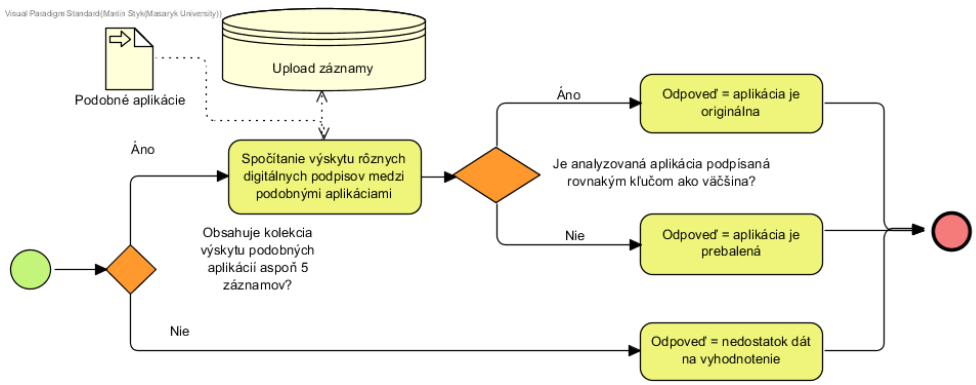
Na základe digitálneho podpisu APK balíčka vieme rozhodnúť, do ktorej z týchto kategórií aplikácia patrí. V prípade zhody certifikátu,

ktorým sú aplikácie podpísané, ide o dve rôzne verzie jednej aplikácie, ktoré pochádzajú od jedného vydavateľa. V prípade podpisu rôznymi certifikátmi považujeme jednu z týchto aplikácií za prebalenú, pretože aplikácie pochádzajú od rôznych vydavateľov a na základe ich nadmiernej podobnosti sú považované za jednu logickú aplikáciu.

Metóda detekcie prebalených aplikácií implementovaná v rámci systému *Apk Analyzer* využíva heuristiku založenú na početnosti výskytu aplikácií. Základom tejto metódy je pozorovanie, že na Android zariadeniach je nainštalovaných viac originálnych aplikácií, ako ich prebalených kópií.

Aplikácie sú rozdelené do skupín podľa ich digitálneho podpisu. Aplikácia, ktorej detekcia prebieha, je označená ako originálna v prípade, že je podpísaná kľúčom, ktorým je podpísaná väčšina APK súborov v klastri podobných aplikácií. V opačnom prípade vyhodnotí algoritmus túto aplikáciu ako prebalenú kópiu. Pri určovaní počtu aplikácií sa zohľadňuje počet výskytu jednotlivých verzií na rôznych zariadeniach.

V prípade, že kolekcia podobných aplikácií neobsahuje záznamy minimálne o piatich aplikáciách prítomných na rôznych zariadeniach, proces je ukončený s odpoveďou oznamujúcou nedostatok dát pre vykonanie spoľahlivej detekcie pôvodu aplikácie.



Obr. 7.4: Postup určenia výsledku detekcie

7.3 Predpoklady navrhnutej metódy

Navrhnutá metóda využíva pri detekcii prebalených aplikácií viacero dôležitých predpokladov.

V obehu je viac originálnych aplikácií ako prebalených

Tento predpoklad vychádza zo základného princípu prebalených aplikácií, ktoré využívajú popularitu originálnej aplikácie za účelom šírenia škodlivej funkcionality.

Obrázkové súbory je možné použiť na detekciu prebalených aplikácií

Metóda detekcie prebalených APK súborov *ImageStruct* založená na porovnaní vlastností obrázkov ukázala, že výskyt rovnakých obrázkov je silným indikátorom prebalenia aplikácie. Funkčnosť a presnosť tohto prístupu bola overená prostredníctvom porovnania s metódou založenou na analýze zdrojového kódu [25]. Podobné zistenia sú prezentované aj v rámci práce *FSquaDRA*, ktorá detekuje prebalené kópie na základe porovnania všetkých súborov aplikácií [26]. V rámci rozširujúcej evaluácie metódy *FSquaDRA* bolo zistené, že presnosť detekcie je možné zvýšiť pomocou vzájomného porovnanie súborov rovnakého typu [27]. Tento prístup sme použili aj v metóde *Apk Analyzer*. Viacero známych metód sa pri identifikácií typov súborov spolieha na štandardnú štruktúra APK balíčka. Práca vyhodnocujúca metódu *FSquaDRA* využíva predpoklad, že obrázkové súbory sa nachádzajú v štandardnom priečinku *res/drawable*. Súborová štruktúra APK balíčka môže byť predmetom obfuskácie, ktorú využívajú vývojári za účelom skomplikovania reverzného inžinierstva, a útočníci prebaľujúci aplikácie za účelom odlíšenia kópie od originálu. Z tohto dôvodu používa náš prístup podobnosť všetkých súborov vo formátoch *png*, *bmp*, *webp*, *jpg* a *gif*. Intuícia za týmto návrhom je založená na pozorovaní, že pri obfuskácii môžu byť zmenené názvy priečinkov a súborov, avšak typ súboru zostáva rovnaký. Tabuľka 7.3 ukazuje zastúpenie formátov obrázkových súborov v analyzovanej vzorke 16 700 aplikácií.

Formát	Priemerný počet	Medián
png	369	144
jpg	7	0
gif	0,4	0
bmp	0,4	0
bmp	0,1	0

Tabuľka 7.1: Obrázkové súbory v APK balíčku

Pri prebaľovaní aplikácií sú obrázkové súbory nepozmenené

Navrhnutá metóda detekcie porovnáva obrázky na základe ich hashov. Aj malá úprava obrázku spôsobí kompletnú zmenu jeho hashu. Preto je pre navrhnutú metódu detekcie kľúčové, aby obrázky v prebalenej aplikácií zostali nezmenené. Skúmaním zmien vykonávaných pri prebaľovaní aplikácie sa zaoberala Olga Gadyatskaya a kolektív v roku 2016 [27]. V tomto období boli už k dispozícií viaceré metódy využívajúce na detekciu prebalených aplikácií podobnosť zdrojových súborov, vrátane metód *FSquaDRA* a *ImageStruct*. Napriek tomu výsledky výzkumu ukázali, že pri prebaľovaní zostávajú zdrojové súbory zvyčajne nemodifikované a útočníci najčastejšie modifikujú súbor *classes.dex* a *AndroidManifest.xml*. Tento fakt môžeme vysvetliť tak, že metódy založené na podobnosti zdrojových súborov nie sú v praxi veľmi rozšírené a útočníci ich nepovažujú za hrozbu ani napriek ich preukázanej kvalite. Preto sa pri prebaľovaní sústredia najmä na zmeny v zdrojovom kóde, a ostatné súbory zostávajú nepozmenené. Nasledujúca tabuľka 7.3 zobrazuje typy súborov a pravdepodobnosť, že súbor daného typu je modifikovaný v prebalenej aplikácii.

Nastavenia detekcie na základe predchádzajúcich výskumov

Známe výskumy prebalených APK súborov pracujú nad offline databázou aplikácií. Funkčnosť a presnosť týchto metód je vyhodnotená prostredníctvom porovnania výsledkov s inými algoritmami. Na základe experimentov sú určené parametre detekcie, ktoré eliminujú nesprávne výsledky algoritmu. V prípade našej metódy je situácia komplikovanejšia, pretože našu databázu netvoria celé APK súbory,

Pravdepodobnosť modifikácie	Typ súboru
0,0212	audio video
0,0624	raw
0,0731	obrázok
...	
0,8476	Resources.arsc
0,9066	AndroidManifest.xml
0,9227	zdrojový kód

Tabuľka 7.2: Zmeny v prebalených APK súboroch

ale len niektoré metadáta. Nie je teda možné spustiť nad týmito dátami iné metódy detekcie, a nastaviť tak parametre algoritmu, ktoré ovplyvňujú výsledky detekcie. Z metadát o jednotlivých aplikáciách nie je možné rozhodnúť, či je daná aplikácia originál, alebo prebalená verzia. Z toho dôvodu sa navrhnutá metóda vo veľkej miere opiera o už existujúce výskumy a preberá ich závery a odporúčané parametre detekcie.

7.4 Hodnotenie

Navrhnutá metóda detekcie prebalených APK súborov bola prakticky implementovaná a nasadená v produkčnom prostredí systému *Apk Analyzer*. Metóda je postavená na základoch aktuálnych poznatkov, ktoré odhaľujú, že pri prebaľovaní aplikácií zvyčajne zostávajú zdrojové súbory nezmenené. V prípade rozšírenia metód využívajúcich podobnosť zdrojových súborov je pravdepodobné, že útočníci začnú aplikovať výraznejšie zmeny v rôznych súboroch obsiahnutých v APK balíčku. Výrazná zmena počtu súborov v balíčku alebo úpravy obsiahnutých obrázkov predstavujú spôsob, akým je možné navrhnutú metódu znefunkčniť. Avšak v súčasnosti takéto zmeny vykonávané nie sú a navrhnutá metóda je schopná odhaliť prebalené aplikácie.

TODO => kolko detekcii, kolko hrozieb, pustit manualne detekciu kazdy z kazdym a vyhodnotit

TODO => popis mozn7ch problemov - zdielane kniznice a podobne

Riešenie je implementované nad databázou *Postgres*. Pri aktuálnom objeme dát, ktoré aplikácia obsahuje je výkonnosť SQL databázového servera dostačujúca. V prípade, že systém Apk Analyzer bude aj naďalej rásť, je možné využiť na ukladanie zoznamu hashov obrázkov systém *Redis*, ktorý výrazne zvýši rýchlosť detekcie prebalených aplikácií.

8 Záver

Primárny cieľom práce bolo vytvorenie softvérového riešenia na detekciu prebalených aplikácií pre mobilný operačný systém Android. Cieľom práce bolo sprístupniť metódu detekcie potenciálne škodlivých aplikácií užívateľom prostredníctvom mobilnej aplikácie, ktorá poskytuje detailné informácie o nainštalovaných aplikačných balíčkoch. Za účelom dosiahnutia tohto cieľa sa táto práca zaoberala dvomi hlavnými témami. Prvou z nich bol vývoj mobilnej aplikácie, ktorá získava metadáta o ostatných aplikáciách. Druhú tému predstavuje metóda detekcie prebalených aplikácií.

V rámci tejto práce bola vyvinutá mobilná aplikácia *Apk Analyzer*. Táto aplikácia poskytuje užívateľovi detailné informácie o aplikačných balíčkoch nainštalovaných na jeho zariadení. Na rozdiel od ostatných dostupných aplikácií ponúkajúcich podobnú funkcionalitu, umožňuje naša aplikácia analýzu APK súborov, ktoré nie sú nainštalované. Aplikácia taktiež poskytuje zrozumiteľné vysvetlenie prezentovaných dát a ponúka užívateľom možnosť detekcie prebalených APK súborov. Aplikácia *Apk Analyzer* (*sk.styk.martin.apkanalyzer*) je dostupná prostredníctvom obchodu *Google Play*. Od prvého vydania aplikácie v októbri 2017 si ju nainštalovalo viac ako 10 000 (TODO) užívateľov. Kvalitu aplikácie dokumentuje jej hodnotenie v službe *Google Play*. Priemerné hodnotenie dosahovalo v februári 2018 hodnotu 4,7 hviezdičiek (maximum je 5) (TODO update).

V práci bola navrhnutá nová metóda detekcie prebalených aplikácií. Metóda je založená na základoch aktuálnych poznatkov, ktoré odhaľujú, že pri prebaľovaní zvyčajne zostávajú zdrojové súbory nezmenené. Na rozdiel od väčšiny existujúcich prístupov je naša metóda schopná identifikovať, ktorá z aplikácií je originálna a ktoré sú prebalené kópie. Táto metóda pracuje s dynamicky sa vyvíjajúcou databázou metadát o aplikáciách. Na tvorbe tejto databázy sa podieľajú užívatelia mobilnej aplikácie *Apk Analyzer*. Navrhnutá metóda detekcie prebalených APK súborov bola prakticky implementovaná a nasadená v produkčnom prostredí systému *Apk Analyzer*. Od svojho spustenia v TODO zaznamenala TODO požiadavkov na detekciu prebalených aplikácií, z toho TODO aplikácií bolo vyhodnotených ako nebezpečné.

Spojením mobilnej aplikácie a serverovej časti vzniklo softvérové riešenie, umožňujúce detekciu prebalených aplikácií priamo v Android zariadení. Obidve tieto časti boli nasadené v produkčnom prostredí a sú dostupné užívateľom. Zdrojový kód aplikácií vytvorených v rámci tejto práce je voľne prístupný.

Literatúra

1. *Media Types* [online]. [cit. 2017-10-21]. Dostupný z: ⟨http://www.iana.org/assignments/media-types/media-types.xhtml⟩.
2. ALLEN, Grant. Inside Your First Android Project. 2015, s. 35–44. Dostupný z: ⟨https://doi.org/10.1007/978-1-4302-4687-9_3⟩.
3. *App Manifest* [online]. 2015 [cit. 2017-10-21]. Dostupný z: ⟨http://developer.android.com/guide/topics/manifest/manifest-intro.html⟩.
4. *Dalvik Executable format* [online]. [cit. 2017-10-21]. Dostupný z: ⟨https://source.android.com/devices/tech/dalvik/dex-format⟩.
5. *Providing Resources* [online]. [cit. 2017-10-21]. Dostupný z: ⟨https://developer.android.com/guide/topics/resources/providing-resources.html⟩.
6. ZHOU, Wu; ZHOU, Yajin; JIANG, Xuxian; NING, Peng. Detecting repackaged smartphone applications in third-party android marketplaces. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*. 2012, s. 317–. Dostupný z: ⟨http://dl.acm.org/citation.cfm?doid=2133601.2133640⟩.
7. [online]. [cit. 2017-10-21]. Dostupný z: ⟨https://developer.android.com⟩.
8. *Application Signing* [online]. [cit. 2017-12-19]. Dostupný z: ⟨https://source.android.com/security/apksigning/⟩.
9. *Sign Your App* [online]. [cit. 2017-12-19]. Dostupný z: ⟨https://developer.android.com/studio/publish/app-signing.html%5C#manage-key⟩.
10. LONG, Lei; SONG, Yang; PAN, Aimin; XIAO, Peng. What can you do to an apk without its private key except repacking? [online]. [Cit. 2017-12-19]. Dostupný z: ⟨https://www.blackhat.com/docs/ldn-15/materials/%20london-15-Xiao-What-Can-You-Do-To-An-APK-Without-Its-%20Private-Key-wp.pdf⟩.
11. ISHII, Yuta; WATANABE, Takuya; AKIYAMA, Mitsuaki; MORI, Tatsuya. Clone or Relative? *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics - IWSPA '16*. 2016, s. 25–32. Dostupný z: ⟨http://dl.acm.org/citation.cfm?doid=2875475.2875480⟩.

12. *Apktool: A tool for reverse engineering Android apk files* [online]. [cit. 2017-12-21]. Dostupný z: ⟨https://ibotpeaches.github.io/Apktool/⟩.
13. *Dex2jar* [online]. [cit. 2017-12-21]. Dostupný z: ⟨https://github.com/pxb1988/dex2jar⟩.
14. *JD-GUI: Yet another fast Java decompiler* [online]. [cit. 2017-12-21]. Dostupný z: ⟨http://jd.benow.ca/⟩.
15. *Smali* [online]. [cit. 2017-12-21]. Dostupný z: ⟨https://github.com/JesusFreke/smali⟩.
16. LUO, Symphony; YAN, Peter. Fake Apps: Feigning Legitimacy. Dostupný z: ⟨https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-fake-apps.pdf⟩.
17. *Android "Master Key" vulnerability – more malware exploits code verification bypass* [online]. 2013 [cit. 2017-12-21]. Dostupný z: ⟨https://nakedsecurity.sophos.com/2013/08/09/android-master-key-vulnerability-more-malware-found-exploiting-code-verification-bypass/⟩.
18. *Number of available applications in the Google Play Store from December 2009 to September 2017* [online]. 2017 [cit. 2017-12-21]. Dostupný z: ⟨https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/⟩.
19. ZHOU, Wu; ZHOU, Yajin; JIANG, Xuxian; NING, Peng. Detecting repackaged smartphone applications in third-party android marketplaces. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*. 2012, s. 317–. Dostupný z: ⟨http://dl.acm.org/citation.cfm?doid=2133601.2133640⟩.
20. ZHOU, Wu; ZHOU, Yajin; GRACE, Michael; JIANG, Xuxian; ZOU, Shihong. Fast, scalable detection of "Piggybacked"mobile applications. *Proceedings of the third ACM conference on Data and application security and privacy - CODASPY '13*. 2013, s. 185–. Dostupný z: ⟨http://dl.acm.org/citation.cfm?doid=2435349.2435377⟩.
21. BALLANO, Mario. Android Threats Getting Steamy. *Symantec Official Blog*. Roč. 2011. Dostupný z: ⟨https://www.symantec.com/connect/blogs/android-threats-getting-steamy⟩.
22. ZHAUNIAROVICH, Yury; GADYATSKAYA, Olga; CRISPO, Bruno. DEMO: Enabling trusted stores for android. 2013, s. 1345–1348. Dostupný z: ⟨http://dl.acm.org/citation.cfm?doid=2508859.2512496⟩.

23. HURLBUT, Dustin. Fuzzy Hashing for Digital Forensic Investigators. [online]. [Cit. 2017-12-21]. Dostupný z: ⟨https://repo.zenk-security.com/Techniques%5C%20d.attaques%5C%20%5C%20.%5C%20%5C%20Failles/Fuzzy%5C%20Hashing%5C%20for%5C%20Digital%5C%20Forensic%5C%20Investigators.pdf⟩.
24. *pHash: The open source perceptual hash library* [online]. 2008 [cit. 2017-12-21]. Dostupný z: ⟨http://www.phash.org/⟩.
25. JIAO, Sibei; CHENG, Yao; YING, Lingyun; SU, Purui; FENG, Dengguo. A Rapid and Scalable Method for Android Application Repackaging Detection. *Information Security Practice and Experience*. 2015, s. 349–364. Dostupný z: ⟨http://link.springer.com/10.1007/978-3-319-17533-1_24⟩.
26. ZHAUNIAROVICH, Yury; GADYATSKAYA, Olga; CRISPO, Bruno; LA SPINA, Francesco; MOSER, Ermanno. FSquaDRA: Fast Detection of Repackaged Applications. In: *Data and Applications Security and Privacy XXVIII: 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, s. 130–145. Dostupný z: ⟨https://doi.org/10.1007/978-3-662-43936-4_9⟩. ISBN 978-3-662-43936-4.
27. GADYATSKAYA, Olga; LEZZA, Andra-Lidia; ZHAUNIAROVICH, Yury. Evaluation of Resource-Based App Repackaging Detection in Android. In: *Secure IT Systems: 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings*. Cham: Springer International Publishing, 2016, s. 135–151. Dostupný z: ⟨https://doi.org/10.1007/978-3-319-47560-8_9⟩. ISBN 978-3-319-47560-8.
28. *Apk Analyzer*. Dostupný z: ⟨https://play.google.com/store/apps/details?id=sk.styk.martin.apkanalyzer⟩.
29. *App Detective*. Dostupný z: ⟨https://play.google.com/store/apps/details?id=com.zmarties.detective⟩.
30. *Apk Analyzer*. Dostupný z: ⟨https://play.google.com/store/apps/details?id=jp.zproject.apkanalyzer⟩.

A Dáta získané analýzou APK súboru

Atribút	Dátový typ
meno balíka aplikácie	string
názov aplikácie	string
názov procesu aplikácie	string
názov verzie	string
kód verzie	integer
systémová aplikácia	boolean
UID	integer
popis	string
zdroj aplikácie	string
ikona	drawable
umiestnenie APK súboru	string
umiestnenie aplikačných dát	string
veľkosť APK súboru	integer (bajty)
najnišia podporovaná verzia SDK	integer
cieľová verzia SDK	integer
prvá inštalácia	timestamps
posledná aktualizácia	timestamps
algorimus podpisu balíčka	string
hash verejného kľuča	string
hash certifikátu	string
sériové číslo certifikátu	integer
začiatok platnosti certifikátu	timestamps
koniec platnosti certifikátu	timestamps
vydavateľ certifikátu	string
organizácia vydavateľa certifikátu	string
krajina vydavateľa certifikátu	string
subjekt podpisu	string

organizácia subjektu podpisu	string
krajina subjektu podpisu	string
vyžadované oprávnenia	zoznam stringov
definované oprávnenia	zoznam stringov
hash classes.dex	string
hash resources.arsc	string
hash AndroidManifest.xml	string
hashe obrazoviek	zoznam stringov
hashe obrázkov	zoznam stringov
hashe menu	zoznam stringov
hashe ostatných súborov	zoznam stringov
počet png súborov	integer
počet png súborov s rôznym názvom	integer
počet xml súborov	integer
počet xml súborov s rôznym názvom	integer
počet jpg súborov	integer
počet gif obrázkov	integer
počet nine patch obrázkov	integer
počet xml obrázkov	integer
počet obrázkov	integer
počet rôznych obrázkov	integer
počet obrazoviek	integer
počet rôznych obrazoviek	integer
počet ldpi obrázkov	integer
počet mdpi obrázkov	integer
počet hdpi obrázkov	integer
počet xhdpi obrázkov	integer
počet xxhdpi obrázkov	integer
počet xxxhdpi obrázkov	integer
počet nodpi obrázkov	integer

počet tvdpi obrázkov	integer
aplikačné triedy	zoznam stringov
počet aplikačných tried (bez vnútorných tried)	integer
dáta o aktivitách	štruktúra
dáta o službách	štruktúra
dáta o poskytovateľoch obsahu	štruktúra
dáta o prijímačoch	štruktúra

B Dáta ukladané v centrálnej databáze APK súborov

Atribút	Dátový typ
identifikátor	integer
meno balíka aplikácie	string
názov aplikácie	string
názov verzie	string
kód verzie	integer
veľkosť APK súboru	integer (bajty)
najnišia podporovaná verzia SDK	integer
cieľová verzia SDK	integer
algorimus podpisu balíčka	string
hash verejného kľuča	string
hash certifikátu	string
sériové číslo certifikátu	integer
počet aktivít	integer
identifikátor aktivít	integer
počet služieb	integer
identifikátor služieb	integer
počet content provider	integer
identifikátor content provider	integer
počet broadcast receiver	integer
identifikátor broadcast receiver	integer
počet definovaných oprávnení	integer
identifikátor definovaných oprávnení	integer
počet vyžadovaných oprávnení	integer
identifikátor vyžadovaných oprávnení	integer
počet využívaných vlastností	integer
identifikátor využívaných vlastností	integer

hash classes.dex	string
hash resources.arsc	string
hash AndroidManifest.xml	string
počet obrázkov	integer
počet obrazoviek	integer
počet menu	integer
celkový počet súborov	integer
počet png súborov	integer
počet png súborov s rôznym názvom	integer
počet xml súborov	integer
počet xml súborov s rôznym názvom	integer
identifikátor png súborov	integer
identifikátor xml súborov	integer
identifikátor obrazoviek	integer
identifikátor menu	integer
počet jpg súborov	integer
počet ldpi obrázkov	integer
počet mdpi obrázkov	integer
počet hdpi obrázkov	integer
počet xhdpi obrázkov	integer
počet xxhdpi obrázkov	integer
počet xxxhdpi obrázkov	integer
počet nodpi obrázkov	integer
počet tvdpi obrázkov	integer
počet aplikačných tried	integer
počet aplikačných tried (bez vnútorných tried)	integer
identifikátor aplikačných tried	integer
hashe png súborov	zoznam stringov
vyžadované oprávnenia	zoznam stringov