Slovenská Technická Univerzita v Bratislave
Fakulta Elektrotechniky a Informatiky
Ústav Informatiky a Matematiky

Evidenčné číslo: FEI-104372-35362

Ing. Juraj Varga

ODHAĽOVANIE MALVÉRU V OS ANDROID

Dizertačná práca

Bratislava, 2018 **Ing. Juraj Varga**

Slovenská Technická Univerzita v Bratislave
Fakulta Elektrotechniky a Informatiky
Ústav Informatiky a Matematiky

Ing. Juraj Varga

Dizertačná práca

Téma dizertačnej práce:	Odhaľovanie malvéru v OS Android
Školiteľ:	prof. Ing. Pavol Zajac, PhD.
Forma štúdia:	denná
Začiatok štúdia:	1.9.2011
Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9 Aplikovaná informatika

2018

MIESTO TEJTO STRANY POJDE ZADANIE PRACE

Anotácia

Slovenská technická univerzita v Bratislave

Fakulta Elektrotechniky a Informatiky

Autor dizertačnej práce:	Ing. Juraj Varga
Vedúci dizertačnej práce:	prof. Ing. Pavol Zajac, PhD.
Názov dizertačnej práce:	Odhaľovanie malvéru v OS Android
Študijný program:	Aplikovaná informatika

15. 10. 2018

Táto práca predstavuje písomnú časť dizertačnej práce Odhaľovanie malvéru v OS Android. Mobilné aplikácie zažívajú rozmach od uvedenia mobilných operačných systémov na trh. Každým rokom ich počet stúpa a pracujú s čoraz väčším objemom citlivých používateľských dát. Preto stúpa potreba ich bezpečnosti, kontroly a zvyšovania bezpečnostného povedomia používateľov. Kľúčovým prvkom zodpovedným za prístup k systémovým zdrojom je v OS Android systém povolení. V našom výskume sme najprv podrobne preštudovali architektúru a štruktúru OS Android. Následne sme spracovali prehľad používaných bezpečnostných mechanizmov, vrátane systému povolení, a ich súvis s architektúrou OS Android. Na základe získaných poznatkov sme si za hlavné ciele výskumu vytýčili možnosť úpravy systému povolení a následne tento systém použiť ako základ nástroja na analýzu a detekciu malvéru (resp. zraniteľných aplikácií) v prostredí OS Android.

Práca je štrukturovaná nasledovne: úvod uvedie čitateľa do problematiky. Prvá kapitola pojednáva najskôr stručne o bezpečnostných modeloch v rôznych operačných systémoch. Následne plynule prejde k podrobnému popisu architektúry OS Android a súvisiacich bezpečnostných mechanizmov. Druhá kapitola je zameraná na aktuálny výskum v oblasti bezpečnosti OS Android. V tretej kapitole je predstavený návrh a implementačné detaily jednoduchej klient-server aplikácie na detekciu malvéru na základe vyžadovaných povolení. Štvrtá kapitola sa venuje popisu testovacej metodiky navrhnutého riešenia a prezentácii dosiahnutých výsledkov.

Kľúčové slová: Android, detekcia malvéru, bezpečnosť, povolenia, mobilná aplikácia

Annotation

Slovak University of Technology Bratislava

Faculty of Electrical Engineering and Information Technology

Author:	Ing. Juraj Varga
Supervisor:	prof. Ing. Pavol Zajac, PhD.
Final project:	Malware detection among Android apps
Degree course:	Applied Informatics

15. 10.2018

This thesis presents a written part of Dissertation thesis named Malware detection among Android apps. Mobile applications are on the rise since mobile operating systems were introduced. Each year their number grows and they work with increasing amount of sensitive user data. Therefore the need for their security, various checks and user awareness rises too. The key element responsible for managing access to system resources is in Android OS the permission model. In our research we firstly thoroughly examined Android structure and architecture. Following that, we created an overview of currently used security mechanisms (including the permission model) and their connection with Android architecture. Based on this knowledge, we set the goals of our research which are: possibilities of permission model modification, and using this system as a foundation of a tool used to analyze and detect malware (or vulnerable apps) in Android ecosystem.

Our work is organized as follows: introduction introduces the reader to the topic. The first part firstly deals briefly with security models in various operating systems. It follows up with thorough examination of Android architecture and related security mechanisms. The second chapter focuses on current research dealing with Android security. At the end of this chapter, the goals of this thesis are specified. The design and implementation details of simple client-server malware detection application based on required permissions are presented in the third chapter. The fourth chapter deals with testing methodology of proposed solution and presents the results we achieved.

Keywords: Android, malware detection, security, permissions, mobile application

Čestné vyhlásenie

Čestne vyhlasujem, že som túto písomnú prácu vypracoval sám s použitím uvedenej literatúry. V prípade vlastných publikácií je spoluautorstvo explicitne uvedené.

V Bratislave, 15. 10. 2018

.....

Ing. Juraj Varga

Pod'akovanie

Chcem sa pod'akovať všetkým, ktorí mi nedovolili to už dávno zabaliť. Každý z Vás prispel svojou troškou k dokončeniu tejto práce, a za to som Vám zo srdca vď'ačný. Bez Vás by som to nezvládol.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

ACL - Access Control List
ADB - Android Debug Bridge
AEAD - Authenticated Encryption with Associated Data
AES - Advanced Encryption Standard
AJAX - Asynchronous Javascript and XML
API - Application Programming Interface
ASLR - Address Space Layout Randomization
CA - Certification Authority
CBC - Cipher Block Chaining
CFS - Completely Fair Scheduler
CGM - CyanoGen Mod
CPU - Central Processing Unit
CSS - Cascading Style Sheet
DAC - Discretionary Access Control
DACL - Discretionary Access Control List
DES - Digital Encryption Standard
DLL - Dynamic Link Library
DOS - Denial of Service
DRM - Digital Rights Management
DVM - Dalvik Virtual Machine
ECC - Elliptic Curve Cryptography
ECDSA - Elliptic Curve Digital Signature Algorithm
EDGE - Enhanced Data for GSM Evolution
ESSIV - Encrypted Salt-Sector Initilization Vector
GID - Group IDentification
GP - Google Play
GPL - General Public License
GPRS - General Packet Radio Service
GPS - Global Positioning System
GSM - Global System for Mobile Communication
GUi - Graphical User Interface
HAL - Hardware Abstraction Layer
HDMI - High Definition Media Interface
HKLM - Hkey_Local_Machine
HTTP(S) - HyperText Transfer Protocol (Secure)
ICC - Inter Component Communication
IMEI - International Mobile Equipment Identity
IMSI - International Mobile Subscriber Identity
IPC - Inter-Process Communication
JPA - Java Persistent API
JSON - JavaScript Object Notation

JVM - Java Virtual Machine
LSASS - Local Security Authority SubSystem
LSM - Linux Security Module
L2TP - Layer 2 Tunneling Protocol
MAC - Mandatory Access Control
MB - Mega Byte
MD5 - Message Digest version 5
MITM - Man-in-the-Middle
MMS - Multimedia Messaging Service
MMU - Memory Management Unit
NFC - Near Field Communication
OEM - Original Equipment Manufacturer
OHA - Open Handset Alliance
OS - Operating System
PBKDF - Password Based Key Derivation Function
PC - Personal Computer
PDA - Personal Digital Assistant
PEP - Proactive Exploit Protection
PIN - Personal Identification Number
PPTP - Point to Point Tunneling Protocol
RAM - Random Access Memory
RBAC - Role Based Access Control
RFID - Radio Frequency Identification
RIL - Radio Interface Layer
ROM - Read-Only Memory
RSA - Rivest, Shamir & Adleman
SACL - System Access Control LIst
SAM - Security Accounts Manager
SD - Secure Digital
SDK - Software Development Kit
SHA - Secure Hash Algorithm
SMS - Short Message Service
SQL - Structured Query Language
SRM - Security Reference Monitor
SSL - Secure Sockets Layer
SVM - Support Vector Machines
TCP - Transport Control Protocol
TCSEC - Trusted Computer System Evaluation Criteria
TLS - Transport Layer Security
UID - Unique IDentification
USB - Universal Serial Bus
VPN - Virtual Private Network
Wi-Fi - Wireless Fidelity
XML - Extensible MArkup Language

XMPP - Extensible Messaging and Presence Protocol

Úvod

Operačný systém Android je najrýchlejšie sa rozvíjajúci OS pre mobilné platformy. Pod pojmom mobilné platformy môžeme v súčasnosti chápať (relatívne) malé zariadenia s vysokým výpočtovým výkonom - chytré telefóny (v tejto práci budeme používať termín smartfóny), tablety a PDA.

Ich typickou črtou je ovládanie dotykovým displejom. Bývajú osadené viacjadrovými procesormi, veľkou operačnou pamäťou a úložným priestorom. Samozrejmosťou sú rôzne komunikačné rozhrania - USB, HDMI, Bluetooth a bezdrôtové pripojenie na Internet či už prostredníctvom bezdrôtových sietí (Wi-Fi) alebo mobilného operátora. Rozmach týchto zariadení sa začal v roku 2008 a odvtedy ich počet každoročne stúpa.

S počtom dostupných zariadení ruka v ruke rastie aj dopad a počet rôznych bezpečnostných hrozieb. Podobne ako sa kedysi pri rozmachu klasických počítačov pravidelne vyskytovali rôzne nové vírusy a chyby, tak podobný trend môžeme sledovať v súčasnosti na mobilných zariadeniach. Bezpečnostné hrozby sa plynule presúvajú na mobilné zariadenia z niekoľkých dôvodov:

- Nachádza sa na nich veľké množstvo osobných a citlivých informácií (prihlasovacie mená a heslá, čísla účtov a platobných kariet . . .).
- Povedomie používateľov o týchto hrozbách nie je také vysoké ako pri klasických PC.
- Android ako open source systém je vhodnou platformou pre "amatérskych" vývojárov.
- Takýto rýchly nástup a rozmach nebol pravdepodobne očakávaný, a príprava na možné riziká teda nebola možná.
- Vďaka open source politike systému absentuje podrobná kontrola vydávaných aplikácií.

Napriek snahe odborníkov o ich elimináciu, počet bezpečnostných chýb a hrozieb neklesá, ale minimálne sa drží na rovnakej úrovni. Z prehľadu aktuálnych zraniteľností a hrozieb sme si vytýčili tri konkrétne ako tézy dizertačnej práce:

- Model povolení a možnosti jeho modifikácie vzhľadom na konkrétnu aplikáciu, či používateľa. Tento model má v aktuálnej podobe viacero nedostatkov, ktoré by mohli byť odstránené drobnými úpravami v operačnom systéme Android. Možnosť výberu povolení, ktoré budú pridelené konkrétnej aplikácii alebo používateľovi, môže výrazne obmedziť realizáciu viacerých útokov, napr. krádež používateľských dát.

- Analýza mobilného malvéru a spôsoby jeho detekcie. Podobne ako pri malvéri zameranom na klasické OS, aj v tejto oblasti badáme rýchly rozvoj. Autori škodlivých aplikácií flexibilne reagujú na zmeny v OS Android, či iné techniky zamerané na zvýšenie jeho bezpečnosti.

- Dodatočné úpravy aktuálnych bezpečnostných mechanizmov tak, aby boli prispôsobené upravenému modelu povolení, resp. aby zvýšili používateľský komfort. Príkladom môže byť jednoduchší a bezpečnejší spôsob odomykania zariadenia.

Štruktúra tejto práce je nasledovná:

1. Úvod a prvé dve kapitoly predstavujú teoretický úvod do problematiky. Postupne prejde od popisu prístupových modelov, používaných v rôznych operačných systémoch, k detailnej architektúre OS Android a podrobnejšiemu prehľadu bezpečnostných mechanizmov tohto operačného systému. Na tieto poznatky priamo nadväzuje prehľad súčasného stavu problematiky.

2. Tretiu kapitolu tvorí prehľadný návrh riešenia spolu s UML diagramami a technické detaily jeho implementácie, vrátane bližšieho popisu najdôležitejších komponentov.

3. Štvrtú kapitolu tvorí popis testovacej metodiky, testovanej vzorky aplikácií, nastavení systému a dosiahnuté výsledky.

4. Prácu uzatvára záverečné zhodnotenie výsledkov dizertačnej práce.

Kapitola 1

Bezpečnostné mechanizmy v OS Android

Predtým, ako sa začneme zaoberať bezpečnostnými mechanizmami v OS Android, musíme si priblížiť všeobecné prístupové modely používané v klasických (desktopových) operačných systémoch. Mnohé tieto mechanizmy sú v OS Android priamo implementované, alebo z nich vychádzajú, resp. sú ich modifikáciou.

1.1 Prístupové modely

Kontrola prístupu je v OS vykonávaná zvyčajne autentifikáciou hlavných aktérov pomocou rôznych mechanizmov (heslá, Kerberos) a následným priradením systémových zdrojov, prístupom k súborom či komunikačným rozhraniam [1]. V individuálnych prípadoch, hlavne na začiatku počítačovej histórie, sa používala na označenie úrovne prístupu trojica *read, write a execute, resp. znak "-"*. Tento spôsob je aktuálne funkčný, ale dá sa povedať, že už nie je postačujúci. Ďalším spôsobom kontroly je používanie tzv. *skupín a rôl*. Skupina či rola v sebe zahŕňa rôzne entity (používatelia, programy), ktoré majú na základe príslušnosti k tejto skupine či role definované svoje práva a povinnosti. Začali sa využívať pri rozmachu počítačových sietí vo veľkých spoločnostiach na oddelenie úrovní prístupu pre zamestnancov. Veľké vylepšenie priniesli tzv. *Access Control Listy - ACL*, ktoré v sebe špecifikujú povolenie na prístup a operácie ku konkrétnym zdrojom alebo súborom. Bližšie si k týmto metódam a ich napojeniu na konkrétne OS povieme v nasledujúcich podkapitolách.

1.1.1 Linux

Najdôležitejšie mechanizmy a ochrany operačného systému Linux boli vytvorené pred viac ako 30 rokmi. Tieto ochrany ale už nie sú bezpečné proti dnešným škodlivým kódom. Základné riadenie prístupu v Linuxovej bezpečnosti môžeme rozdeliť do hlavných troch modelov [3] [4]:

DAC model

V modeli DAC (Discretionary Access Control) je prístup založený na identite používateľa. Je to najmenej obmedzujúci model. Aby používateľ získal práva k prístupu, tak musí byť umiestnený do ACL zoznamu, teda do zoznamu riadenia prístupu. Ak je používateľ alebo skupina vlastníkom objektu, môže udeľovať práva ďalším používateľom alebo skupinám. DAC umožňuje úplnú kontrolu nad objektmi. To ale spôsobuje jeho náchylnosť k bezpečnostným hrozbám. Po prvé umožňuje koncovému používateľovi absolútnu kontrolu nad bezpečnostnými nastaveniami ostatných používateľov, čo môže viesť k tomu že používatelia získajú vyššie práva, ako by mali mať. Po druhé, ak používateľ spustí program, tak ten zdedí jeho práva a to môže viesť k spusteniu škodlivého kódu bez toho, aby o tom daný používateľ niečo vedel.

MAC model

V modeli MAC (Mandatory Access Control) je prístup zabezpečený na základe overenia totožnosti subjektov alebo skupín, do ktorých patria. V tomto modeli práva k objektom alebo zdrojom môže udeliť len administrátor. To znamená, že koncový používateľ nemá žiadnu kontrolu nad nastaveniami žiadnych privilégií. V súčasnosti existujú dva bezpečnostné modely spojené s MAC a to sú Biba [5] a Bell-LaPadula [6]. Model Biba sa zameriava na integritu informácií, zatiaľ čo Bell-LaPadula na dôvernosť informácií. Na rozdiel od DAC reguluje prístup používateľov ku zdrojom pomocou bezpečnostnej politiky, ktorá je založená na úrovni zabezpečenia objektu. Táto politika je súbor pravidiel, ktoré určujú, aké prístupy sú povolené v systéme. V modeli Biba objekt s nižšou úrovňou zabezpečenia môže čítať informácie z vyššej úrovne zabezpečenia a objekty s vyššou úrovňou zabezpečenia môžu zapisovať informácie do nižších úrovní. Tento model je využívaný hlavne v podnikoch, kde zamestnanci s nižšou úrovňou môžu čítať informácie z vedenia, teda z vyššej úrovne, ktoré môže zapisovať, aby informovalo nižšie úrovne. V modeli Bell-LaPadula objekt s vyššou úrovňou môže zapisovať iba na danej úrovni, ale zato môže čítať informácie aj z nižších úrovní. Bell-LaPadula bol vyvinutý pre vládne a vojenské účely.

RBAC model

Model RBAC (Role Based Access Control) obsahuje role priradené k používateľovi. V tomto modeli administrátor pridelí rolu používateľovi s danými právami a privilégiami, teda nepridieľuje konkrétne práva a privilégiá, ale rolu, kde sú zahrnuté. RBAC model zjednodušuje prácu administrátorom. Problém však nastáva vtedy, ak jeden používateľ danej role potrebuje prístup k súborom, ktoré tejto roli nie sú prístupné. Po pridelení oprávnení tomuto používateľovi ich získava celá rola, vrátane používateľov, pri ktorých to nemusí byť bezpečné.

Problémy a vylepšenia bezpečnostných modelov v OS Linux

OS Linux má implementovanú jednoduchú, ale efektívnu kontrolu DAC. Táto kontrola nie je napriek svojej efektivite považovaná za dostatočnú. Kontrola je definovaná prostredníctvom štandardných prístupových práv, ktoré sú aplikované na objekty v súborovom systéme. Každý objekt obsahuje informáciu o vlastníkovi a skupine: UID a GID. Podľa toho sa definujú práva vlastníka, skupiny a používateľov systému - čítanie, zápis a spustenie. Neskôr Linux kvôli bezpečnosti tento základný bezpečnostný model rozšíril o POSIX (jadro v2.2) a o ACL, ktoré určujú, kto alebo čo môže pristupovať k objektu a aké operácie s ním vykonávať. Jeden z najväčších problémov je používanie super používateľa - root. Tento používateľ má plnú moc nad správou systému. Hoci bolo navrhnuté množstvo vylepšení modelov riadenia prístupu i rámcov a boli aj úspešne realizované a otestované, tradičné OS stále nemajú podporu týchto vylepšení. Táto absencia bezpečnosti je spôsobená tým, že doteraz nebolo v rámci bezpečnostnej komunity dohodnuté, ktorý z týchto modelov je správny.

Avšak Linux dlhodobo podporuje dynamické načítanie modulov jadra, predovšetkým pre ovládače zariadení, ale aj pre ďalšie komponenty, ako sú súborové systémy. Dnes sú rozšírenia riadenia prístupu vykonávané pomocou modulov linuxového jadra, čo umožnilo vytvorenie rôznych bezpečnostných modelov. Vytváranie efektívnych bezpečnostných modulov je problematické, pretože jadro neposkytuje žiadnu infraštruktúru umožňujúcu modulom jadra sprostredkovať prístup k jadrovým objektom. Z tohto dôvodu mnoho projektov zaviedlo rozšírené rámce pre riadenie prístupu alebo modelov pre linuxové jadro ako záplaty do jadra. Tzv. Linux Security Module (LSM) poskytuje iba mechanizmus na presadzovanie politiky rozšírenia riadenia prístupu. LSM moduly implementujú osobitnú politiku a sú rozhodujúce pre preukázanie funkcie jadra.

1.1.2 Windows

OS Windows od verzie NT 4.0 je držiteľom C2 bezpečnostnej klasifikácie – stredné zabezpečenie proti škodlivému softvéru a útokom. Prístupový model Windows je založený na diskrétnej kontrole prístupu [2] [4]. Bezpečnostný deskriptor pre zaistiteľné objekty môže obsahovať dva typy ACL - DACL a SACL:

- **Discretionary Access Control List (DACL)** - identifikuje administrátorov, ktorí majú alebo nemajú prístup k zaistiteľnému objektu. Keď proces skúša pristúpiť k zaistiteľnému objektu, systém kontroluje ACE v objektoch DACL aby určil či procesu pridelí prístup. Access Control Entry (ACE) je element v ACL. Ak objekt nemá DACL, systém pridelí plný prístup všetkým. Ak DACL objekt nemá žiadne ACE, systém zamietne všetky požiadavky pre prístup k objektu, pretože DACL nepovolí žiadne prístupové práva. Systém kontroluje ACE postupne až kým nenájde jeden alebo viac ACE, ktoré povoľujú všetky požadované prístupové práva, alebo kým nejaké z požadovaných prístupových práv nie sú zamietnuté.

- **System Access Control List (SACL)** - umožňuje administrátorom zaznamenávať pokusy o prístup k zaistiteľným objektom. Každý ACE špecifikuje typy prístupových pokusov pomocou špecifikovaného administrátora, čo spôsobí že systém vygeneruje záznam do bezpečnostného záznamu. ACE v SACL môže generovať dopytové záznamy keď pokus o prístup zlyhá, nezlyhá alebo oboje. Systémové objekty, súbory, nastavenia registru a kernelové objekty sú chránené ACL. Tieto zoznamy sú však zraniteľné na chyby používateľov a programátorov. Preto Windows obsahuje mechanizmus tzv. úrovní integrity, ktoré slúžia ako sekundárny systém na kontrolu prístupu. Objekty a procesy obdržia označenie integrity a to nízka, stredná alebo vysoká. Windows nepovoľuje procesu meniť objekt vyššej integrity ako on sám bez ohľadu na nastavenie ACL.

Komponenty systémovej bezpečnosti

- **Security Reference Monitor (SRM)** - súčasť *Ntoskrnl.exe*, zodpovedná za:
  - Definovanie dátových štruktúr prístupových tokenov určených na reprezentáciu bezpečnostného kontextu.
  - Vykonávanie kontroly prístupov k objektom.
  - Manipulovanie privilégií a generovanie správ bezpečnostných auditov.
- **Local Security Authority Subsystem (LSASS)** - proces používateľského módu využívajúci obraz *Lsass.exe*, ktorý je zodpovedný za (väčšina tejto funkcionality je implementovaná v knižnici *Lsasrv.dll* , ktorú LSASS využíva.):
  - Lokálnu systémovú bezpečnostnú politiku.
  - Zoznam používateľov, ktorí sa môžu prihlásiť do systému.
  - Politika hesiel.
  - Privilégia udelené používateľom a skupinám.
  - Nastavenie systémových bezpečnostných auditov.
  - Autentifikáciu používateľov.
  - Odosielanie auditov do Event Logu.
- **LSASS politiková databáza** - databáza obsahujúca nastavenia lokálnej bezpečnostnej politiky. Databáza je uložená v HKLM. Obsahuje zoznam dôveryhodných domén, ktoré vykonávajú autentifikáciu loginov, zoznam kto má povolenie na prístup do systému a ako.
- **Security Accounts Manager (SAM)** - služba zodpovedná za manažovanie databázy obsahujúcej mená používateľov a skupín definovaných na lokálnom počítači. Táto služba je implementovaná v *Samsrv.dll* .

- **SAM databáza** - databáza obsahujúca definované mená používateľov a skupín spolu s ich heslami a ostatnými atribútmi. Databáza je uložená v registri pod HKLM.

- **AppLocker** - mechanizmus povoľujúci administrátorom špecifikovať, ktoré spúšťateľné súbory, DLL knižnice a skripty môžu byť využívané jednotlivými používateľmi a skupinami. AppLocker pozostáva s ovládača *AppId.sys* a služby *AppId-Svc.dll* bežiacich v rámci procesu *SvcHost*.

- **ASLR** - náhodné rozloženie adresného priestoru. ASLR slúži ako ochrana pred viacerými útokmi tým, že zabraňuje malým úsekom vloženého kódu aby sa dostali do kódu, ktorý je práve nahrávaný do procesu ako súčasť normálnej operácie. Táto bezpečnostná zábrana spôsobí, že systém pod takýmto útokom zlyhá resp. spadne skôr, ako by prišlo k jeho ovládnutiu týmto kódom.

- **PEP** - proaktívna ochrana proti exploitom. Novšie čipy Intel aj AMD sú založené na AMD64, ktorá dovoľuje označenie stránkam v pamäti, že nemôžu obsahovať spustiteľný inštrukčný kód. Windows sa preto snaží takto označiť dátovú pamäť, aby sa nedala využiť na spustenie kódu. Zabraňuje to tak útokom, v ktorých kvôli chybe programu príde k pretečeniu zásobníka, a potom je program oklamaný a spustí obsah zásobníka.

1.1.3 Ďalšie spôsoby kontroly prístupu

Okrem mechanizmov obsiahnutých priamo v OS existujú aj rôzne dodatočné ochrany vyvíjané buď pre špecifické použitie alebo na využitie špecializovaného hardvéru [1]. Do tejto kategórie môžeme zaradiť:

- Middleware: pod týmto pojmom v práci budeme chápať softvér, ktorý slúži na prístup k nejakému produktu. Jedná sa hlavne o rôzne perzistentné vrstvy a prepojenia, napr. medzi databázou a samotnou aplikáciu, ktorá nad ňou beží. Ako súčasť bezpečnosti je to dobrý koncept, ktorý však naráža na viacero problémov. Jedným z nich je konzistencia a spolupráca viacerých takýchto systémov od rôznych výrobcov. Z tohoto dôvodu sa v praxi zo začiatku nakupovali riešenia od jedného výrobcu, aby neboli problémy s kompatibilitou. Dnes sú tieto riešenia väčšinou štandardizované a ich spolupráca je výrazne lepšia. Ďalším problémom môže byť ich nedostatočná granularita, kde dochádza k zamedzeniu prístupu k dátam a znefunkčneniu celého riešenia.

- Sandboxing a Proof-Carrying kód: Sandboxing slúži na spúšťanie potenciálne nebezpečného kódu v bezpečnom prostredí, kde nemá prístup k systémovým zdrojom a môže komunikovať len s používateľom. Proof-Carrying kód je kód, ktorý na svoje spustenie potrebuje mať v sebe dôkaz, že nekompromituje lokálnu bezpečnostnú politiku.

- Virtualizácia: Pod týmto pojmom chápeme možnosť behu viacerých OS na jednom zariadení. Je veľmi atraktívna pre rôzne hostingy a má tiež uplatnenie v počítačovej bezpečnosti. Slúži napríklad na testovanie podozrivých súborov v bezpečnom prostredí, keďže virtuálny OS nemá prístup k fyzickému hardvéru ani k vonkajšiemu prostrediu. Takisto slúži na oddeľovanie súkromných a pracovných dát na firemných zariadeniach - na prácu je OS s viac reštriktívnou politikou a na súkromné účely s menej reštriktívnou politikou. Častým problémom je však potreba používateľov prenášať dáta, a je tak možné preniesť škodlivé súbory do bezpečného prostredia napr. na USB kľúči.

- Trusted Computing: táto iniciatíva vzišla za účelom predstavenia takej platformy, kde by si bol používateľ istý, že program, ktorý spúšťa, je bezpečný. Hardvér aj softvér by sa v tomto prípade správali vždy rovnako, a toto správanie by bolo vynucované implementovaným hardvérovým čipom. Jeho prednosťou je zvýšenie celkovej bezpečnosti, možnosť vzájomnej autentifikácie zariadení a vyššia ochrana pred škodlivým kódom.

1.2 Vlastnosti OS Android

OS Android je momentálne najrozšírenejším operačným systémom pre mobilné telefóny, tablety a iné zariadenia [7]. Jeho vývoj začal v roku 2003 v spoločnosti Android Inc. O dva roky neskôr túto spoločnosť kúpil Google a roku 2007 bola jeho prvá verzia predstavená verejnosti. Prvý telefón so systémom Android bol uvedený na trh spoločnosťou HTC v USA v roku 2008, na Slovensko sa dostal v lete roku 2009. Medzi výhody tohto systému patrí otvorenosť, prostredie a možnosti ktoré ponúka. [8][12]

Základnými vlastnosťami OS Android (aj ako mobilnej platformy) sú:

- Otvorenosť operačného systému a platformy ako celku - zdrojové kódy linuxového jadra sú voľne dostupné na Internete [15].

- Open source komunita - konzorcium OHA (Open Handset Alliance) ako majiteľ licencie k OS Android poskytuje všetkým jeho používateľom neobmedzenú tvorbu vlastných aplikácií, ich zdieľanie a predaj[16], [21].

- Systémové aktualizácie - OS Android je aktualizovaný na novú verziu zhruba raz ročne [14].

- Adaptabilita na rôzne veľkosti a rozlíšenia displeja zariadenia.

- Vysoká konektivita - podpora všetkých súčasných komunikačných technológií (GSM, EDGE, 3G, Wi-Fi, Bluetooth, GPRS ...), pripojenie k Internetu, podpora video hovorov, posielanie správ.

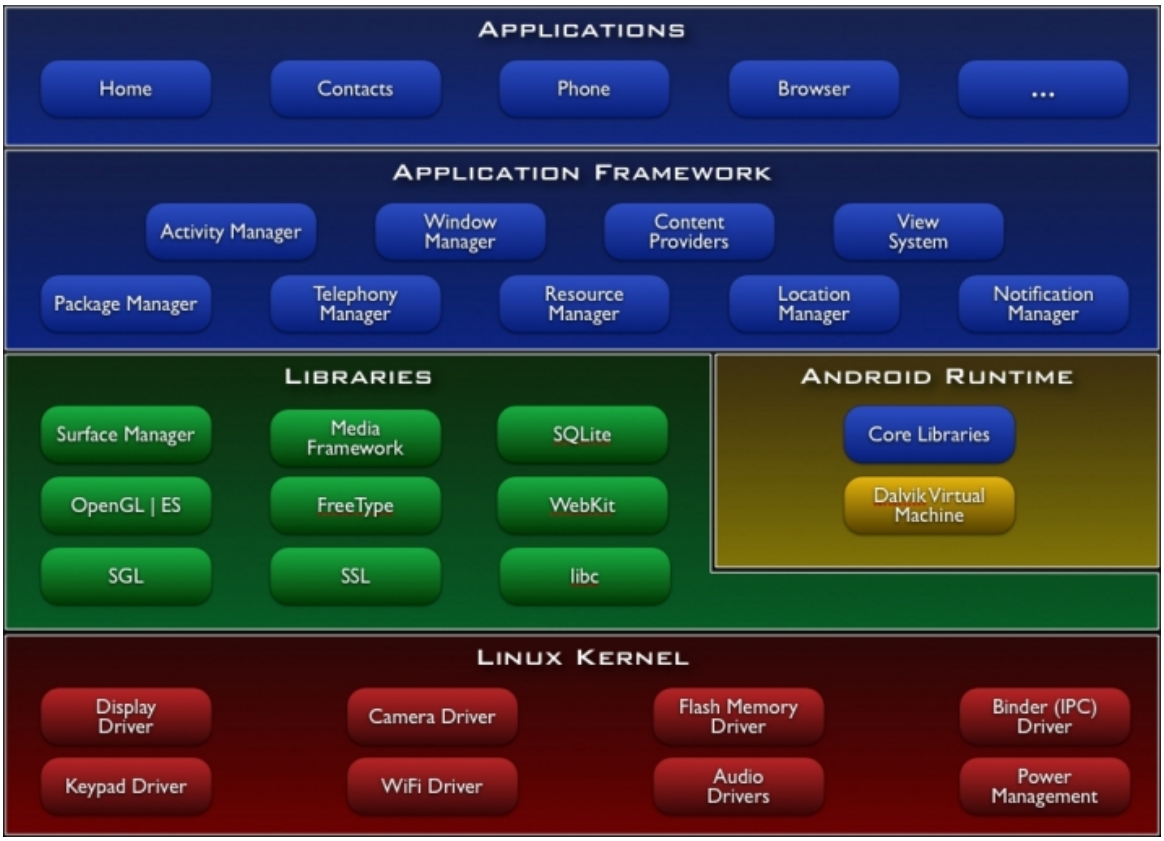
- Dátové úložisko - uchovávanie dát v databáze, resp. na SD karte a jednoduchý prístup k nim.
- Prispôsobenie sa a využívanie dostupného hardvéru - fotoaparát, akcelerometer ...
- Používateľské rozhranie - ovládanie zariadenia je realizované pomocou dotykového displeja, ktorý prenáša používateľove akcie na reálne akcie ako je posúvanie, klikanie atď. Ďalšie ovládacie prvky sú zakomponované do ostatných hardvérových prvkov zariadenia - akcelerometra či gyroskopu. Tieto časti mali pôvodne slúžiť len na ovládanie hier a multimediálnych funkcií, postupom času sa na ne bude pravdepodobne presúvať ovládanie celého zariadenia spolu s hlasovým ovládaním [9], [10], [71].
- Linux - jadro systému je postavené na Linuxe vo verzii 2.6, od verzie 4.0 sa prešlo na Linux verzie 3.x a vyššie. Architektúra jadra bola pozmenená kvôli prispôsobeniu sa menším hardvérovým nárokom, preto napríklad nepodporuje všetky grafické knižnice alebo je problematické portovanie natívnych linuxových aplikácií na Android [11].
- Správa pamäte - keďže mobilné zariadenia sú napájané z batérií, je potrebné obmedziť spotrebu zariadenia na minimum. Nepoužívané aplikácie sú preto systémom pozastavené do operačnej pamäte (RAM). Zvyšuje sa tak reakčný čas zariadenia, lebo aplikácie netreba znova spúšťať a šetrí sa tak aj batéria a systémové zdroje. Správa pamäti je automatická, pokiaľ sa pamäť zaplní, operačný systém začne rušiť nepoužívané aplikácie v závislosti od doby nečinnosti (najdlhšie nečinné sú zrušené ako prvé) [13].

1.3 Architektúra OS Android

OS Android je rozdelený do 5 funkčných vrstiev, ako je vidieť na obrázku 1.1. Každá vrstva má svoj účel a nemusí byť priamo oddelená od ostatných vrstiev:

1.3.1 Jadro OS - Linux Kernel

Základom OS Android je Linuxové jadro vo verzii 2.6, na ktorej boli vykonané rôzne úpravy z dôvodu optimalizácie celkovej architektúry na zariadenia s malým výpočtovým výkonom. Jadro slúži na komunikáciu medzi hardvérom a softvérom príslušného zariadenia za pomoci ovládačov (drivers). Tieto navyše slúžia na správu procesov, pamäti, napájania, atď. Architektúra jadra umožňuje spúšťaniu viacerých aplikácií súčasne, kedy každá aplikácia je považovaná za samostatného používateľa. Podobne ako v klasických OS, každá spustená aplikácia dostane pridelené svoje ID a systém na základe tohto ID prideľuje aplikácii systémové zdroje a nastavuje práva pre všetky súbory. Týmto je zaručená nezávislosť aplikácií, stabilita a čiastočne aj bezpečnosť systému [22], [23].



Obr. 1.1: Rozdelenie funkčných vrstiev OS Android [17]

1.3.2 Knižnice - Libraries

Ďalšou vrstvou systému sú systémové knižnice napísané v C/C++. Zabezpečujú prístup aplikácií k systémovým komponentom a prácu s rôznymi typmi dát. Patria sem napríklad:

- android.media - knižnica na prácu s médiami. Podporuje prehrávanie rôznych audio a video formátov a tiež na prácu s obrazovými dátami. Okrem iného podporuje aj mediálne služby ako napr. streamovanie hudby.
- android.telephony - slúži na sprostredkovanie základnej komunikácie medzi zariadeniami, poskytuje informácie o stave telefónu, SMS správach atď.
- android.location - poskytuje prístup k GPS modulu ak je v príslušnom zariadení a tým aj ku geografickej polohe zariadenia.
- android.hardware - poskytuje prístup k hardvérovému vybaveniu zariadenia - gyroskop, akcelerometer, fotoaparát ...
- android.webkit - knižnica na prácu s webovým obsahom, s plnou podporou CSS, JavaScript, AJAX ...

- OpenGL - grafická knižnica používaná hlavne pri mobilných hrách na vykresľovanie 3D grafiky.
- OpenSSL - knižnica na prácu so štandardom Secure Sockets Layer na zabezpečenie internetovej komunikácie.
- SQLite - knižnica implementujúca do určitej miery samostatnú SQL databázu. Pôvodná SQL databáza bola kvôli použitiu na mobilných zariadeniach do značnej miery odľahčená.

1.3.3 Android runtime

Vrstva Android runtime obsahuje Dalvik Virtual Machine. DVM je aplikačný virtuálny stroj s registrovo orientovanou architektúrou. Bol navrhnutý pre nízky výpočtový výkon a nízke pamäťové nároky. Využíva základné vlastnosti linuxového jadra ako napr. koordinácia procesov, správa pamäti, práca s vláknami atď. Každá aplikácia spustená pod OS Android má okrem vlastného procesu aj vlastnú inštanciu DVM. Okrem DVM táto vrstva obsahuje aj základné knižnice programovacieho jazyka Java. Svojim obsahom sa najviac približujú platforme Java SE [22], [23].

1.3.4 Aplikačný framework

Aplikačný framework je z hľadiska vývoja aplikácií najdôležitejšou vrstvou. Poskytuje totiž prístup k triedam a rozhraniam potrebným na vývoj aplikácií - prístup k hardvéru zariadenia, grafickému rozhraniu atď:

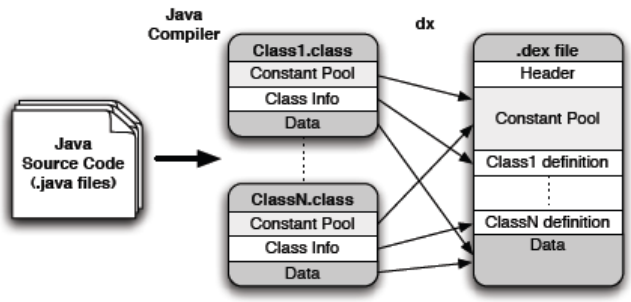
- Activity manager - komponenty na riadenie životného cyklu aplikácií.
- View manager - prvky používané na vytvorenie používateľského rozhrania.
- Telephony manager - prvky potrebné na prístup k telefónnym službám zariadenia.
- Resource manager - správa zdrojov.
- Content provider - slúži na prístup k centrálnemu dátovému úložisku.
- Notification manager - slúži na vyvolanie a správu upozornení, či už ide o zmenu stavu zariadenia alebo správu z aplikácií bežiacich na pozadí [22], [23].

1.3.5 Aplikácie

Najvyššou funkčnou vrstvou OS Android sú aplikácie. Štandardný zoznam aplikácií je predinštalovaný spolu s operačným systémom - e-mailový klient, SMS aplikácia, kalendár, mapy, internetový prehliadač, manažment kontaktov a ďalšie v závislosti od verzie systému a hardvérových možností zariadenia. Ďalšie aplikácie zvyknú do zariadení predinštalovávať aj ich výrobcovia. Okrem toho je možné aplikácie získavať z iných zdrojov:

- Google ako majiteľ práv na OS Android poskytuje niekoľko cloudových služieb využiteľných kompatibilnými zariadeniami:
  - Google Play: súbor služieb, ktorý používateľom umožňuje hľadanie, inštaláciu a nákup aplikácií z ich zariadení alebo Internetu. Uľahčuje vývojárom zverejňovanie svojich aplikácií a tým aj nárast počtu potenciálnych zákazníkov. Poskytuje tiež možnosti recenzovania aplikácií, bezpečnostné skenovanie aplikácií a iné bezpečnostné služby.
  - Android Updates: zabezpečuje aktualizáciu funkčných aj bezpečnostných mechanizmov pre zariadenia bežiace pod OS Android.
  - Android Services: Služby povoľujúce aplikáciám využívanie cloudových služieb - zálohovanie dát a nastavení alebo tzv. cloud-to-device posielanie správ.
- Z neoficiálnych (third party) dátových úložísk - tieto portály poskytujú len možnosť sťahovania aplikácií, akékoľvek aktualizácie musia byť vykonané používateľom manuálne [22], [23].

Aplikácie pre OS Android sú písané v jazyku Java, sú zložené z jednej alebo viacerých tried - *.class*. Java Virtual Machine (JVM) načítava bytecode pre konkrétnu triedu z príslušného *.class* súboru v závislosti na referencovaní pri behu programu. Aplikácia pre DVM naopak pozostáva len z jedného *.dex* súboru, v ktorom sú obsiahnuté všetky potrebné triedy. Po vytvorení JVM bytecode, dalvikovský *dx* kompilátor rozloží všetky *.class* súbory, rekompiluje ich do dalvik bytecode a zapíše do jediného *.dex* súboru, viď obrázok 1.2. V tomto procese dochádza k prekladu, rekonštrukcii a interpretácii základných prvkov aplikácie: konštánt, tried a dát.

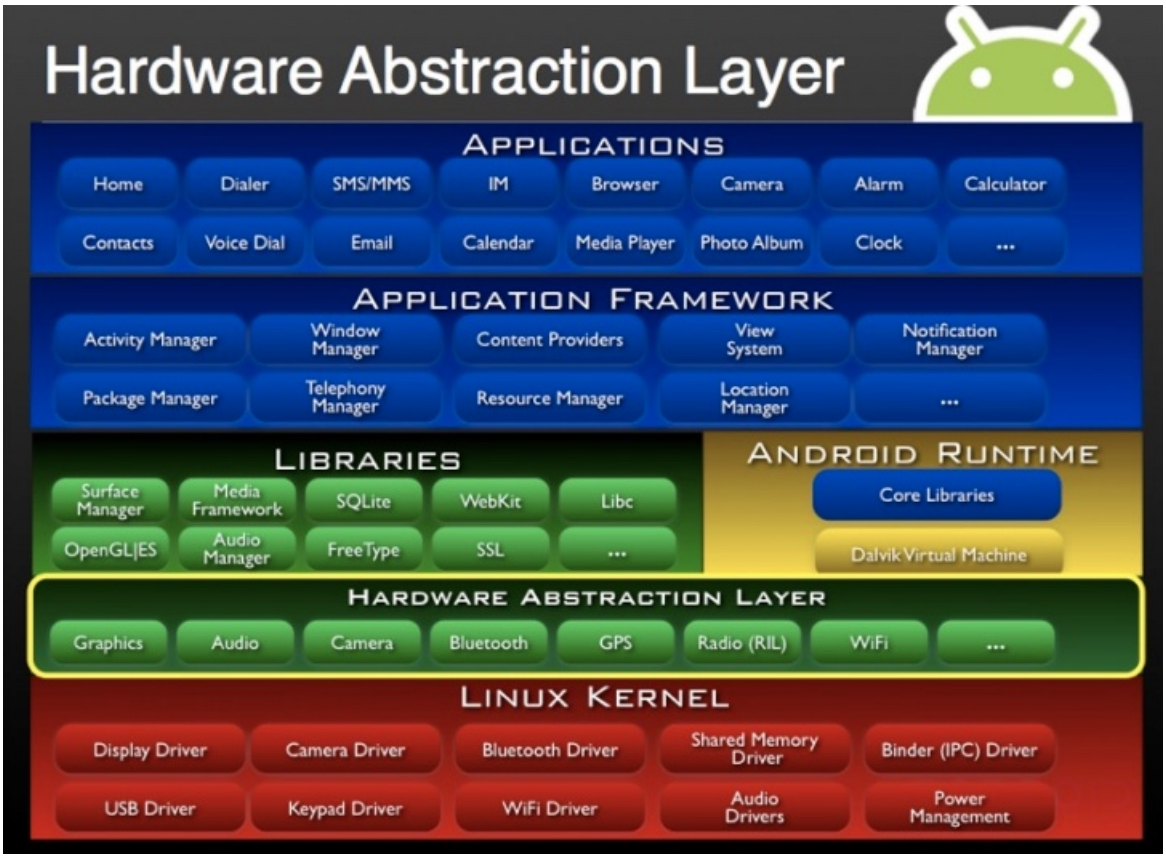


Obr. 1.2: Postup pri kompilácii Android aplikácie [20]

1.3.6 HAL

Hardware Abstraction Layer, skrátene HAL, je novou vrstvou OS Android pridanou vo verzii 6.0, viď obrázok 1.3. HAL je používaný rôznymi bezpečnostnými mechanizmami

(napr. API na prácu s odtlačkami prstov či šifrovanie súborového systému) na ochranu kľúčov pred kompromitovaním jadra resp. lokálnymi útokmi (pri odcudzení zariadenia). Vo svojej podstate predstavuje prepojenie medzi aplikačným frameworkom a softvérom spravujúcim špecializovaný hardvér, napr. snímač odtlačkov prstov.

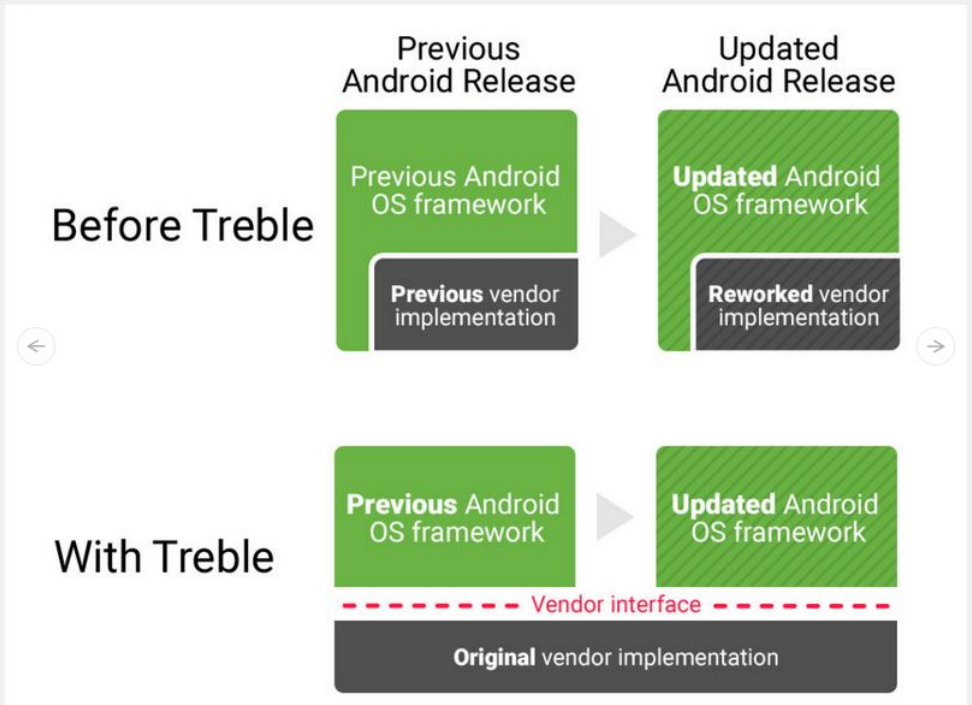


Obr. 1.3: Upravená architekúra OS Android [18]

1.3.7 Project Treble

V snahe o urýchlenie vydávania aktualizácií pre čo najviac zariadení prišla vo verzii 8.0 zmena v modularizácii OS Android [37] nazvaná Project Treble. Zmena spočíva v oddelení "čistého" operačného systému od ovládačov a iného hardvérovo špecifického kódu. Táto zmena výrazne urýchľuje úpravy aktuálnej verzie OS Android pre špecifické potreby výrobcov zariadení resp. mobilných operátorov. Vytvorený bol tzv. "vendor interface" podobný aplikačnému frameworku používaného vývojármi aplikácií, viď obr. 1.4. S touto modularizáciou úzko súvisí aj rozdelenie HAL-u na viacero podsystémov ako napr. fotoaparát, lokalizačné služby, Wi-Fi a pod. Každý z týchto podsystémov sa bude vyvíjať samostatne podľa potreby, resp. podľa uvádzania nových technológií na trh. Od verzií týchto podsystémov bude však závisieť, či bude možné v budúcnosti

aktualizovať OS na novú verziu. Touto úpravou sa docieli aj oddelenie systémových procesov od procesov výrobcu zariadenia resp. mobilného operátora, viď obr. 1.5. Dôsledkom toho je zvýšená bezpečnosť pred zraniteľnosťami ako napr. Stagefright [19]. Okrem štandarných verzií OS Android budú túto modifikáciu využívať aj všetky upravené verzie OS Android, tzv. Custom ROMs ako napr. LineageOS [39], viď kapitola 1.10.1.

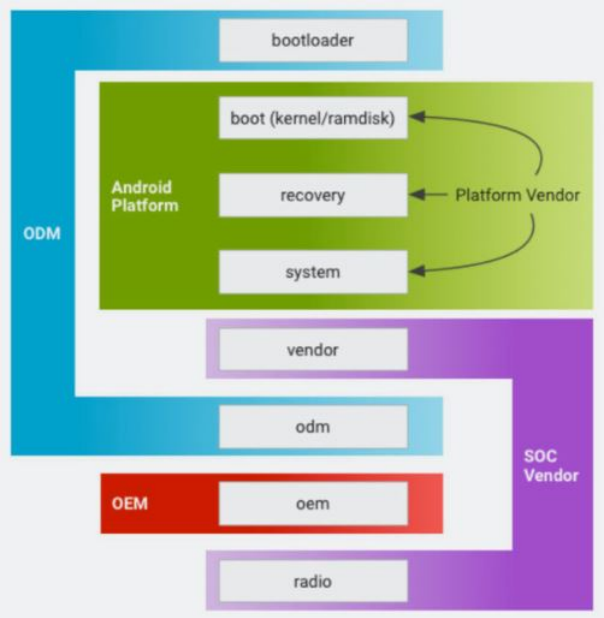


Obr. 1.4: Project Treble - nový model aktualizácie OS [37]

1.4 Bezpečnostné mechanizmy v OS Android

Operačný systém Android bol od začiatku vyvíjaný ako otvorená mobilná platforma. Umožňuje aplikáciám využívať zabudovaný hardvér a softvér spoločne s lokálnymi aj vzdialenými dátami tak, aby používateľom poskytli žiadaný komfort a funkcionalitu. K tomuto všetkému musí navyše platforma poskytovať prostredie a mechanizmy na ochranu používateľských dát, aplikácií a celého zariadenia ako takého.

Zabezpečenie otvorenej platformy vyžaduje robustnú bezpečnostnú architektúru a dôkladne navrhnutý bezpečnostný systém. Tým, že je Android navrhnutý ako viacvrstvový OS, poskytuje jednak potrebnú flexibilitu pri vývoji a na druhej strane poskytuje určitú úroveň ochrany. Bezpečnostné opatrenia boli navrhované so zreteľom na vývojársku komunitu - znalci vedia s mechanizmami jednoducho pracovať, začiatočníci sú chránení bezpečnými štandardnými nastaveniami. Používatelia majú možnosť



Obr. 1.5: Project Treble - oddelenie softvérov vývojárov OS a výrobcov zariadení [37]

sledovať ako jednotlivé aplikácie fungujú a majú nad nimi kontrolu. Tento prístup však umožňuje vykonávať jednoduché útoky ako napr. sociálne inžinierstvo a následnú inštaláciu malvéru. Architektúra Androidu redukuje možnosti takýchto útokov a limituje ich škodlivosť v prípade úspešného vykonania. Základné bezpečnostné mechanizmy môžeme rozdeliť podľa úrovne, na ktorej sú v návrhu architektúry OS Android (ilustrované na obr. 1.6) aplikované:

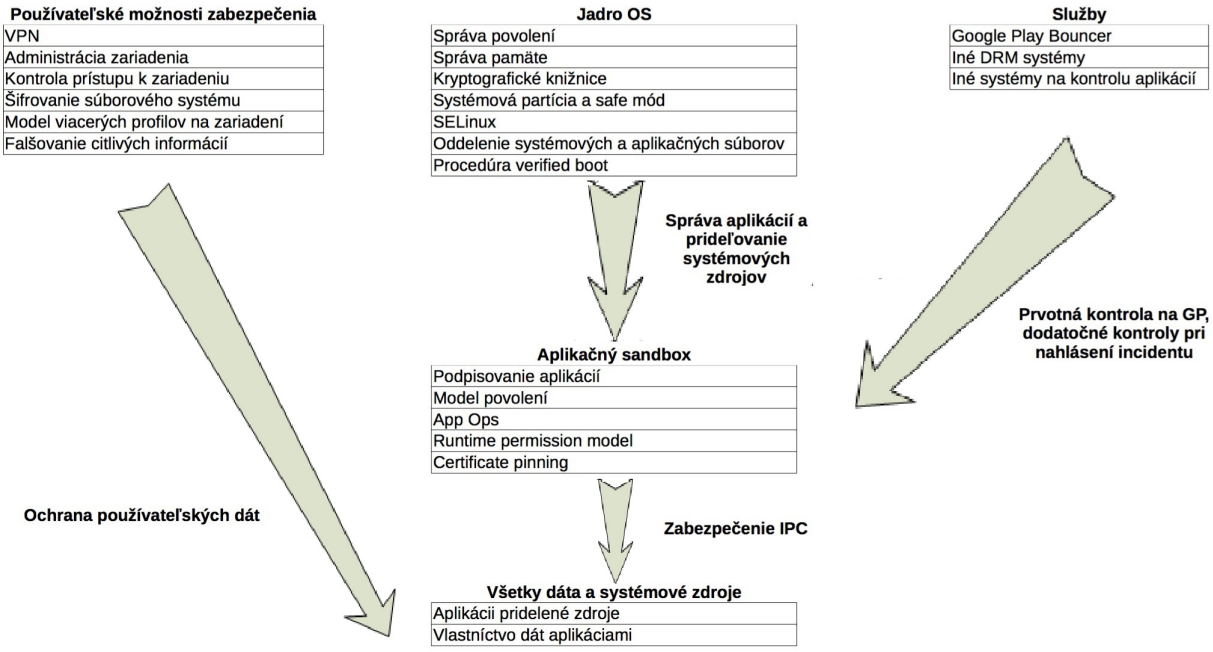
- Jadro a knižnice.
- Android runtime, aplikačný framework a samotné aplikácie.
- Zabezpečenie na úrovni používateľa.
- Dodatočné služby ako Google Play a DRM ochrana.

Tieto mechanizmy sú vo svojej podstate kombináciou bezpečnostných mechanizmov fungujúcich na iných OS (hlavne Linux), prebratých a upravených tak, aby vyhovovali potrebám mobilného OS a zariadení s limitovanými hardvérovými možnosťami. Patria sem napríklad [24]:

- Povinný sandboxing všetkých aplikácií.
- Bezpečná medziprocesová komunikácia.
- Podpisovanie aplikácií.
- Povolenia definované a používateľom schválené pre každú aplikáciu.

Aj týmito prostriedkami sa autori OS snažia dosiahnuť:

- Ochranu používateľských dát pred krádežou alebo iným únikom.
- Ochranu systémových zdrojov pred vyčerpaním, resp. znemožnením používania.



Obr. 1.6: Bezpečnostný model OS Android

1.5 Bezpečnosť na úrovni jadra OS

Na úrovni operačného systému poskytuje platforma Android bezpečnosť Linuxového jadra, ako aj bezpečnú medziprocesovú komunikáciu (IPC) medzi aplikáciami bežiacimi v rôznych procesoch. Tieto bezpečnostné prvky na úrovni OS zabezpečujú, že aj natívne aplikácie podliehajú aplikačnému sandboxingu. Systém tak zabraňuje škodlivým aplikáciam v poškodzovaní iných aplikácií, zariadenia či samotného OS.

1.5.1 Bezpečnosť Linuxu

Ako bolo povedané v úvode, základom OS Android je Linuxové jadro. Samo o sebe je toto jadro rozšírené a používané už dlho rokov aj v citlivých prostrediach. Je neustále skúmané, napádané a optimalizované komunitou odborníkov. Androidu poskytuje niekoľko kľúčových bezpečnostných mechanizmov:

- Model povolení závislých na používateľovi - každá aplikácia obsahuje zoznam povolení, ktoré potrebuje na svoje fungovanie a používateľ ich môže prijať alebo nie.
- Izoláciu procesov - oddelenie jednotlivých aplikácií, resp. im pridelených zdrojov a dát.
- Mechanizmy pre bezpečnú IPC - mechanizmy slúžiace na bezpečnú komunikáciu medzi aplikáciami resp. ich procesmi.
- Možnosť odstránenia nepotrebných a potenciálne nebezpečných častí jadra - jadro Linuxu je modulárne, okrem základných častí sú všetky ostatné oddeliteľné a v nich obsiahnuté komponenty sa nebudú využívať.

Keďže je OS Linux viacpoužívateľský OS, jeho základným cieľom je vzájomné oddelenie systémových zdrojov. Preto Linux [46]:

- Zabraňuje používateľovi A bez oprávnenia (povolenia) čítať a manipulovať s dátami používateľa B.
- Zabezpečuje aby A nevyčerpal pamäť B.
- Zabezpečuje aby A nevyčerpal procesorový čas B.
- Zabezpečuje aby A nevyčerpal systémové zdroje B (GPS, Bluetooth).

Navyše Android využíva linuxový prvok Completely Fair Scheduler (CFS), ktorý zabezpečuje, že medzi procesy sa čas CPU delí rovnomerne. Špecifické procesy môžu dostať väčšiu časovú porciu, ale tento systém im stále zabraňuje v monopolizovaní CPU [53].

1.5.2 Aplikačný sandbox

Android každej aplikácii prideľuje unikátny identifikátor používateľa (UID) a každá aplikácia sa spúšťa ako samostatný používateľ v separátnom procese. Každý používateľ má pridelené určité systémové zdroje a miesto v súborovom systéme. Takto sa pomerne jednoducho dá zabezpečiť oddelenie systémových zdrojov a súborov jednotlivých používateľov, v tomto prípade jednotlivých aplikácií. Tento postup je odlišný ako u iných OS, kde viaceré aplikácie môžu bežať s rovnakými používateľskými právami.

Toto je základom aplikačného sandboxu na úrovni jadra. Jadro zabezpečuje bezpečnosť medzi aplikáciami a systémom na procesovej úrovni pomocou štandardných linuxových možností, napr. prideľovaním UID a G(roup)ID aplikáciám. Aplikácie v štandardnom nastavení nemôžu medzi sebou komunikovať a majú obmedzený prístup k OS. Keďže má každá aplikácia oddelené dáta a zdroje od ostatných aplikácií, akýkoľvek útok je tak redukovaný len na napadnutú aplikáciu [53].

Keďže je aplikačný sandbox v jadre, bezpečnostný model ho rozširuje do natívneho kódu a systémových aplikácií. V podstate všetok softvér, vrátane knižníc OS, aplikačného frameworku a runtime prostredia beží v sandboxe. Na platforme neexistujú žiadne obmedzenia na dosiahnutie bezpečnosti v podobe špeciálneho frameworku, API alebo programovacieho jazyka, natívny kód je rovnako bezpečný ako interpretovaný kód. Nemôže sa tu vyskytnúť prípad poškodenia celej operačnej pamäte, keďže daná aplikácia má prístup len k jej vyhradenej časti. Hoci nie je aplikačný sandbox dokonalý, na jeho prelomenie na riadne nakonfigurovanom zariadení by však bolo potrebné prelomiť zabezpečenie linuxového jadra.

1.5.3 Systémová partícia a Safe mód

Systémová partícia obsahuje jadro OS, jeho knižnice, aplikačný runtime a aplikácie. Je nastavená len na čítanie (read-only). Keď používateľ nabootuje zariadenie do bezpečného (safe) módu, dostupné má len základné androidové aplikácie. Toto zaručuje, že sa používateľ dostane do prostredia, kde nie sú aplikácie tretích strán.

Pri normálnom spustení je obraz systému nastavený ako read-only. Všetky dôležité vykonávateľné a konfiguračné súbory sú uložené v ramdisku alebo na obraze systému. Takto sa zabezpečí, že aj keď útočník získa možnosť zapisovať do súborov na disku, stále mu nie je dovolené meniť alebo nahradiť kritické súbory [47].

1.5.4 Súborový systém a povolenia

Všetky súbory v OS Android podliehajú linuxovému mechanizmu povolení - Read Write a eXecute (rwx), ktoré sú prideľované na základe UID resp. GID. Systémové súbory patria používateľovi "system"/"root" a aplikačné súbory patria príslušným aplikáciám [47].

V UNIX-ovom prostredí, povolenia súborového systému zabezpečujú, že žiadny používateľ nemôže bez povolenia čítať ani meniť súbory iného používateľa. V našom prípade každá aplikácia reprezentuje jedného používateľa. Pokiaľ vývojár explicitne nesprístupní súbory svojej aplikácie iným aplikáciám, súbory nemôžu byť prístupné iným aplikáciám.

1.5.5 Kryptografické knižnice

OS Android poskytuje na používanie pre aplikácie niekoľko kryptografických API. Tieto obsahujú implementácie štandardných a bežne používaných kryptografických primitív ako AES, RSA, DES a SHA. Navyše tieto API poskytujú aj protokoly vyšších vrstiev, ako napr. SSL a HTTPS. Vo verzii Android 4.0 bol predstavený *KeyChain* - trieda umožňujúca aplikáciám ukladať svoje privátne kľúče a certifikáty do systémového úložiska citlivých údajov. Vo verzii 5.0 bolo umožnené používanie TLSv1.2 a TLSv1.1. Od tejto verzie sa takisto preferuje používanie tzv. "forward secrecy", na

šifrovanie sa odporúča AES-GCM a slabé kryptografické primitívy ako napr. MD5 a 3DES už nie sú podporované [48].

1.5.6 Verified boot

Procedúra verified boot bola pridaná vo verzii 6.0 na overenie integrity jednotlivých komponentov OS pri bootovacom procese. Jeho podstata je v množine kryptografických kontrol jednotlivých komponentov pred ich vykonaním (v našom prípade pred bootovaním konkrétneho komponentu). Týmto spôsobom sa zaručuje integrita a bezpečnosť OS od bootloadera až po najvyššiu vrstvu OS [48]. Od verzie 8.0 táto procedúra obsahuje tzv. "rollback ochranu" [37]. Takéto zariadenie nebude možné spustiť so staršími verziami OS Android, v ktorých bolo možné obchádzať rôzne bezpečnostné mechanizmy a dostať sa tak do zariadenia. Nepriamym dôsledkom pridania tejto ochrany je aj sťaženie odomknutia bootloaderu bez povolenia používateľa.

1.5.7 Seamless updates

Procedúra "aktualizácie bez čakania" bola pridaná vo verzii 7.0. Jej hlavnou úlohou a je zabezpečenie nainštalovania bezpečnostných aktualizácií na zariadenie v čo najkratšom čase. Využíva na to dve systémové partície - na jednej sa dá normálne pracovať a druhá čaká na prípadnú aktualizáciu. Ak sa nejaká vyskytne, nainštaluje sa na druhú partíciu (vrátane aktualizácie všetkých aplikácií), zariadenie sa reštartuje a pokračuje v chode na tejto partícii. Prvá partícia teraz čaká na ďalšiu aktualizáciu a celý proces sa takto opakuje. Tento spôsob má niekoľko výhod - je rýchly, takže čakanie pri inštalácii sa zredukovalo na reštart zariadenia a neodrádza používateľa od tohto úkonu. Po druhé sa výrazne zvyšuje bezpečnosť zariadenia, pretože do aktualizovanej partície majú prístup iba systémové aplikácie a procesy, ktoré vykonávajú aktualizáciu a nič iné [49].

1.5.8 Key attestation

Key attestation umožňuje vývojárom aplikácií uchovávať kľúče používané v ich aplikáciách v bezpečnom hardvérovom úložisku kľúčov na zariadení. Pokiaľ zariadenie takýto komponent nemá použije sa softvérové systémové úložisko. Mechanizmus overovania kľúčov je na báze certifikátov a je spojený s procedúrou Verified boot [49].

1.5.9 Správa pamäte a vylepšenia jej bezpečnosti

Požiadavkou mnohých moderných OS a hlavne Linuxu je tzv. Memory Management Unit (MMU) - hardvérový komponent zabezpečujúci separáciu procesov do rôznych adresných priestorov (virtuálna pamäť). MMU zabezpečuje, že jeden proces nemôže čítať pamäťové stránky iného procesu alebo poškodiť jeho pamäť. Redukuje sa tým možnosť privilege escalation útokov, lebo proces nemôže spustiť svoj kód v privilegovanom móde prepísaním privátnej pamäte OS [47].

Android obsahuje veľa prvkov, ktoré sťažujú zneužívanie častých bezpečnostných chýb. Postupom času sa tieto prvky podarilo úspešne implementovať a jednoducho distribuovať cez aktualizácie systému, napr.:

- Rôzne ochrany pred pretečením zásobníka.
- Mechanizmy na správnu alokáciu pamäte.
- Ochrany zraniteľnosti reťazcových (string) formátov.
- Address Space Layout Randomization (ASLR) - znáhodnenie ukladania kľúčov v pamäti.
- Ochrana pri dereferencovaní null pointera.
- Podpora Position Independent Executable - kódu vykonávateľného bez ohľadu na absolútnu adresu v pamäti.

1.5.10 Rootovanie zariadenia

V štandardnom nastavení beží s právami roota (administrátora) len jadro a malý počet kľúčových aplikácií. Android nezabraňuje aplikáciám ani používateľom s root právami v zmene OS, jadra a iných aplikácií. Vo všeobecnosti, root má neobmedzený prístup ku všetkým aplikáciám a dátam. Používatelia prideľovaním root práv aplikáciám riskujú vystavenie sa rôznym bezpečnostným hrozbám.

Možnosť modifikácie svojho zariadenia je dôležitá pre vývojárov. Na mnohých zariadeniach majú používatelia možnosť odomknúť bootloader a nainštalovať iný operačný systém. Tieto alternatívne OS dávajú používateľom root práva na debugovanie aplikácií a systémových súčastí, resp. na získanie prístupu k prvkom nedostupným aplikáciám androidovými API. Na niektorých zariadeniach sa môže nový OS inštalovať jednoducho cez USB kábel a tak získať potrebné práva. Na ochranu používateľských dát odomykací mechanizmus bootloaderu vyžaduje zmazanie týchto dát. Táto ochrana sa dá obísť využitím chyby v jadre alebo inej bezpečnostnej chyby.

Šifrovanie dát kľúčom uloženým na zariadení nechráni aplikačné dáta pred používateľmi s root právami. Aplikácie sa proti tomuto môžu brániť použitím hesla uloženého mimo zariadenia - na vzdialenom serveri, resp. vypýtaním si hesla od používateľa. Toto je však len dočasná ochrana - pokiaľ je heslo poskytnuté aplikácii, tak je okamžite prístupná root používateľom. Robustnejší prístup ako chrániť dáta sa dá uskutočniť pomocou rôznych hardvérových metód - na limitáciu prístupu k špecifickému obsahu (DRM pre prehrávanie videa, NFC bezpečné úložisko pre Google wallet). V prípade stratenia alebo odcudzenia zariadenia môže kompletné šifrovanie súborového systému využívajúce nejaký hlavný šifrovací kľúč zamedziť prístupu k dátam aj v prípade modifikácie bootloaderu [24].

1.5.11 SELinux

Vzhľadom na výskyt rôznych bezpečnostných chýb v systéme a potenciálnych hrozieb, sa výskumní pracovníci začali orientovať na možnosti zlepšenia zabezpečenia OS Android. Vznikali tak rôzne snahy o tzv. platform hardening. V podstate ide o vylepšenia operačného systému o rôzne nové bezpečnostné mechanizmy, ktoré nie sú zakomponované v aktuálnej verzii OS. Mnoho týchto vylepšení si nakoniec našlo cestu aj do aktualizácií operačného systému. Vzhľadom na obmedzené hardvérové možnosti zariadení sa vývojári zamerali na SELinux [72].

SELinux nie je linuxová distribúcia, ale súbor úprav a modifikácií samotného jadra systému vrátane doplnenia o používateľské nástroje. SELinux implementuje MAC (Mandatory Access Control) ako doplnok k systému prístupu typu DAC (Discretionary Access Control), ktorý používajú unixové systémy. SELinux spĺňa kritériá dôveryhodnej platformy podle TCSEC (Trusted Computer System Evaluation Criteria). SELinux vynucuje administrátorom definovanú bezpečnostnú politiku nad všetkými subjektami a objektami v OS. Umožňuje tak nastaviť jemnejšie prístupové práva (vyššiu granularitu) k dátam a zamedziť ich zmenám. SELinux navyše poskytuje možnosť izolácie domén na úrovni jadra systému. Takouto izoláciou vieme predísť viacerým hrozbám zo strany aplikácií alebo služieb. Veľa vírusov sa šíri pomocou SMS/MMS správ, izoláciou týchto služieb by sme zabránili, resp. výrazne spomalili ďalšie šírenie týchto škodlivých aplikácií. Je však potrebné zabezpečiť izoláciu bez prílišného obmedzenia interakcie medzi procesmi, prístupom k súborom a systémovým zdrojom. Toto práve zabezpečuje MAC-based Secure Kernel implementovaný pomocou Linux Security Modules (LSM) v jadre Linuxu.

Pokusy o implementáciu SELinuxu na Android začali už v r. 2010 [73]. V tejto práci sa autori priamo zameriavajú na využitie SELinux na platforme Android. Primárnym cieľom bolo nájsť spôsob ako obmedziť právomoci root používateľa a tak limitovať útoky typu privilege escalation. Výsledným riešením bolo implementovanie LSM, kde bola demonštrovaná efektivita SELinuxu v týchto prípadoch.

Integrácia SELinux do OS Android

Od verzie 4.3 je Android sandbox zosilnený SELinuxom použitím jeho systému MAC v linuxovom jadre. Toto zosilnenie je pre vývojárov a používateľov neviditeľné a pridáva robustnosť do bezpečnostného modelu OS Android pri zaručení kompatibility s existujúcimi aplikáciami. Na zaručenie tejto kompatibility bol v tejto verzii SELinux nastavený v tzv. permissive móde, kedy len zaznamenával (logoval) narušenia bezpečnostnej politiky, ale nenarúšal činnosť aplikácií alebo správanie sa systému.

Vo verzii 4.4 bol SELinux prepnutý do tzv. enforced módu. V tomto móde sú všetky binárne súbory z domény *root* "donútené" dodržiavať bezpečnostné politiky. Všetky ostatné časti systému boli ponechané v permissive móde. Posledná zmena nastala vo

verzii 5.0. Od tejto verzie sú všetky časti systému v enforced móde, t.j. dodržiavanie bezpečnostných politík je vynucované naprieč celým systémom [48].

1.6 Bezpečnosť Android aplikácií

1.6.1 Prvky aplikácií

Android poskytuje otvorenú platformu a aplikačné prostredie pre mobilné zariadenia. Aplikácie sú najčastejšie písané v jazyku Java a bežia v DVM, môžu byť však písané aj v natívnom kóde. Inštalované sú z jediného súboru typu *.apk*. Základnými stavebnými prvkami aplikácie sú:

- AndroidManifest.xml: kontrolný súbor, ktorý vraví systému čo robiť s komponentami vyšších vrstiev (aktivity, služby) v danej aplikácii. Navyše sú tu špecifikované potrebné povolenia.
- Aktivity (Activities): vo všeobecnosti ide o jednu úlohu zameranú na používateľa - väčšinou totiž zobrazujú používateľské rozhranie (user interface).
- Služby (Services): kód bežiaci v pozadí. Môže bežať ako samostatný proces alebo v kontexte procesu inej aplikácie. Iné komponenty môžu byť viazané na služby a volať metódy cez volania vzdialených procedúr (hudobný prehrávač).
- Broadcast receiver: objekt, ktorý je volaný keď IPC mechanizmus, tzv. intent je vydaný operačným systémom alebo inou aplikáciou. Aplikácie sa k nim registrujú, napr. pri obdržaní informácie o slabej batérii sa zmení ich správanie.
- Content provider: v podstate ide o databázy, používajú sa buď ako perzistentné interné dátové úložisko alebo ako mechanizmus na zdieľanie informácií medzi aplikáciami [81].

Na komunikáciu medzi komponentami (aktivity, služby, broadcast receivery) Android využíva sofistikovaný systém posielania správ, tzv. intentov (intents). Ide o správy so špecifikovaným adresátom a dátami - špecifikuje vzdialenú procedúru, ktorá má byť zavolaná a obsahuje príslušné argumenty. Využívajú sa na inter- aj intraaplikačnú komunikáciu. OS ich používa ako notifikáciu dôležitých správ (system broadcast intents). Môžu byť využité na explicitnú (špecifikovaný príjemca) aj implicitnú (ľubovoľná aplikácia podporujúca danú operáciu) komunikáciu medzi aplikáciami.

1.6.2 Model povolení - prístup k chráneným API

Ako bolo povedané v predošlej časti, aplikácie bežia v aplikačnom sandboxe a majú prístup k obmedzeným systémovým zdrojom. Systém spravuje prístup aplikácií k zdrojom a pokiaľ je použitý nesprávne alebo zlomyseľne, môže závažne ovplyvniť celkové fungovanie zariadenia alebo ohroziť dáta. Tieto obmedzenia sú implementované rôznymi spôsobmi. Niektoré možnosti sú obmedzené nedostatkom príslušných API pre

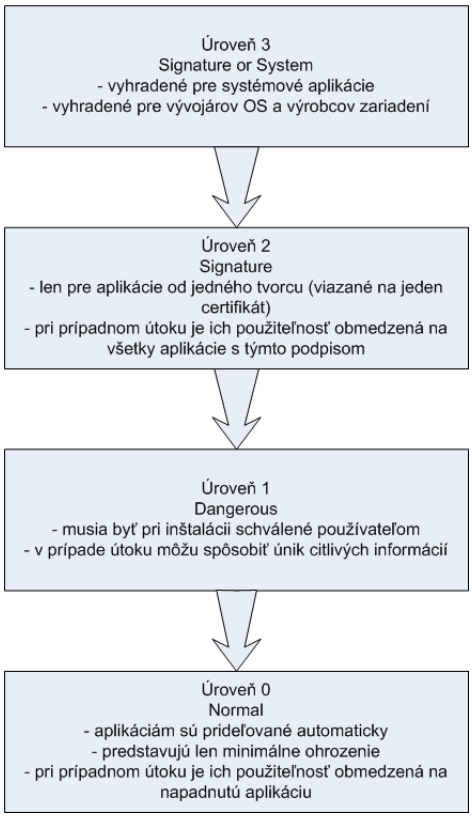
konkrétnu funkcionalitu, ďalšie napr. separáciou rolí. Citlivé API sú používané len dôveryhodnými aplikáciami a chránené systémom povolení. Povolenia sú rozdelené do do štyroch skupín podľa úrovne ochrany (pre ilustráciu viď obr. 1.7):

- Normal - povolenia aplikačnej úrovne, nepredstavujú vážne riziko pokiaľ ich aplikácia využíva.
- Dangerous - nebezpečné povolenia, ktoré môžu spôsobiť únik a manipuláciu s citlivými dátami alebo využívať potenciálne nebezpečné zdroje zariadenia. Musia byť explicitne potvrdené používateľom pri inštalácii aplikácie. Patria sem napr.:
  - lokačné dáta z GPS (ACCESS_FINE_LOCATION)
  - sieťové/dátové spojenie (ACCESS_NETWORK_STATE)
  - funkcie telefónie (CALL_PHONE)
  - funkcie SMS/MMS (WRITE_SMS)
  - prístup k systémovým nastaveniam (CHANGE_CONFIGURATION)
- Signature - povolenia, ktoré je možné pridelit' len aplikáciám podpísaným privátnym kľúčom zodpovedajúcim certifikátu ako má aplikácia, ktorá ich volá. Sú využívané vývojármi na zdieľanie informácií medzi ich aplikáciami.
- Signature-or-system - zvláštny typ povolenia, ktoré je možné pridelit' len aplikáciám inštalovaným v systémovom obraze alebo aplikáciám alebo sú podpísané rovnakým certifikátom ako systémový obraz.

Okrem toho sa povolenia delia na dve skupiny podľa spôsobu ich prijímania:

- Time-of-use - používateľ ho musí potvrdiť pri vykonávaní citlivej operácie (napr. prístup k polohe zariadenia). Je to jediný spôsob ako zabrániť aplikácii v prístupe k zdrojom zariadenia.
- Install-time - prijímajú sa v čase inštalácie aplikácie, používateľ ich prijíma ako celok, nemôže si pri danej aplikácii vyberať, ktoré povolenia prijme a ktoré odmietne, viď obr 1.8.

Tieto zdroje sú prístupné len z OS. Aplikácie musia mať v manifeste povoleniami špecifikované aké má požiadavky na tieto zdroje. Do vydania verzie 6.0 pri inštalácii aplikácie sa tieto povolenia zobrazili a používateľ ich mohol prijať alebo odmietnuť. Po prijatí sa pokračovalo v inštalácii a systém tieto povolenia akceptoval. Nebolo možné vyberať, ktoré povolenia chcel používateľ povoliť, museli byť povolené ako celok, čo mohlo viesť k bezpečnostným incidentom. Povolenia boli aplikácii pridelené počas celej doby čo bola nainštalovaná v zariadení a neboli dodatočné pýtané od používateľa. Odstránené boli v momente odinštalovania aplikácie. Dali sa pozrieť v nastaveniach aplikácií a mohli byť obmedzené vypnutím globálnej funkcionality, napr. vypnutím wi-fi alebo GPS. V prípade, že sa aplikácia pokúšala dostať k prvkom na ktoré nemala oprávnenie, tak

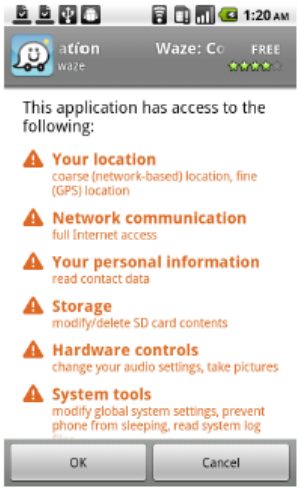


Obr. 1.7: Model povolení v OS Android

vyvolala bezpečnostnú výnimku a chybové hlásenie v aplikácii. Kontroly povolení pre chránené API sú vykonávané na čo najnižšej úrovni, aby sa zabránilo ich obchádzaniu. Niektoré možnosti zariadenia nie sú dostupné pre aplikácie tretích strán, ale môžu byť používané predinštalovanými aplikáciami. Kompletný zoznam povolení je dostupný na stránke venovanej vývoju na OS Android [54], [46], [85].

1.6.3 App Ops

V procese integrácie SELinuxu do jadra Androidu (verzia 4.3) sa medzi systémovými aplikáciami objavila aplikácia zvaná App Ops [25]. Či bola zverejnená náhodne alebo to bolo zámerov vývojárov sa doteraz nevie. Podstatné je, že umožňovala používateľom spravovať jednotlivé povolenia pridelené aplikáciám. Používatelia tak mohli zablokovať prístup k systémovým zdrojom aplikáciám, keď to považovali za bezpečnostnú hrozbu, zbytočnosť, alebo dochádzalo k zbytočnému vybíjaniu batérie, viď obr 1.9. Keďže vývojári aplikácií neboli na túto možnosť pripravení, aplikácie ktorým bolo odobraté nejaké povolenie padali. Hlavne kvôli týmto problémom bola App Ops vo verzii 4.4.2 odstránená, oficiálne z bezpečnostných dôvodov. Napriek tomu ju bolo možné aj v tejto verzii obnoviť, stiahnuť sa dá doteraz na staršie zariadenia napr. tu [26].



Obr. 1.8: Príklad inštalácie aplikácie [103]

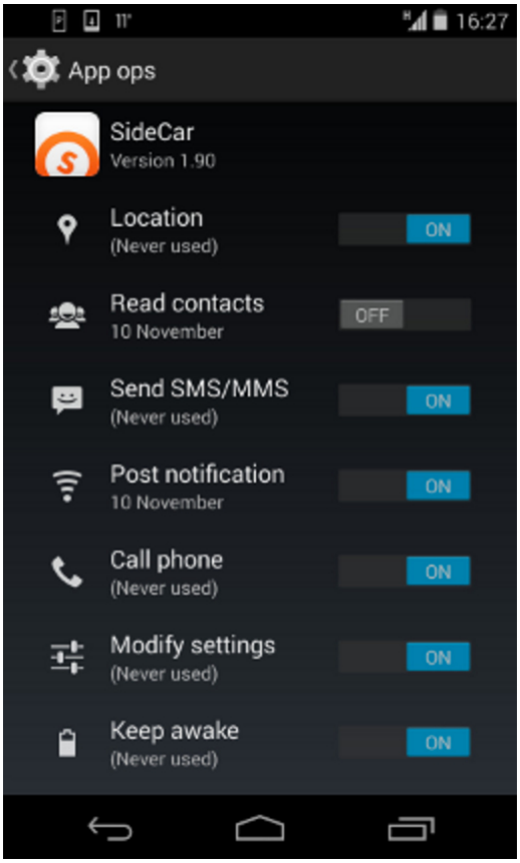
1.6.4 Runtime Permission Model

Odstránenie App Ops bolo napriek problémom a sťažnostiam predzvesťou doteraz asi najväčšej bezpečnostnej aktualizácie OS Android, a tým je uvedenie tzv. "runtime" povolení [48] vo verzii 6.0. Runtime povolenia sú vo svojej podstate time-of-use povolenia spomenuté vyššie. Pokiaľ chce aplikácia pristúpiť ku konkrétnemu systémovému zdroju, musí o to požiadať používateľa, viď obr 1.10. Ten sa môže rozhodnúť prístup povoliť alebo zamietnuť, pokiaľ sa mu zdá, že požiadavka nie je oprávnená. Vzhľadom na fakt, že väčšina zariadení stéle beží na starších verziách OS Android, je tento model plne funkčný len pre aplikácie vyvýjané priamo na verziu 6.0 (API level 23) [29]. Staršie aplikácie sú nateraz z tohto modelu vynechané, fungujú normálne a aj ich inštalácia je rovnaká ako pri predošlom modeli. Pokiaľ sa však používateľ rozhodne blokovať im prístup k nejakým zdrojom, aplikácia spadne, podobne ako pri App Ops. Zmena modelu povolení má zatiaľ pomerne zmiešané ohlasy. Používatelia sú spokojní s možnosťou riadenia prístupu aplikácií k systémovým zdrojom. Na druhej strane, vývojárom táto zmena prinesie veľké množstvo práce navyše, lebo sa výrazne zmenila logika volania metód pracujúcich s týmito zdrojmi [30].

1.6.5 Cost-Sensitive API

Cost-Sensitive API je funkcia, ktorej spustenie môže používateľa alebo sieť niečo stáť. Tieto API sú taktiež v zozname chránených API kontrolovaných OS. Používateľ musí vystaviť explicitné povolenie aplikáciám tretích strán pokiaľ k nim chcú pristúpiť. Patria sem napríklad:

- funkcie telefónie
- SMS/MMS



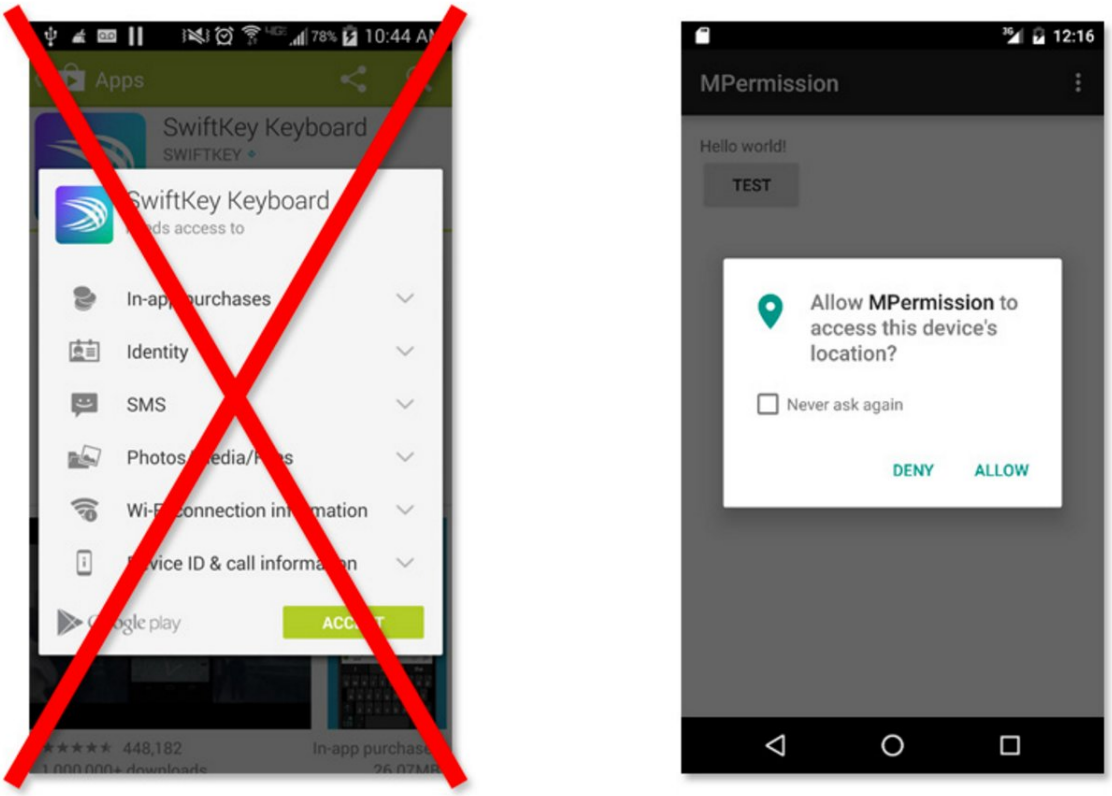
Obr. 1.9: Ukážka aplikácie App Ops [25]

- prístup k sieti a dátam
- spracovanie finančných údajov v aplikácii
- prístup k NFC

Android verzie 4.2 pridáva ďalšiu kontrolu k SMS - notifikácie ak sa nejaká aplikácia pokúsi poslať SMS na tzv. premium čísla (zväčša platené služby). Poslanie SMS je tak v kompetencii používateľa.

1.6.6 Osobné informácie

API poskytujúce prístup k používateľovým dátam patria tiež medzi chránené. Pri normálnom používaní androidové zariadenia zberajú používateľské dáta v inštalovaných aplikáciách tretích strán. Aplikácie, ktoré chcú zdieľať tieto informácie môžu použiť kontrolu povolení v rámci OS na ochranu dát pred týmito aplikáciami, viď obrázok 1.11.



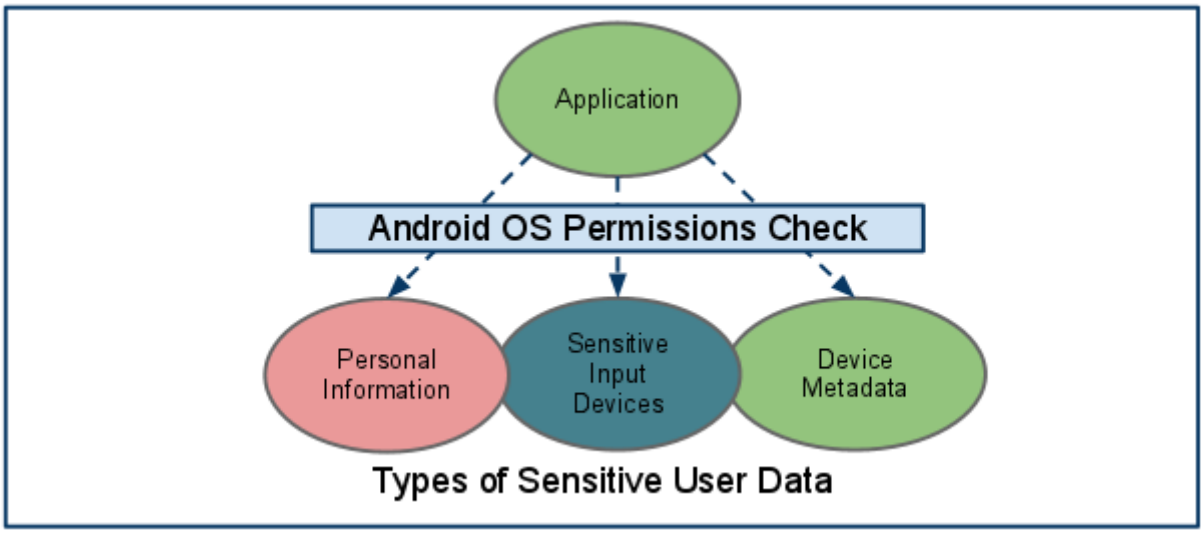
Obr. 1.10: Ukážka runtime permission modelu [30]

Systémové komponenty pracujúce s dátami, tzv. content providers, obsahujúce osobné údaje (kontakty, kalendár) majú špecifické povolenia. Táto granularita používateľovi napovedá, aké informácie môžu byť danej aplikácii poskytnuté. Počas inštalácie môže aplikácia o tieto povolenia požiadať. Pokiaľ budú schválené, bude ich môcť využívať po celý čas čo bude nainštalovaná na zariadení. Každá aplikácia pracujúca s osobnými údajmi by ich mala mať obmedzené pre svoje použitie. Ak by ich chcela zverejniť ostatným aplikáciám cez IPC, tak táto aplikácia môže cez IPC mechanizmus uplatniť povolenia týkajúce sa týchto dát a ktoré sú vyžadované OS.

1.6.7 Medziprocesová komunikácia

Procesy medzi sebou komunikujú štandardnými UNIX-ovými mechanizmami ako sú signály alebo sockety. Aj tu však platia linuxové povolenia, preto Android poskytuje iné možnosti IPC:

- Binder: jednoduchý mechanizmus volania vzdialenej procedúry navrhnutý pre vysoký výkon pri práci s vnútro- a medziprocesových volaniach. Implementovaný je pomocou linuxového ovládača.



Obr. 1.11: Riadenie prístupu k citlivým dátam [24]

- Služby (services): horeuvedené služby poskytujú rozhrania pre priamy prístup keď sa použije binder.
- Intents: jednoduché správy reprezentujúce úmysel niečo vykonať. Keď chceme aby aplikácia vykonala nejakú činnosť, vyšle sa intent pre systém. Ten lokalizuje príslušnú aplikáciu, ktorá vie s intentom pracovať a spustí ju. Používajú sa aj na broadcast dôležitých informácií.
- Content Providers: dátové úložisko poskytujúce prístup k dátam zariadenia, napr. zoznamu kontaktov. Aplikácie môžu pristupovať k dátam, ktoré iné aplikácie zverejnili cez content provider.

V súčasnosti je možné implementovať IPC rôznymi mechanizmami ako sú napr. sockety. Je však odporúčané používať na to vyhradené frameworky, aby sa zamedzilo úniku dát a výskytu bezpečnostných chýb.

1.6.8 Certificate Pinning

Od verzie 4.2 je súčasťou bezpečnostného modelu OS Android tzv. Certificate Pinning ("pripínanie certifikátov") [28]. Jeho podstatou je proces asociácie hosta (v našom prípade aplikácie alebo zariadenia) s očakávaným X509 certifikátom alebo verejným kľúčom. Tento mechanizmus využíva znalosť vopred existujúceho vzťahu medzi používateľom a organizáciou resp. službou na zjednodušenie bezpečnostných rozhodnutí. Vďaka tomu, že aplikácia má zabudovanú informáciu o serveri alebo službe, netreba sa spoliehať na všeobecné mechanizmy riešiace problém distribúcie kľúčov [52]. Pripnuté domény dostanú správu o neúspešnej validácii certifikátu, ak sa tento certifikát

nedá priradiť do množiny očakávaných certifikátov. Tieto správy chránia používateľa pred možnosťou kompromitácie certifikačnej autority. Od verzie 4.4 je zabudovaná detekcia a prevencia podvodných Google certifikátov používaných v bezpečnej SSL/TLS komunikácii [48].

1.6.9 Prístup k SIM karte

Nízkoúrovňový prístup k SIM karte nie je pre aplikácie tretích strán povolený. Všetku komunikáciu so SIM kartou obstaráva OS, vrátane prístupu k citlivým informáciám. Aplikácie navyše nemajú prístup k AT príkazom, lebo tie sú riadené výhradne vrstvou Radio Interface Layer (RIL).

1.6.10 Zadávanie citlivých dát

Androidové zariadenia často poskytujú citlivé údaje na interakciu aplikácií s okolím - fotoaparát, GPS. Pre aplikácie tretích strán musí byť tento prístup zabezpečený príslušným povolením. Pokiaľ je napr. schválené povolenie na zisťovanie polohy zariadenia a používateľ ho už nechce využívať, tak nemusí aplikáciu odinštalovať, ale len vypnúť GPS a bezdrôtové siete v nastaveniach zariadenia. So zadávaním citlivých informácií úzko súvisí aj možnosť používania tzv. StrictMode od verzie 6.0. Používanie, resp. vynútenie tohto módu vývojármi zaručuje, že aplikácia pri komunikácii a práci s používateľskými dátami ich nebude používať vo forme otvoreného textu [48].

1.6.11 Metadáta zariadenia

Android sa snaží obmedziť prístup k dátam, ktoré nie sú citlivé, ale môžu nepriamo odhaliť charakteristiky používateľa, nastavenia zariadenia a spôsob jeho používania. Štandardne nemajú aplikácie prístup k logom OS, histórii internetového prehliadača alebo telefónnemu číslu. Aplikácia si ich znova môže vyžiadať pri inštalácii a je na používateľovi či ich prijme alebo nie.

1.6.12 Podpisovanie aplikácií

Podpisovanie zdrojového kódu slúži na identifikáciu autora aplikácie a na aktualizovanie aplikácie bez potreby vytvárať zložité rozhrania a povolenia. Každá aplikácia musí byť podpísaná výrobcom, nepodpísané aplikácie nemožno nainštalovať do zariadenia. Vytvára sa tak určitá dôvera medzi distribútorom a tvorcom aplikácie - výrobca vie, že jeho aplikácia nebude ďalej modifikovaná a distribútor vie zobrať autora na zodpovednosť za správanie sa aplikácie.

Podpísanie aplikácie je prvým krokom jej vloženia do sandboxu - podpísaný certifikát aplikácie definuje, ktoré UID bolo aplikácii pridelené. Zabezpečuje sa tak, že aplikácia nemá prístup k iným aplikáciám pokiaľ nemá definované príslušné IPC. Pri inštalácii sa overuje podpis v .apk súbore. Aplikácie môžu byť podpísané treťou stranou alebo

samé sebou. Aplikácie nemusia byť podpisované centrálnou autoritou a Android nevykonáva CA verifikáciu pre tieto certifikáty.

Aplikácie môžu navyše deklarovať bezpečnostné povolenia na úrovni podpisovej ochrany, obmedzujúc tak prístup len na aplikácie podpísané rovnakým kľúčom, ale s rôznymi UID a sandboxami.

1.6.13 Overovanie aplikácií - Verify Apps

Od verzie 4.2 je v OS Android možnosť overovania aplikácií. Používateľ môže povoliť overenie aplikácie ešte pred jej inštaláciou. Môže tak byť upozornený na potenciálne nebezpečnú aplikáciu a dokonca môže tento overovač zablokovať inštaláciu pokiaľ sa jedná o škodlivú aplikáciu. Pokiaľ je všetko v poriadku, naďalej pravidelne skenuje aplikácie na zariadení (aj pri inštalácii), vyhodnocuje ich správanie pomocou cloudovej služby a v prípade detekcie škodlivej aplikácie zobrazí varovanie, alebo ju zablokuje. V oboch prípadoch používateľa vyzve na jej odinštalovanie, ktoré je možné vhodným nastavení zautomatizovať. Takéto aplikácie už neskôr v budúcnosti nebude možné nainštalovať. Výhodou tejto aplikácie je aj fakt, že kontroluje aj aplikácie nainštalované z iných zdrojov ako oficiálny obchod Google Play.

1.7 Používateľské možnosti zabezpečenia

1.7.1 Fyzický prístup k zariadeniu

Prvým a hlavným spôsobom zabezpečenia androidového zariadenia je nastavenie fyzického prístupu k zariadeniu. Mobilné zariadenia sa dajú ľahko odcudzit' (alebo stratiť) a bez tejto ochrany by sa mohol útočník dostať ku všetkým dátam uloženým v zariadení (kontakty, fotografie, firemné dáta atď.), vedel by používať zariadenie na bežnú prevádzku ako legitímny používateľ alebo ho zneužiť na škodlivú činnosť (spam, botnety). Spôsobená škoda je jednak vo forme ceny zariadenia, rôznych účtov a faktúr, ak je zariadenie použité na finančné operácie, a tiež aj možné stíhanie v prípade spammingu. Zariadenia sa dajú chrániť dvomi spôsobmi:

- Nastavením PIN kódu - najzákladnejší spôsob zabezpečenia fyzického prístupu. Používateľ musí na odblokovanie zariadenia zadať štvormiestny PIN kód, pokiaľ ho po zvolenom počte pokusov zadá nesprávne, zariadenie sa zablokuje na stálo (na odblokovanie ho treba odniesť do servisu).

- Nastavením vzoru - pokročilejší spôsob zabezpečenia fyzického prístupu. Používateľ musí na odblokovanie zariadenia nakresliť prstom vopred zvolený vzor. Ako v predošlom prípade, pokiaľ ho na určitý počet pokusov nenakreslí správne, zariadenie sa uzamkne na stálo.

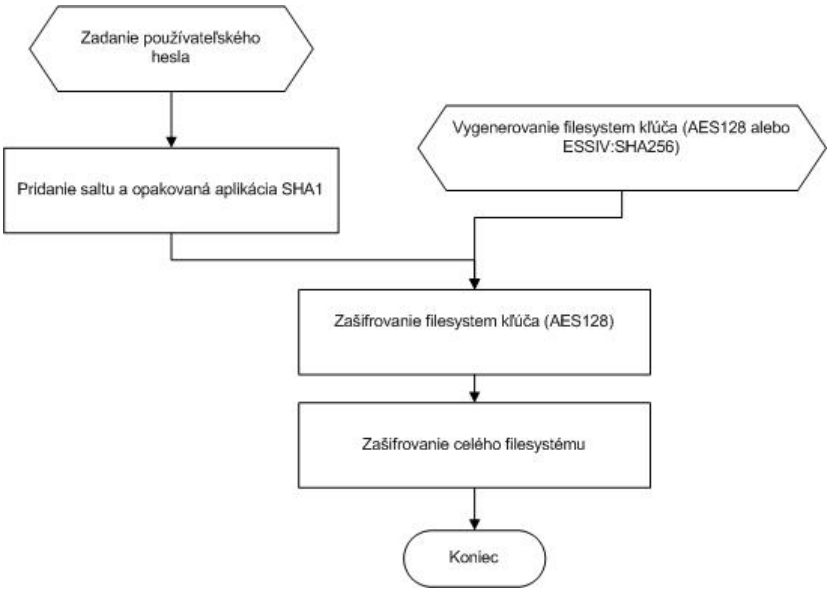
- Heslová ochrana - namiesto vzoru či PIN kódu je možné nastaviť aj alfanumerické heslo. Tento spôsob je však pomerne zdĺhavý a väčšina používateľov dáva prednosť predošlým dvom.
- Biometria - od uvedenia predného fotoaparátu sa experimentovalo s možnosťou odomykania zariadenia nasnímaním tváre používateľa. Tento spôsob sa však ukázal ako veľmi nepresný a pomalý. Vo verzii 6.0 však s pridaním snímača odtlačkov prstov pribudla možnosť odomykania pomocou odtlačkov prstov [48]. Tento spôsob je rýchly, efektívny a relatívne bezpečný, kvôli ukladaniu odtlačkov vo vrstve HAL.
- Smart Lock - od verzie 5.0 majú používatelia možnosť nakonfigurovať tzv. Smart Lock [50]. Táto technológia mala za úlohu zlepšiť používateľský komfort, keďže umožňovala mať zariadenie odomknuté pri splnení určitých podmienok. Tými boli napr. prítomnosť na známej, resp. dôveryhodnej lokalite, spárovanie s dôveryhodnými Bluetooth zariadeniami, či blízkosť používateľa. Splnením týchto podmienok nemusel používateľ neustále odomykať zariadenie, čo zredukovalo nutnosť odomykania zariadenia stále zložitejším heslom. Od verzie 7.0 boli pridané ďalšie senzorické vstupy, ktoré je možné využiť - blízkosť používateľovho tela, rozpoznanie jeho hlasu a obrazu jeho tváre.

Obe možnosti nastavenia sú dostupné pri prvotnom bootovaní zariadenia - po prinesení domov a rozbalení z krabice. Niektoré služby ako napr. Exchange Server vyžadujú konfiguráciu jednej z týchto ochrán pred tým, než sa vôbec používanie tejto služby povolí.

1.7.2 Šifrovanie súborového systému

Android od verzie 3.0 poskytuje kompletné šifrovanie súborového systému, viď obr. 1.12, takže všetky používateľské dáta môžu byť v jadre šifrované pomocou *dmcrypt* implementácie AES128 s CBC alebo ESSIV:SHA256. Šifrovací kľúč je chránený AES128 kľúčom odvodeným z používateľského hesla - tým sa zamedzí prístupu k dátam bez znalosti tohto hesla. Na ochranu pred systematickým hádaním hesla (brute force, rainbow tabuľky) je toto heslo kombinované s náhodným saltom a opakovane hashované pomocou SHA1 so štandardným PBKDF2 algoritmom predtým ako je použité na dešifrovanie kľúča súborového systému. Na ochranu pred slovníkovými útokmi Android vyžaduje určitú úroveň zložitosti hesla, ktoré sú nastaviteľné administrátorom zariadenia resp. vynútené operačným systémom. Toto šifrovanie funguje len s použitím hesla, uzamykanie pomocou nakreslenia vzoru nie je podporované. Hlavným dôvodom na pridanie tejto možnosti je zlepšenie ochrany dát pri strate alebo odcudzení zariadenia.

Z technického hľadiska sa dá táto možnosť považovať za úspešnú. Z pohľadu používateľov to však bolo pomerne nešťastné riešenie. Vždy keď sa zariadenie uzamklo, súborový systém bol zašifrovaný, a vždy keď ho chcel používateľ odomknúť, musel byť súborový systém dešifrovaný. Toto viedlo k nespokojnosti používateľov, lebo čas



Obr. 1.12: Schéma šifrovania súborového systému

odozvy zariadenia sa výrazne zvýšil oproti módu s vypnutým šifrovaním. Preto sa do verzií 4.x určených aj pre smartfóny táto možnosť nedostala.

Šifrovanie súborového systému sa vo verzii 5.0 vrátilo pre všetky zariadenia [48]. Oproti verzii 3.0 však bolo kompletne prerobené, a je povinné zapnuté pre všetky nové zariadenia (s predinštalovaným Android 5.0 a vyššie), môže však byť z rozhodnutia výrobcu zariadenia vypnutá alebo voliteľná [28]. Na starších zariadeniach, ktoré sa aktualizujú na verziu 5.0, môže byť táto funkcionalita tiež povolená. Napriek zlepšeniu výkonnosti šifrovacích algoritmov a celkovej optimalizácii šifrovanie stále nedosahuje ideálne výsledky. Testy [27] preukázali, že celkový výkon zariadení so zapnutým šifrovaním sa v niektorých prípadoch môže znížiť až o 50%.

Vo verzii 7.0 sa toto šifrovanie zmenilo na tzv. file-level šifrovanie, teda na nezávislé šifrovanie jednotlivých súborov [49]. Podobne ako v predošlých verziách je šifrovanie zariadenia povinne zapnuté. Tento nový spôsob má však dve podstatné výhody. Prvým je zvýšenie výkonu zariadenia, keďže šifrovanie a dešifrovanie celého súborového systému menej výkonné zariadenia výrazne spomaľovalo. Druhým je možnosť použitia metódy AEAD (Authenticated Encryption with Associated Data) [51]. AEAD výrazne sťažuje prístup k dátam pre neautorizovaného používateľa alebo aplikáciu.

1.7.3 Direct boot

Spojitosť s file-level šifrovaním plne využíva funkcia direct boot. Pokiaľ sa zamknuté zariadenie reštartuje, tak niektoré aplikácie budú stále prístupné, napr. telefón, budík

a pod., hoci s limitovaným prístupom. To znamená, že na ich obmedzené používanie netreba odomykať a dešifrovať celé zariadenie. Po odomknutí zariadenia sa sprístupnia aj ostatné aplikácie [49].

1.7.4 Heslová ochrana

OS Android môže byť nakonfigurovaný aby overoval zadané heslo predtým ako povolí prístup k zariadeniu. Okrem zabráneniu neautorizovanému prístupu k zariadeniu toto heslo navyše chráni kryptografický kľúč ku kompletnému šifrovaniu súborového systému. Zložitosť hesla a jeho použitie môže byť nastavené administrátorom zariadenia. Od verzie 8.0 je heslo potrebné aj na odomknutie vývojárskych možností [37].

1.7.5 Administrácia zariadenia

Android od verzie 2.2 poskytuje tzv. Android Device Administration API, ktoré ponúka možnosti administrácie zariadenia na systémovej úrovni. Napr. bola pridaná podpora Exchange do vstavanej e-mailovej aplikácie. Cez túto aplikáciu môžu Exchange administrátori vynucovať heslové politiky alebo dokonca vzdialene vymazávať odcudzené alebo stratené zariadenia.

1.7.6 Uchovávanie credentials

V štandardnom nastavení Android uchováva súbor preddefinovaných certifikačných autorít (CA), ktoré sú dôveryhodné (trusted) pre operácie ako napr. nadväzovanie SSL spojenia v internetovom prehliadači. Od verzie 4.0 môžu používatelia rušiť predinštalované CA v systémových nastaveniach. Používatelia môžu importovať nové dôveryhodné CA do systému cez USB z nejakého úložiska. Android 4.1 pridal možnosť pridania hardvérového KeyChain úložiska, ktoré viaže privátne kľúče k zariadeniu, na ktorom sú uložené.

1.7.7 VPN

Android obsahuje zabudovaného VPN klienta s podporou PPTP, L2TP a IPsec VPN. Verzia 4.0 predstavila triedu *VpnService* pre podporu VPN riešení od tretích strán. Verzia 4.2 predstavila možnosť nastavenia VPN do módu permanentného zapnutia (always on), aby sa aplikácie pripájali k sieti len pri pripojení cez VPN [24]. Od verzie 4.4 je možné používať tzv. per-user VPN pri zariadeniach s viacerými používateľskými profilmi [48].

1.7.8 Model viacerých používateľov na jednom zariadení

Architektúra smartfónov predpokladá jedného fyzického používateľa. Tento fakt ale nebráni možnosti mať na jednom zariadení vytvorených viacej používateľských profilov podobne ako na klasickom PC. Táto možnosť tak môže slúžiť na oddelenie súkromných

a pracovných dát. Takýto návrh predstavili už v r. 2009 Liu et. al. vo svojom článku [86]. Ďalším spôsobom oddeľovania súkromných a pracovných dát môže byť napr. virtualizácia. Od verzie 5.0 je v OS Android možnosť nastavenia viacerých používateľských profilov, "restricted" profilov a hosťovských (guest) profilov na zariadeniach [48]. Využiť sa to dá napr. pri "rodinných" zariadeniach ako napr. tablety, alebo keď používateľ potrebuje niekomu na chvíľu požičať svoje zariadenie, ale nechce povoliť prístup k svojim dátam a aplikáciam.

1.7.9 Falšovanie citlivých informácií

Ďalším spôsobom ako sa môžu používatelia chrániť pred únikov citlivých informácií je inštalácia aplikácií, ktoré buď priamo generujú falošné citlivé informácie (SMS, záznamy v kalendári atď.) alebo ich nejakým spôsobom falšujú a vydávajú za pravé. Krátky popis týchto aplikácií je v článku od Encka [85]. Možnosť "podhadzovania" falošných dát aplikáciám, ktoré vyžadujú na inštaláciu povolenie na prístup k nim, ale reálne ich nepotrebujú prestavili Beresford et. al. [100]. Tento systém zvaný MockDroid je upravenou verziou OS Android, pridáva ale možnosť každej nainštalovanej aplikácií upraviť nastavenia povolení tak, aby nemohla pristupovať k zdrojom, ktoré si používateľ zvolí, napr. k lokácii zariadenia, prístupu k sieti, posielaniu/čítaniu SMS správ, k identifikátoru zariadenia atď. Po nastavení tento systém pri volaní týchto zdrojov poskytuje buď konštantné falošné hodnoty (identifikátory) alebo simuluje nenadviazanie spojenia so sieťou mobilného operátora resp. nemožnosť pripojenia k Internetu.

1.8 Služby

1.8.1 Google Play

Google Play (vznikla zlúčením služieb Android Market a Google Music 6. marca 2012), je oficiálna distribučná služba aplikácií určených pre platformu Android, ktorú vyvinula a spravuje spoločnosť Google. Táto služba umožňuje používateľom prezerať a sťahovať nielen programy a aplikácie ale aj hudbu, časopisy, knihy, filmy a mnohé iné dáta, ktoré boli zverejnené prostredníctvom Google. Aplikácie sú poskytované buď bezplatne alebo za určitý poplatok, v závislosti od aplikácie. Aplikácie možno sťahovať priamo zo zariadenia alebo vo webovom prehliadači.

Zabezpečenie Google Play je rozdelené na viac častí. Keďže je Android otvorená platforma, je prístupná veľkému množstvu vývojárov. Pri tempe vydávania aplikácií aké je v súčasnosti je nemožné všetky tieto aplikácie podrobne kontrolovať a schvaľovať, tak ako to robí Apple na svojom AppStore. Google sa skôr spolieha na prevenciu, detekciu škodlivých aplikácií a rýchle riešenie vzniknutých problémov [31], [32], [33], [35]:

- Kontrola návrhu a audit bezpečnostnej architektúry pred vydaním novej verzie OS.

- Kontrola a skúmanie aktuálnej verzie OS komunitou vývojárov a expertov - lepšie odhaľovanie chýb a zlepšenie zabezpečenia.
- Rýchla odozva na hlásené incidenty - aktívne zapojenie používateľov.
- Pravidelné previerky a sťahovanie nebezpečných aplikácií z Google Play (a následné blokovanie vývojárskych účtov).
- Google Bouncer - kontrola aplikácií - služba Bouncer kontroluje všetky nové aplikácie po vzoru AppStoru (vrátane aktualizácií), vývojárske účty a spätne aj staršie aplikácie.
- Od verzie aplikácie Google Play Store 3.9.16 je zapracovaná lokálna kontrola inštalovaných aplikácií. Používatelia sú varovaní pred možným rizikom (písomne aj varovnými ikonami), prípadne je inštalácia konkrétnej aplikácie zablokovaná.

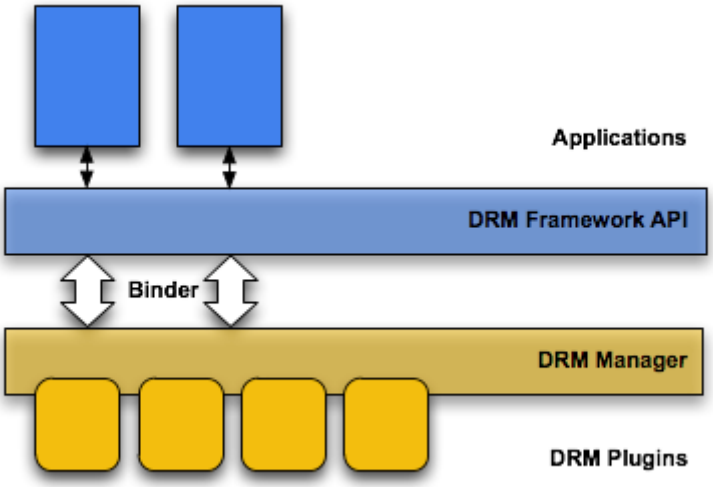
Obr. 1.13: Google Play [33]

1.8.2 Digital Rights Management

Android ako platforma poskytuje rozšírený DRM framework, ktorý umožňuje aplikáciám spravovať obsah chránený autorskými právami [24]. Podporuje veľa DRM schém,

každé zariadenie podporuje tie schémy, ktoré určí výrobca zariadenia. Framework je implementovaný v dvoch vrstvách, graficky znázornené na obrázku 1.14:

- API, ktoré je viditeľné pre aplikácie cez aplikačný framework a beží v DVM pre štandardné aplikácie.
- DRM manažér písaný v natívnom kóde implementujúci DRM framework a zviditeľňujúci rozhranie pre DRM agentov, ktoré riadia správu práv a dešifrovanie rôznych DRM schém.



Obr. 1.14: Architektúra DRM na platforme Android [24]

1.8.3 Safe Browsing

Služba Safe Browsing [50] bola uvedená do prehliadača pre klasické PC Google Chrome v roku 2005. Jej ochranná funkcia spočíva v poskytovaní kontroly URL adries pre klientské aplikácie. Pred pripojením sa na neznámu URL adresu sa aplikácia, napr. prehliadač, dopytuje na zoznam nebezpečných URL adries, kde sú zaznamenané rôzne phishingové weby a pod. Pokiaľ sa dopytovaná adresa nachádza v zozname, prehliadač vypíše varovanie pre používateľa. Táto služba je pre OS Android, resp. mobilnú verziu prehliadača Chrome dostupná od roku 2015.

1.8.4 Google Play Protect

Táto služba bola uvedená v roku 2017 ako súčasť verzie 8.0. Vo svojej podstate ide o zjednotenie všetkých bezpečnostných mechanizmov v OS Android poskytovaných

spoločnosťou Google do jednej ucelenej aplikácie, resp. frameworku [36]. Táto služba napr. zahŕňa:

- Verify Apps - ochrana pred malvérom.
- Find My Device - aplikáciu na vyhľadanie strateného zariadenia.
- Safe Browsing - ochrana pred prístupom na nebezpečné weby.

1.9 Súhrnný prehľad bezpečnostných mechanizmov v OS Android

V tabuľke 1.1 je zhrnutý vývoj bezpečnostných mechanizmov v OS Android, v závislosti od verzie systému. Ako je v nej vidieť, dôraz na lepšie zabezpečenie systému sa začal klásť až od verzie 4.0 pridaním ASLR či KeyChain-u. V ďalších verziách 4.x boli pridané zabezpečenie komunikácie Certificate Pinning-om či Premium SMS správy. Vo verzii 5.0 vývojári pridali šifrovanie súborového systému a TLS zabezpečenie spojenia. Najviac podstatných zmien z hľadiska bezpečnosti sa zatiaľ udialo vo verzii 6.0 - boli pridané Runtime permissions, Verified boot a možnosť používania odtlačkov prstov na odomknutie zariadenia.

Tabuľka 1.1: Vývoj bezpečnostných mechanizmov v OS Android

	1.5	1.6	2.0	2.2	2.3	3.x	4.0	4.1	4.2	4.3	4.4	5.x	6.x	7.x	8.x
Aplikačný sandbox	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Systémová partícia	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Safe mód	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Krypto knižnice	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
KeyChain	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
TLS v1.x	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x
Verified boot	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x
ASLR	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
Podpora PIE	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
SELinux	-	-	-	-	-	-	-	-	-	x	x	x	x	x	x
Model povolení	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
App Ops/Runtime	-	-	-	-	-	-	-	-	-	x	-	-	x	x	x
Premium SMS	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x
Certificate Pinning	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x
Kontrola Gooogle cert.	-	-	-	-	-	-	-	-	-	-	x	x	x	x	x
Strict Mode	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x
Overovanie aplikácií	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x
PIN/vzor/heslo	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Odtlačky prstov	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x
Šifrovanie FS	-	-	-	-	-	x	-	-	-	-	-	x	x	x	x
Administrácia zar.	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x
VPN	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x
Profily na zar.	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x
GP Bouncer	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x
Seamlsess updates	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x
Key attestation	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x
File-level šifrovanie	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x
Direct boot	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x
GP Protect	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x
Project Treble	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x

1.10 Upravené verzie OS Android - Custom ROMs

Na záver tejto kapitoly spomenieme komunitné úpravy OS Android zvané aj Custom ROMs. Všetky tieto upravené "Androidy" vychádzajú z "vanilla" Androidu, teda z verzie, ktorá bola vývojármi uvoľnená pre verejnosť. Existuje viacero takto upravených verzií Androidu, ich spoločnými znakmi sú napr.:

- Práca s "čistým" Androidom, bez komponentov pridaných výrobcami zariadení.
- Zvýšená kontrola nad zariadením, skoro vždy je používateľ v "Superuser (su)" móde.
- S tým spojená vyššia kontrola nad súkromím a používateľskými dátami.
- Predlžovanie životného cyklu zariadení - veľké množstvo zariadení stráca podporu novej verzie OS Android a musí fungovať na starej. Pokiaľ teda chce používateľ využívať výhody novej verzie OS, musí si kúpiť nové zariadenie.

V nasledujúcich častiach si spomenieme niekoľko najpoužívanejších upravených verzií OS Android.

1.10.1 CyanogenMod/LineageOS

CyanogenMod (CGM) je vylepšený open source OS založený na OS Android [38]. Nová verzia prichádza vždy po niekoľkých mesiacoch od vydania "vanilla" Androidu. Na rozdiel od Androidu, ktorý je v zariadeniach výrobcov, tento Android je v "čistej" forme, teda bez rôznych pridaných aplikácií. Obsahuje tiež viac možností nastavenia bezpečnostných prvkov a dovoľuje aj meniť výkonnostné nastavenia zariadenia. Od prvých verzií bol v ňom zakomponovaný SELinux, Open VPN klient, aplikácia na manažment povolení Privacy Guard, odomknutý bootloader a *root* prístup k zariadeniu. Čo sa týka výkonu, je v ňom možné meniť taktovaciu frekvenciu procesora, meniť jeho napätie či lepšia práca s pamäťou zariadenia. Veľké množstvo používateľov označuje CGM ako stabilnejší Android [40]. Aktuálne je pomerne ťažké porovnať CGM s OS Android, pretože rozdiely medzi nimi sa postupne stierajú:

- Vizuálne sú veľmi podobné, keďže používatelia sú navyknutí na používateľské rozhranie OS Android, nebolo pre vývojárov CGM potrebné meniť túto časť OS.
- V zariadení s CGM sa nenachádza žiadny spyware či bloatware od výrobcov, keďže aj tieto môžu narušať súkromie používateľov [41]. Tieto aplikácie v OS Android nie je buď možné odstrániť, alebo to ide len veľmi zložitým spôsobom.
- CGM obsahuje od začiatku prvky SELinux, Android ich pridal až od verzie 4.3.
- CGM poskytuje skúseným používateľom možnosti na modifikovanie výkonu a nastavení hardvéru svojich zariadení. Používatelia majú tiež prístup k bootloaderu a *root* právam. V OS Android používatelia takéto možnosti nemajú.

- Od verzie 11 môžu používatelia spravovať povolenia a súkromie pomocou aplikácie Privacy Guard [42]. Používatelia takto môžu obmedziť prístup aplikácií k citlivým dátam a systémovým zdrojom pomocou modelu povolení. Táto možnosť bola v OS Android krátko prítomná len vo verzii 4.3. Vo verzii 6.0 sa táto funkcionalita vrátila, správa a ochrana súkromia však v OS Android stále chýba.
- V CGM je možnosť úpravy povolení pre systémové aplikácie. Týmto spôsobom môžu používatelia zabrániť aplikáciám "zobudiť" zariadenie a šetriť tak batériu [42].
- CGM je dostupný pre širšiu škálu zariadení, keďže novšie verzie OS Android nie sú podporované na starších zariadeniach a takto predlžuje životný cyklus zariadení.
- Komunita starajúca sa o CGM rýchlo reaguje na rôzne bezpečnostné hrozby a poskytuje pravidelné aktualizácie. Táto výhoda sa však postupne vytráca, keďže už aj Google si osvojil túto politiku rýchlych bezpečnostných záplat.

Podpora CyanogenMod bola ku koncu roka 2016 ukončená, ale jeho nástupcom sa stal LineageOS [39], na ktorom pracuje časť vývojárov CGM.

1.10.2 MIUI

Druhým veľkým výrobcom upraveného OS Android je spoločnosť Xiaomi so svojím OS MIUI [43]. Jeho špecifikom je, že výrobca ho priamo nasadzuje do vlastných zariadení. Keďže vychádza z OS Android, tak sa výrobca sústreďuje hlavne na vizuálne úpravy používateľského rozhrania a rôznych aplikácií pre svoj okruh používateľov, resp. špecifické pre východoázijský trh. V predošlých verziách sa však objavilo viacero užitočných vylepšení [44]:

- Šetriaci mód, ktorý výrazne zlepšuje výdrž batérie keď je zariadenie v stand-by móde.
- Zabudovaná aplikácia Permissions riadi prístup aplikácií k používateľským dátam. Týmto používateľ dokáže predchádzať úniku citlivých dát.
- Obsahuje tiež vlastnú aplikáciu Virus Scanner, ktorý kontroluje aplikácie pred inštaláciou a dokáže takto odhaliť škodlivé aplikácie.
- Taktiež zabudovaná aplikácia Clean Master slúži na čistenie a optimalizáciu práce s pamäťou, urýchľuje chod zariadenia a čiastočne chráni zariadenie pred rôznymi hrozbami.
- Pomocou aplikácie Keyword Filter vie používateľ filtrovať správy či emaily spamového charakteru.

- Okrem toho dokáže zabudovanou aplikáciou sledovať internetové pripojenia jednotlivých aplikácií, či už ide o rýchlosť pripojenia, cieľ pripojenia alebo objem prenesených dát.

Na záver môžeme skonštatovať, že veľké množstvo vylepšení z upravených verzií OS Android sa dostalo do oficiálnej vývojovej vetvy OS Android. Týmto spôsobom sa zmenšujú rozdiely medzi upravenými a oficiálnou verziou OS Android, čo vo veľkej miere prispelo k zlepšeniu celkovej bezpečnosti OS Android. Aktuálny trend naznačuje, že väčšina upravených verzií OS Android prináša len kozmetické zmeny a postupne stráca používateľskú základňu, ako napr. Paranoid Android, AOKP, či Replicant [45], alebo sa pretransformovalo v štandardné spoločnosti a svoj OS poskytujú pre komerčne predávané zariadenia, ako napr. MIUI, či CGM.

Kapitola 2

Súčasný výskum v oblasti bezpečnosti OS Android

V predchádzajúcej kapitole sme si ukázali aké sú v súčasnosti možnosti zabezpečenia zariadení s OS Android. Ako otvorená platforma je Android dokonalým cieľom pre rôzne pokusy o narušenie jej bezpečnosti. OS Android bol vyvíjaný do veľkej miery "za pochodu", takže zo začiatku obsahoval rôzne chyby a zraniteľnosti, ktoré boli postupne odstraňované. V súčasnosti je úroveň zabezpečenia platformy na pomerne dobrej úrovni, ale napriek tomu sa stále nájdu problémy, ktoré treba odstrániť. O nich bude táto kapitola pojednávať.

2.1 Problémy fyzického prístupu k zariadeniu a postranné kanály

V prvej časti tejto kapitoly si predstavíme špecifické problémy spojené:

- s malými rozmermi zariadení a s tým súvisiacou kontrolou prístupu,
- so špecifickými senzormi využívanými v zariadeniach
- a s útokmi pomocou postranných kanálov.

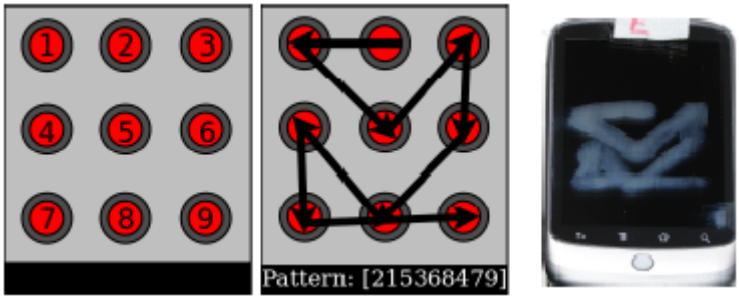
Jedná sa síce len o okrajovú časť problematiky (čo sa zameraniu výskumu týka), ale škody spôsobené týmito problémami môžu výrazne ohroziť súkromie používateľov.

2.1.1 Problémy fyzického prístupu k zariadeniu

Jedným z najjednoduchších spôsobov ako zneužiť klasický mobilný telefón je ukradnúť ho a útočník z neho môže bez problémov volať a posielať SMS správy. Pri smartfónoch je tento prístup rovnako účinný za predpokladu, že nie je nastavené zabezpečenie fyzického prístupu k zariadeniu. Ako bolo spomenuté v predošlej kapitole, toto nastavenie je odporúčané vykonať hneď na začiatku práce so zariadením, resp.

dodatočne, ak si to niektoré aplikácie alebo služby vyžiadajú.

Výskum však ukazuje, že aj tento spôsob je nepostačujúci. Počet PIN kódov je obmedzený počtom zadávaných miest, navyše aj počet vhodných vzorov je menší než koľko sa predpokladá. Je to dôsledkom politík pri nastavovaní vzoru - minimálny počet bodov, cez ktoré musí vzor prechádzať, minimálny počet "otočení" vzoru atď. Aviv a kol. [55] študovali šmuhy, ktoré ostávali na dotykovej obrazovke po zadávaní PIN kódu a kreslení vzoru. Tieto šmuhy a odtlačky ostávajú na miestach zadávania PIN kódu a kreslenia vzoru - z toho bol odvodený názov smudge attacks. Tieto stopy je možné s dostatočne dobrou fotografickou výbavou odfotografovať a zanalyzovať. Autorom sa podarilo v skoro 70% prípadov získať použiteľný vzor a pomocou neho sa dostať do "odcudzeného" zariadenia. Ochranou pred týmto typu útoku môže byť jednoduché utieranie displeja pred odložením telefónu, resp. neodkladať telefón displejom nahor na verejnom mieste (v bare, reštaurácii...).



Obr. 2.1: Zľava: obrazovka zariadenia pred zadaním vzoru, zvolený testovací vzor a jedna z výsledných fotografií [55]

Aviv a kol. nadviazal na svoj výskum v r. 2012 článkom [56] čiastočne vychádzajúcom z [70], v ktorom sa zamerali na odchytávanie PIN kódov a vzorov pomocou dát získaných z akcelerometra. Vytvorená aplikácia pracovala v kontrolovanom (sediaci používateľ) a nekontrolovanom prostredí (kráčajúci používateľ). Výsledky mali pomerne pôsobivé: v kontrolovanom prostredí bola úspešnosť odhalenia PIN kódu 43% a vzoru 73%, v nekontrolovanom prostredí 20% resp. vzoru 40%, na maximálne 5 pokusov. Tento výskum má ešte značné medzery, napr. ako dostať monitorovaciu aplikáciu do zariadenia, či vysokú citlivosť akcelerometra pri chôdzi, ale smeruje k doteraz nie veľmi skúmanej oblasti postranných kanálov v mobilných zariadeniach.

Podobným smerom výskumu sa vydali aj Cai a Chen vo svojej práci [57]. Autori sa zamerali na monitorovanie pohybu zariadenia pomocou akcelerometra pri zadávaní znakov na numerickej klávesnici. Použili na to vlastný nástroj TouchLogger, s ktorým po natrénovaní, klasifikácii, evaluácii príznakov a vyhodnotení zistili úspešnosť vyše 70%. V neskoršej verzii [58] sa autori zamerali na vyriešenie niekoľkých významných

otázok:

- rozdielnosť použitého hardvéru v zariadeniach
- variácia rozmerov zariadení - prechod aj na tablety
- variácie v rozložení klávesnice
- variácie v štýle písania používateľov
- výber použitého senzoru - akcelerometer alebo gyroskop
- výber vhodnej rozhodovacej techniky

Okrem toho svoj výskum rozšírili aj na znaky abecedy, čo malo za následok zhoršenie celkových rozlišovacích schopností aplikácie TouchLogger - 33% v prípade písmena a okolo 50% v prípade čísla na numerickej klávesnici. Na numerickej klávesnici po vyskúšaní 81 4-miestnych PIN kódov je pravdepodobnosť správneho PIN 65%. Problémom tohto postupu je, podobne ako v prvom prípade, spôsob zavedenia aplikácie do zariadenia.

Na našom ústave sme sa snažili na túto tematiku nadviazať v rámci diplomových prác. V rokoch 2014 a 2015 boli vypracované práce Švandu [60] a Varcholu [61]. Výsledkom prác je systém na odomknutie zariadenia pomocou gesta. Systém využíva genetické algoritmy na rozpoznanie gesta z akcelerometra zariadenia na jeho odomknutie. Systém bol dôkladne otestovaný a vykazuje relatívne malú chybovosť (v závislosti na zložitosti a dôkladnosti natrénovania gesta) a aj odolnosť voči odpozeraniu gesta útočníkom.

Najnovším príspevkom v tejto oblasti je práca Uellenbecka a kol. [59]. Predstavený spôsob prihlasovania sa je odolný voči odpozeraniu šmúh na displeji zariadenia či "naťukávaného" PIN kódu. Využíva hardvérový "one-time pad", kde používateľ musí po zadaní PIN kódu zadať ešte dodatočný verifikačný kód, ktorý je výsledkom súčtu PIN kódu a počtu vibrácií zariadenia. Počet vibrácií je vždy rôzny. Odolnosť celého systému spočíva v dvoch faktoroch: vibrácie sú neviditeľné pre útočníka (teda pokiaľ zariadenie nie je položené na stole a teda je počuť) a vibrovanie prebieha vždy v rovnakom časovom intervale, bez ohľadu na ich počet. Útočník teda nevie odhadnúť ich počet na základe trvania vibrovania.

2.1.2 Problémy technológie NFC

Ďalším problémom prepájajúcim fyzické zabezpečenie zariadenia s postrannými kanálmi je problém bezpečnosti technológií RFID a NFC. Tieto technológie sa v mobilnej sfére používajú na vykonávanie platieb pomocou mobilného zariadenia "spárovaného" s platobnou kartou používateľa. Existuje veľa publikácií a článkov, kde sa autori venujú zneužívaniu dier v bezpečnosti týchto technológií a načrtávajú možné riešenia, napr.

[64], [65], [66] či [67]. Zaujímavá myšlienka ako zvýšiť bezpečnosť týchto technológií je článok Shresthu a kol. [62]. Autori tu predstavili možnosť dodatočnej identifikácie platobného terminálu na základe informácií z okolia - teploty, nadmorskej výšky, vlhkosti vzduchu a plynového senzora. Táto technológia dosahuje v testoch veľmi dobré výsledky, lebo je veľmi ťažké pre útočníka manipulovať s uvedenými hodnotami vonkajšieho prostredia. Okrem toho, spracovanie výsledkov je veľmi rýchle, a systém navyše zaručuje aj vysokú mieru ochrany súkromia používateľov.

2.1.3 Postranné kanály

Prvým zástupcom v tejto časti je práca Kenworthyho a Rohatgiho [68]. V tejto práci použili klasický prístup techniky postranných kanálov. Na troch zariadeniach mali spustené aplikácie vykonávajúce RSA (merané boli operácie Modular Square a Modular Multiply), ECC (Point Double, Point Add) a AES (CBC šifrovanie). Na každom zariadení spustili príslušnú aplikáciu a jednoduchou anténou s digitalizérom (sonda) merali elektromagnetické vyžarovanie zo zariadení. Výsledky pozorovaní boli nad očakávania dobré:

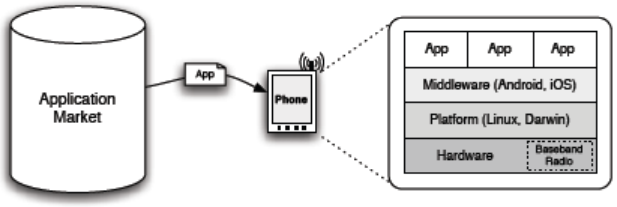
- Exponenty dp a dq boli odhalené z jedinej 2048-bitovej RSA-CRT operácie (sonda bola v blízkosti zariadenia).
- Tajná správa m bola odhalená z jednej operácie ECC nad P-571 (sonda bola vo vzdialenosti 3 metrov od zariadenia).
- Po 12500 AES-CBC operáciách boli identifikované miesta kde takmer s istotou dochádzalo k úniku informácií spracovateľnými ďalšími metódami.

Tento článok z r. 2012 ukazuje, že každé zariadenie môže byť takýmto spôsobom "odpočúvané", pokiaľ útočník vie akú technológiu zariadenie využíva. Takto sa napríklad dá ohrozit' technológia NFC - jednoduchým umiestnením sondy do blízkosti platobného terminálu môžeme čítať všetky dáta v príslušnej komunikácii. Nevýhodou tohto útoku je jeho náročnosť na hardvérové vybavenie, resp. na jeho rozmery - na dôkladné zamaskovanie musia byť sondy čo najmenšie.

Druhou významnou prácou je článok Genkina a kol. z roku 2016 [69]. Autori článku do veľkej miery zjednodušili vyššie spomenutý útok a úspešne ho aplikovali na ECDSA algoritmus. Tento algoritmus bol spustený na mobilnom telefóne a autori merali elektromagnetické vyžarovanie tohto telefónu. Použité boli pritom veľmi jednoduché a lacné sondy a senzory. Vykonanie takéhoto typu útoku je v súčasnosti časovo aj finančne nenáročné. Autori navyše odhalili zraniteľnosti vo viacerých kryptografických knižniciach, napr. staršie verzie OpenSSL (1.0.x, 1.1.x), iOS do verzie 9.x, CoreBitcoin či Bouncy Castle pre Android. Väčšina z týchto knižníc sa v súčasnosti už nepoužíva.

2.2 Aplikačná bezpečnosť

Bezpečnosť aplikácií je najväčším problémom platformy Android. Vďaka jej otvorenosti môžu rôzni vývojári publikovať svoje aplikácie na rôznych miestach, najčastejšie na rôznych serveroch a fórach (ilustrované na obr. 2.2). Tieto zdroje s výnimkou oficiálneho Google Play nezabezpečujú nijakú ochranu voči potenciálne nebezpečným aplikáciám. V tejto časti práce si ukážeme rôzne typy škodlivých aplikácií, ich funkcionalitu a životný cyklus, spôsoby prenosu, niekoľko možných útokov a nakoniec aj zopár návrhov ako tieto aplikácie detegovať a zabrániť tak prípadnému útoku.



Obr. 2.2: Mechanizmus získavania aplikácií na platforme Android [85]

2.2.1 Malvér

Podrobný prehľad a charakteristiku existujúceho malvéru publikovali v roku 2012 Zhou a Jiang [82]. S rastúcim počtom predaných zariadení úmerne stúpal aj počet vytvorených škodlivých aplikácií. Autori ako prví zozbierali ucelenú databázu vzoriek androidového malvéru (1260 záznamov). Pomocou nej ich dokázali rozdeliť do špecifických rodín (49) a vytvoriť časovú os ich evolúcie.

Charakteristika malvéru

Škodlivé aplikácie môžeme rozdeliť do viacerých kategórií. Podľa spôsobu inštalácie ich delíme na:

- Repackaging alebo znovu zabalené aplikácie. Asi najčastejší spôsob šírenia malvéru, útočník stiahne legitímnu aplikáciu, upraví ju (doplní škodlivú funkcionalitu) a publikuje na market - v drvivej väčšine prípadov sa útočníci vyhýbajú oficiálnym marketom, kde je možnosť kontroly a zachytenia takto upravenej aplikácie. Takýmto spôsobom sa upravujú všetky typy aplikácií od hier, správcov súborov až po rôzne platené aplikácie. Autori používajú rôzne techniky na zamaskovanie skrytej funkcionality, od "nevinných" názvov metód po obfuskáciu častí kódu.

- Útok prostredníctvom aktualizácie je technika podobná predošlej. Rozdiel je v tom, že do legitímnej aplikácie sa doplní len aktualizačná funkcia, ktorá stiahne škodlivú časť kódu ako legitímnu aktualizáciu aplikácie.
- Drive-by stiahnutie je podobné klasickým vírusom z PC. Cez prehliadač sa používateľ pripojí na falošnú stránku, ktorá mu ponúkne na stiahnutie nejakú úžasnú aplikáciu (napr. na zvýšenie výdrže batérie), no v skutočnosti si stiahne do zariadenia malvér.
- Ostatné spôsoby zahŕňajú napr. spyware aplikácie na monitorovanie zariadenia, aplikácie maskujúce sa za legitímne aplikácie atď.

Po nainštalovaní sa aplikácia musí aktivovať, aby bolo možné využívať jej škodlivú funkcionalitu. Na aktiváciu aplikácií sa vo všeobecnosti používajú systémové udalosti, napr. prijatie SMS správy, nabootovanie OS a podobne. Tieto dve spomenuté sú aj dve najčastejšie udalosti, pomocou ktorých sa môže malvér aktivovať. Ich autori sa snažia aj tieto spôsoby maskovať, najčastejšie využívaním zriedkavých udalostí (zapnutie Bluetooth, pripojenie k PC), resp. používajú zriedkavé kombinácie systémových udalostí (súčasná zmena Wi-Fi siete a zapnutie GPS).

Po aktivácii aplikácie nastáva spustenie útoku. Podobne ako vírusy na PC, aj tie na mobilných zariadeniach sledujú takmer rovnaké ciele:

- Využitie root exploitov na získanie root práv a následné spustenie útokov privilege escalation (budú spomenuté v nasledujúcej podkapitole).
- Získať vzdialenú kontrolu nad zariadením a zapojiť ho do botnetu. Takýmto spôsobom je potom jednoduché vykonávať útoky typu DoS a DDoS. Zariadenia sú ovládané z kontrolných serverov, ktorých adresy sú uložené v aplikácii v šifrovanej podobe.
- Vykonávanie finančných operácií pomocou platených SMS správ na prémiové čísla. Autori takýchto aplikácií často odchytávajú potvrdzovacie správy a legitímny vlastník zariadenia sa o útoku dozvie až z faktúry od operátora.
- Zber informácií je tiež rozšíreným spôsobom útočenia. Ide hlavne o zber kontaktov, odchytávanie správ, elektronickej komunikácie a všetkých prístupových údajov
- Pokiaľ aplikácie neobsahujú žiadne exploity, ich použitie je limitované pridelenými povoleniami. Zaujímavé je, že väčšina malvéru a legitímnych aplikácií sa v povoleniach veľmi nelíši.

Evolúcia malvéru

Na porovnanie evolúcie malvéru v priebehu jedného roka si Zhou a Jiang vybrali dve najpočetnejšie rodiny:

- Rodina DroidKungFu sa objavila v r.2011, odvtedy je známych minimálne 6 verzií. Využívajú rôzne druhy exploitov a tiež rôzne druhy ich šifrovania. Všetky obsahujú informácie o riadiacich serveroch, uložené sú rôzne, od otvorenej formy cez štandardné až po vlastné šifrovacie mechanizmy. Inštalujú skryté aplikácie, ktoré pokračujú v činnosti aj po odstránení originálnej aplikácie, na maskovanie používajú obfuskáciu zdrojového kódu.

- Rodina AnserverBot sa objavila niekoľko mesiacov po DroidKungFu a je pokladaná za jednu z najsofistikovanejších. Využíva metódy antianalýzy - sleduje či originálna aplikácia, z ktorej sa inštaloval, nebola modifikovaná, obfuskuje triedy, metódy a všetky ich volania. Navyše aktívne deteguje bezpečnostný softvér a používa dva druhy riadiacich serverov - jeden na prijímanie príkazov a druhý na aktualizáciu dát aplikácie.

Zaujímavosťou je, že po vykonaní testov detekcie škodlivých aplikácií štyrmi antivírovými aplikáciami bola zistená maximálna úspešnosť len necelých 80%, čo nás privádza k ďalšej časti a tou sú spôsoby detekcie malvéru na mobilných platformách.

V roku 2013 na prácu Zhoua a Jianga nadviazal Suarez-Tangil a kol. [83]. Autori sa najprv zamerali na bezpečnostnú analýzu OS Android a jej nedostatky z pohľadu možností infikovania zariadenia malvérom. V ďalšej fáze sa zamerali na spôsoby infekcie zariadenia, podobne ako v predošlom článku. Túto časť rozširujú o nové poznatky, napr. o rootkity či grayware, vrátane konkrétnych príkladov (Aurora Feint, iPhone/Mobileconfigs). V tomto článku sa prvýkrát objavuje aj komplexnejší prehľad o spôsoboch detekcie malvéru. Popísané sú typy detekcie a monitorovania aplikácií, granularita detekcie, typy analýzy a identifikácie, miesta, kde sa majú tieto procesy vykonávať a pod. Práca je veľmi obsiahla, obsahuje veľké množstvo tabuliek a obrázkov s rozdeleniami a výsledkami testov a analýz. Taktiež spracováva prehľad aktuálne dostupných riešení, niektoré z nich sú popísané v podkapitole Detekcia malvéru.

Aktuálna situácia

Komplexnejší prehľad o aktuálnej situácii na poli mobilného malvéru určeného pre OS Android nám poskytuje publikácia od Španielskeho národného inštitútu pre kybernetickú bezpečnosť [84]. Správa sa touto tematikou zaoberá z viacerých pohľadov. Skúmané boli napr. typy súborov v inštalačných balíčkoch *.apk*, kde veľa malvéru v sebe obsahuje ďalšie inštalačné súbory, čo môže viesť k nenápadnému šíreniu malvéru. Tiež konštatujú pretrvávajúce problémy so samopodpísanými (self-signed) certifikátmi, viď. kapitola Nezabezpečená komunikácia nižšie. Ďalšou hrozbou sú knižnice starajúce sa o reklamu - autori zistili, že 8 z 10 najpoužívanejších knižníc obsahuje chyby, resp. vyžaduje nadmerné množstvo povolení, čo môže ľahko viesť k ich zneužitiu. Okrem toho autori konštatujú veľký výskyt aplikácií, ktoré vyžadujú viacej povolení ako by v skutočnosti mali. S týmto úzko súvisí aj veľký počet potenciálne nebezpečných metód - API volaní, ktoré sa v aplikáciách vyskytujú. Zaujímavosťou bolo skúmanie kam sa vlastne malvér aplikácie pripájajú - najviac spojení bolo do Číny a Spojených štátov

amerických. V závere autori konštatujú, že najrozšírenejším typom malvéru sú stále trójske kone a najrozšírenejšou rodinou malvéru je rodina FakeInstaller.

2.2.2 Detekcia malvéru

Ako bolo spomenuté v predošlej časti, mobilný malvér sa vyvíja veľkou rýchlosťou a rôznymi smermi. Toto je asi najväčším problémom pri ich detekcii - väčšinou je totiž založená na analýze príbuzných vzoriek malvéru, čo je len do istej miery efektívny spôsob. Pokiaľ sa však objaví nový druh malvéru, tak trvá pomerne dlho, kým sa odchytia a zanalyzujú nejaké vzorky. Preto je potrebné vymysliet' nové spôsoby ako odhaľovať škodlivé aplikácie bez potreby mať vopred odchytené a zanalyzované vzorky tohto malvéru. Viacero vedeckých tímov navrhlo postupy ako detegovať takéto aplikácie. V krátkosti si predstavíme niekoľko detekčných metód resp. aplikácií, ktoré ich priamo implementujú:

SCanDroid

Jednou z prvých použiteľných aplikácií je SCanDroid - dielo Fuchsa a kol. [87]. Autori sa zamerali na štatistickú kontrolu dátového toku, ktorý prechádza jednotlivými aplikáciami. K tomu analyzuje aj povolenia konkrétnej aplikácie uvedené v manifeste. Na základe znalosti povolení a sledovania dátového toku vie určiť, či daná aplikácia pracuje s dátami ako má alebo nie, a teda, či nie je pre používateľa nebezpečná. Pokiaľ je, tak ju automaticky označí ako potenciálne nebezpečnú a upozorní na ňu používateľa. Okrem toho je možné aplikácie kontrolovať aj offline na Google Play.

Kirin a TaintDroid

Ďalší raný spôsob detekcie nebezpečných aplikácií predstavili Enck a kol. [96]. Vzhľadom na to, že Android sa v tom čase ešte len vyvíjal, sa autori rozhodli využívať systém certifikácie aplikácií pri ich inštalácii. Tento systém bol nazvaný Kirin a na detekciu využíva natívny systém povolení. Pri inštalácii aplikácie vyberie z manifestu zoznam povolení a vyhodnotí túto konfiguráciu podľa stanovených bezpečnostných pravidiel:

- S jedným povolením - tu ide hlavne o povolenie na debugovanie iných aplikácií.
- S kombináciou viacerých povolení - napr. prístup k lokácii zariadenia a k internetu.
- S kombináciou povolenia a intentu - napr. pri výbere preferovanej aplikácie na vykonanie nejakej činnosti.

Tento spôsob detekcie bol v danom období jedným z priekopníkov mobilnej bezpečnosti. V súčasnosti so stále sa vyvíjajúcimi hrozbami je už neefektívny a je potrebné hľadať sofistikovanejšie spôsoby detekcie škodlivých aplikácií.

Nadstavbou Kirinu bol TaintDroid [97]. Jeho podstatou bolo značkovanie používateľom definovaných citlivých dát a sledovanie (na úrovni súborov, metód a premenných) ako s nimi aplikácie nakladajú. Jeho hlavnou výhodou je, že je schopný analýzy aplikácií počas ich behu a pokiaľ sa v správaní aplikácií vyskytla nejaká anomália alebo podozrivá činnosť, bola nahlásená ako nedôveryhodná. Autori experimentálne overili, že v článku popísaný spôsob sledovania značiek je úspešný a nepredstavuje výrazné obmedzenie výpočtových nárokov mobilných zariadení.

Statická analýza aplikácií pomocou strojového učenia

V roku 2010 publikoval Shabtai a kol. krátky článok [88] o detegovaní podozrivých aplikácií statickou analýzou kódu. Základom detekcie boli *.apk* súbory rôznych aplikácií (nástroje a hry). Autori sa zamerali na skúmanie:

- vlastností *.apk* súborov - veľkosť, počet a typy súborov ...
- vlastností *.xml* súborov - elementy, atribúty, reťazce, povolenia ...
- vlastností *.dex* súborov - triedy, metódy ...

Po zadefinovaní zvolených detegovaných vlastností týchto súborov sú vybrané aplikácie analyzované niekoľkými klasifikátormi (rozhodovací strom, naivný Bayes, atď.). Z testov vyplynulo, že pri vhodnom nastavení filtrov môže táto metóda slúžiť ako nástroj na detekciu škodlivých aplikácií.

Zaujímavým riešením je aplikácia Drebin [94]. Staticky analyzuje zdrojové kódy, získané skupiny dát ukladá ako vektory a nakoniec aplikuje analýzu pomocou lineárnych SVM (Support Vector Machines). Zaujímavosťou je, že všetky tieto procesy - dekompilácia *.apk* súborov, extrakcia dát a analýza sa deje na zariadení, resp. v androidovej aplikácii. Autori podrobne popisujú extrahované dáta (použité a potrebné povolenia, podozrivé API volania atď.) a ukladanie dát do binárnych vektorov dĺžky závislej od počtu extrahovaných príznakov. Nakoniec ich SVM klasifikuje podľa vopred natrénovanej charakteristiky. Úspešnosť detekcie malvéru autori deklarujú ako vysokú (skoro 94%) a tiež aj rýchlosť spracovania *.apk* súborov (v priemere do 20 sekúnd). Napriek týmto výsledkom aplikácii skončila technická podpora od autorov a nie je voľne dostupná.

Behaviorálna detekcia

Zaujímavý spôsob detekcie škodlivých aplikácií predstavili Burguera a kol. [89]. Vo svojej práci sa zaoberali detekciou škodlivých aplikácií pomocou sledovania ich správania sa počas bežnej prevádzky napadnutého zariadenia. Na tento účel bola vyvinutá aplikácia Crowdroid, ktorá na zariadení sleduje volania jadra OS (open, access, read atď.) a spracované ich posiela na centrálny server (nejde o osobné údaje). Na serveri sa získané dáta analyzujú, vyhodnocujú a podľa určených charakteristík sa rozdelia na potenciálne nebezpečné a neškodné. V tomto prípade však platí, že čím

viac dát od používateľov systém získa, tým je efektívnejší. Autori deklarujú 100% úspešnosť detekcie vlastného malvéru a minimálne 85%-nú pri vzorkách skutočného malvéru. Otázkou je, ako sa takýto systém vysporiada so stále novšími a zákernejšími spôsobmi maskovania skutočnej funkcionality aplikácií.

V roku 2014 bol predstavený systém Andrubis [90], jeden z najrozsiahlejších výskumov publikovaných v posledných rokoch. Autorom sa podarilo za 2 roky výskumu zozbierať a zanalyzovať vyše 1000000 unikátnych aplikácií. Andrubis čiastočne využíva aj statickú analýzu, ale len kvôli správnemu nastaveniu systému a ako pomoc pri stimulácii aplikácií (stimulácia aplikácií je zjednodušene povedané simulovanie používateľských vstupov v kontrolovanom prostredí). Analýza pozostáva z niekoľkých krokov, ktoré už boli spomenuté pri iných systémoch - stimulácia aplikácií (simulácia používateľského vstupu), sledovanie toku citlivých dát podobne ako TaintDroid, sledovanie používania metód a sledovanie systémových volaní z vonku OS Android. Navyše autori zapojili aj dodatočnú analýzu sieťovej prevádzky, teda s kým/čím aplikácie komunikujú, kam sa pripájajú atď. Výskum ukázal, ktoré nebezpečné povolenia malvér najčastejšie využíva, poukázal na problém so self-signed certifikátmi, či v tej dobe aktuálnu chybu v tzv. Master Key (bolo možné zmeniť obsah aplikácie bez zmeny jej podpisu). Ďalej boli zistené rôzne dovtedy neznáme fakty, napr. spôsoby šifrovania v malvéri, používané porty v sieťovej prevádzke, ako často sa pristupuje k súborom na zariadení a k akým, alebo prvé náznaky tzv. croos-platform malvér, keď sa malvér z OS Windows pokúšal infikovať zariadenie cez Android Debug Bridge (ADB). Aplikácia dostala aj GUI vo forme androidovej aplikácie, ale podobne ako pri Drebine jej skončila technická podpora, hoci je stále dostupná na Google Play.

V roku 2015 bol predstavený CopperDroid [91], systém na detekciu malvéru pomocou rekonštrukcie správania sa aplikácie. CopperDroid vo svojej podstate sleduje a zachytáva všetky volania, resp. komunikáciu medzi procesmi (IPC) alebo komponentami (ICC). Samostatne odchytáva a analyzuje systémové volania. Na analýzu ostatných volaní využíva tzv. "unmarshalling" orákulum. "Unmarshalling" je proces rozloženia týchto volaní na jednotlivé komponenty, napr. kto požiadal o vykonanie akcie, komu je volanie adresované, čo sa má vykonať a ďalšie parametre. Tieto dáta sú uložené v objektoch, ktoré sú vstupom do orákula. To následne iteratívne prechádza všetky tieto dáta až kým nenájde definované primitívne typy, napr. reťazce. Z týchto primitívnych typov vie pomocou stimulácie aplikácie určiť jej správanie (resp. jej častí). Nakoniec bola vytvorená mapa správania sa aplikácií, z ktorej bola vybratá časť pozostávajúca zo škodlivého správania. Autori testovali systém na viacerých databázach malvéru a dosiahli úspešnosť detekcie v priemere 70%. Zaujímavým zistením bolo, že stimulácia nemá v niektorých prípadoch výrazný dopad na úspešnosť detekcie (prístup k sieti, prístup k súborovému systému).

V tomto roku bola predstavená nadstavba systému Andrubis zvaná CuriousDroid [93]. Jej hlavné vylepšenie je v novom spôsobe stimulácie aplikácie pri analýze. Na rozdiel od ostatných prác, kde sa stimulácia vykonáva vo forme náhodných príkazov,

CuriousDroid sa snaží aplikácie stimulovať "inteligentným" spôsobom, teda akoby to robil bežný používateľ. K tomu bolo potrebné zanalyzovať jednotlivé komponenty UI aplikácie a k nim prislúchajúce metódy. Na základe tejto analýzy boli potom vytvorené modely predpokladaného používateľského správania. Pri testoch sa tento prístup osvedčil a bolo spustených viacero dovtedy nepozorovaných aktivít ako napr. sieťová komunikácia, či posielanie SMS správ na prémiové čísla.

Jedným z najnovších článkov venujúcich sa behaviorálnej detekcii malvéru je práca Heusera a kol. [92]. V svojej práci predstavujú proof-of-concept model systému na dlhodobú forenznú analýzu malvéru, zvaný DroidAuditor. Konkrétne sa špecializuje na detekciu privilege escalation útokov. Ide o klient-server riešenie postavené na frameworku Android Security Modules od rovnakého tímu. Tieto moduly sledujú rôzne prvky aplikácie, či už ide o medziprocesovú komunikáciu alebo prístup k systémovým zdrojom. Každý modul zbiera informácie spojené s bezpečnou prevádzkou a ochranou súkromia a hodnotiaci algoritmus určuje, či aplikáciu označí ako podozrivú alebo nie. Autori hodnotia dosiahnuté výsledky ako sľubné, ale je potrebné toto riešenie ešte dopracovať.

Kombinácia statickej a dynamickej analýzy

V mnohých prípadoch mobilného malvéru je problém s detekciou pomocou len statickej alebo dynamickej (behaviorálnej) analýzy. Častým riešením je ich prepojenie, resp. vykonanie najprv statickej a potom dynamickej analýzy. Existuje viacero riešení zameraných na tento spôsob, v krátkosti predstavíme niekoľko z nich.

Jedným z najaktuálnejších článkov je práca Colettu a kol. [95]. Autori sa v nej zameriavajú výhradne na kontrolu tzv. bankových trójskych koní. Vo všeobecnosti ide o kategóriu malvéru zameranú na finančné transakcie. V tomto prípade sa zamerali na detekciu Command & Control koncových bodov, teda bodov, cez ktoré môže byť vedená komunikácia do a zo zariadenia - telefónne čísla, URL adresy a pod. Tieto údaje sú staticky získané z manifestu aplikácie resp. dekompilovaného kódu a následne overené dynamickou analýzou, či cez ne ide nejaká komunikácia - a ak áno, potom aká. Zo zhromaždených dát sa podarilo autorom odhaliť viacero nebezpečných domén a telefónnych čísiel používaných známymi bankovými malvérmi, napr. ZitMo. Na záver autori konštatujú, že práca ma sľubné výsledky, ale je potrebné do tohto systému dodávať aktuálne dáta, lebo je veľmi jednoduché prepísať v kóde telefónne číslo alebo URL adresu a aktualizovať malvér.

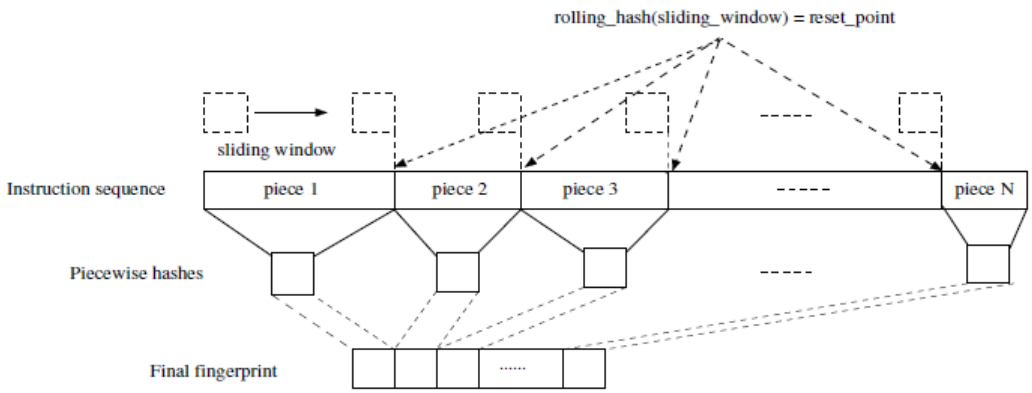
Detekcia repackov

Dôležitou súčasťou ochrany používateľa systému Android je možnosť detekcie repackovaných aplikácií. Ako bolo spomenuté vyššie, tieto aplikácie navonok vyzerajú ako legitímne aplikácie, dokonca ponúkajú tú istú funkcionalitu, ale "obohatenú" o skryté možnosti útočenia pridané útočníkom. Jednu z možností detekcie predstavili Zhou a kol.

v článku z r. 2012 [75]. Predstavená metóda (DroidMOSS) pracuje v troch krokoch:

- Extrakcia príznakov - inštrukcie aplikácie získané disasemblovaním súboru *classes.dex* a získanie informácií o autorovi (meno, kontakt, odtlačok verejného kľúča) z adresára *META-INF*.
- Vygenerovanie a priradenie odtlačku - na získaný kód sa aplikuje metóda *fuzzy hashing*. Táto metóda redukuje veľkosť odtlačkov a výrazne urýchľuje hľadanie drobných zmien v kóde dvoch veľmi podobných aplikácií, viď obr. 2.3.
- Vyhodnotenie podobnosti - v poslednom kroku sa aplikácie rozdelia na získané z oficiálnych marketov a na tie z neoficiálnych. Na základe porovnávania odtlačkov aplikácií sa určuje miera podobnosti dvoch aplikácií. Pokiaľ je táto miera prekročená a sú podpísané rôznymi certifikátmi, tá z neoficiálneho marketu je označená ako repackovaná.

Podľa prezentovaných výsledkov je táto metóda pomerne úspešná v detekcii repackov, má len 10 % mieru nesprávneho určenia (false negative rate). Postup je pomerne nový a inovatívny a jeho úspešnosť sa dá vylepšovať pridávaním nových filtrov a pravidiel.

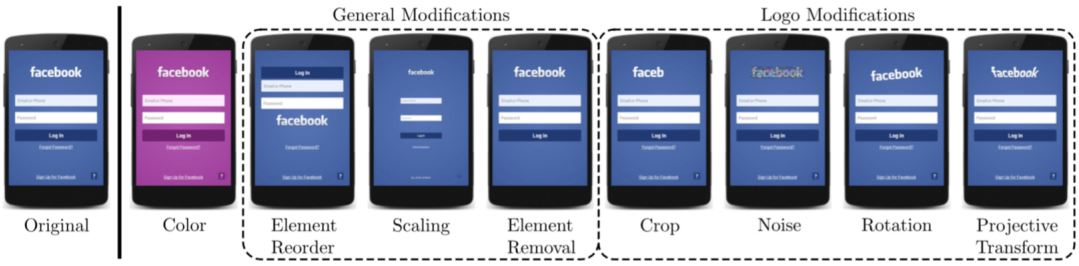


Obr. 2.3: Ukážka fuzzy hashingu [75]

Vizuálna detekcia repackov

Veľmi zaujímavý návrh na detekciu repackov predstavili autori v článku [77]. Častokrát sa stáva, že repacknuté aplikácie nie sú 100-percentným klonom pôvodnej aplikácie - ani funkčným ani vizuálnym. Preto sa autori rozhodli o vizuálnu detekciu takýchto repackov. Vo svojej štúdii sa autori zamerali na analýzu screenshotov prihlasovacích obrazoviek niektorých známych aplikácii. Tieto obrazovky autori rôznym spôsobom pomenili - zmenili farbu pozadia, loga, veľkosť textu, prehodili pozície loga a prihlasovacích formulárov, viď obrázok 2.4. Tieto zmenené obrázky potom dali na testovanie používateľom, ktorí mali povedať, či by do tejto aplikácie zadali

svoje prihlasovacie údaje alebo nie. Výsledky práce ukazujú, že používatelia nevedia rozhodnúť o škodlivosti aplikácie, ak boli premiestnené elementy na obrazovke - zmenené pozície loga, formulárov atď. Naopak, zmena farby pozadia či operácie s logom sú pomerne jasne rozpoznateľné.



Obr. 2.4: Ukážka vizuálnych zmien. [77]

S vizuálnou detekciou repackov úzko súvisí aj možnosť útokov pomocou falošného používateľského rozhrania (user interface - UI). Tejto tematike sa venuje práca Fernandesa a kol. [78]. Autori tu popisujú niekoľko variantov tohto útoku, od klasického phishingu, cez activity hijacking po tzv. clickjacking. K väčšine týchto útokov dochádza kvôli chybám pri implementácii aplikácie, resp. využívajú sa chyby Android UI. Je totiž možné, aby bolo legitímne okno aplikácie napr. na zaplatenie platobnou kartou prekryté oknom z malvéru, do ktorého používateľ zadá tieto údaje. Malvér ich prijme, odošle, vypíše chybu, skončí a odhalí UI legitímnej aplikácie. Používateľ to považuje za chybu a zadá znova údaje do legitímnej aplikácie a nič si nevšimne. Týmto útokom sa dá pomerne dobre predchádzať pomocou navrhnutého riešenia, kde autori kontrolujú medziprocesovú komunikáciu zraniteľných aplikácií a UI, či UI tejto aplikácie nie je náhodou prekryté UI inej - škodlivej - aplikácie.

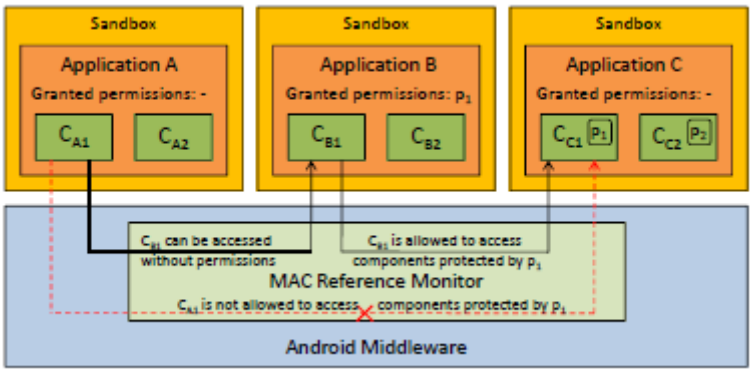
2.2.3 Systém povolení a útoky typu privilege escalation

Najväčším problémom systému povolení v Androide je fakt, že tvorcovia aplikácií dávajú svojim aplikáciám aj množstvo povolení, ktoré v skutočnosti na svoj chod vôbec nepotrebujú. Väčšina aplikácií ako napr. rôzne kalkulačky, kancelárske aplikácie a utility na správu zariadenia nepotrebujú prístup na Internet, ale aplikácia toto povolenia vyžaduje, lebo sa pomocou neho sťahujú reklamy, z ktorých autori primárne získavajú peniaze na ďalší vývoj. Väčšina používateľov tieto reklamy toleruje, ale použité knižnice môžu predstavovať bezpečnostné riziko [101]. Pokiaľ má aplikácia pridelených viac povolení ako reálne potrebuje môže byť zneužitá na útok tzv. privilege escalation.

Felt a kol. vo svojej práci [102] skúmali výskyt nadmerného množstva povolení v

aplikáciách. Autori sa každej metóde v API snažili priradiť povolenie ak bolo nutné na jej vykonanie. Autori použili vlastný nástroj na analýzu dekompilovaných *.dex* súborov - API volaní, intentov a content providers. Z tejto analýzy vytvorili mapu povolení pre testované aplikácie. Oproti dostupnej dokumentácií o systéme povolení našli autori 15-krát viac metód, kde sa kontroluje, či má aplikácia potrebné povolenie. Celkovo sa dá predpokladať, že zhruba tretina dostupných aplikácií má pridelených viac povolení ako potrebuje. Toto číslo bude pravdepodobne rásť pokiaľ sa nevytvorí systém ako vývojárov donútiť k väčšiemu dôrazu na kontrolu svojich aplikácií.

Ako bolo spomenuté vyššie, aplikácie s veľkým počtom pridelených povolení môžu slúžiť ako prostriedok na privilege escalation útoky. Podstatou tohto útoku je, že takáto aplikácia vykonáva API volania ako delegát inej aplikácie, ktorá takéto povolenia nemá [109]. Takýmto spôsobom sa dá kompletne obísť systém povolení, na ktorom je postavená bezpečnosť OS Android, viď obr 2.5. Pridelením povolenia aplikácii si táto vytvorí interface a odhalí ho pre ostatné aplikácie. Škodlivá aplikácia sa môže na tento interface dopytovať a spustiť žiadané volanie API, na ktoré nemá oprávnenie.



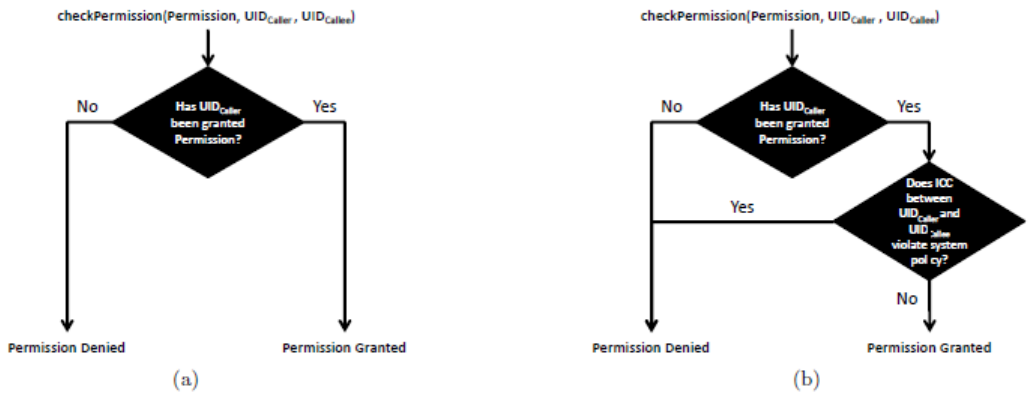
Obr. 2.5: Ukážka privilege escalation útoku, varianta confused deputy [109]

Jeden z prvých návrhov ako týmto útokom predchádzať bol predstavený v [110]. Tento bezpečnostný framework bol nazvaný XManDroid a autori ho testovali napr. aj ako komplementárnu časť k detektorom malvéru Kirin a TaintDroid. XManDroid vykonáva monitorovanie systému, vrátane všetkých inštalovaných aplikácií a komunikačných kanálov medzi nimi. Volá sa vždy, keď sa objaví tzv. Inter Component Communication (ICC) volanie (intent). Toto volanie sa vyhodnocuje a určuje, či je možné ho zneužiť na privilege escalation útok alebo nie (viď obr. 2.6). Pokiaľ ho vyhodnotí ako potenciálne nebezpečné, tak ho zablokuje. Všetky svoje rozhodnutia si uchováva v databáze a sú použité vždy, keď sa takéto komunikácie objavia.

Framework XManDroid je zložený z troch hlavných častí:

- Runtime monitor - základná časť obsahujúca komponenty na monitorovanie a vyhodnocovanie ICC volaní.
- Vlastný inštalátor aplikácií spolupracujúci s monitorovacou časťou.
- Inštalátor systémovej politiky, ktorým sa do systému zavádzajú nové rozhodovacie pravidlá.

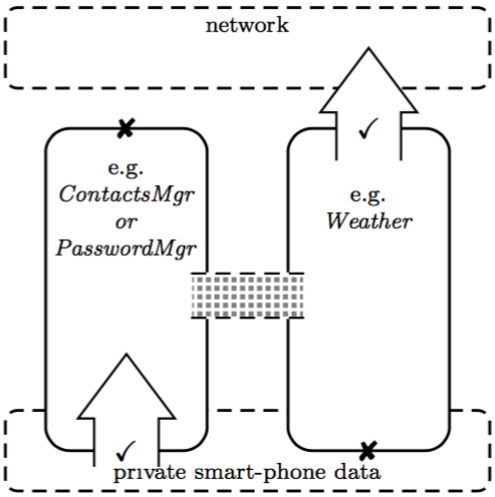
Úspešnosť detekcie nebezpečnej komunikácie závisí len na používateľom (výrobcom) zadefinovaných pravidlách a bezpečnostnej politike. Toto však nie je jednoduchá úloha ani pre odborníkov a bude ešte chvíľu trvať, kým bude dostatočne schopná zachytávať všetky pokusy o útoky. Na reprezentáciu systému - aplikácií a komunikačných kanálov medzi nimi autori používajú grafy, konkrétne knižnicu *JGraphT*. Pri testovaní výkonu aplikácie na mobilnom zariadení bolo zistené, že aplikáciu je nutné optimalizovať, lebo počiatočné napĺňanie databázy rozhodnutí je časovo náročné a každým novým záznamom obmedzuje používateľa v práci so zariadením. Taktiež autori zaznamenali zaujímavé výsledky pri testovaní aplikácií - nebezpečné aplikácie, resp. ich komunikáciu bol systém schopný odhaliť. Problémom bol však veľký počet nesprávne určených hrozieb pri testoch neškodných aplikácií, čo však môže byť odstránené vylepšením pravidiel bezpečnostnej politiky.



Obr. 2.6: Ukážka kontroly prístupu k zdrojom na základe povolení: a) OS Android b) OS Android s frameworkom XManDroid [110]

O rok neskôr Bugiel a kol. rozšírili framework XManDroid v ďalšom článku [111]. Systém bol rozšírený o modul na úrovni jadra (TOMOYO Linux na prepojenie jadra a middlewaru), kvôli lepšej práci so systémovými volaniami na úrovni jadra. Doplnené a vylepšené boli aj pravidlá detekcie a rozhodovania o potenciálne nebezpečných intentoch. Okrem toho autori vylepšili systém monitorovania ICC, ktoré je teraz oveľa efektívnejšie a nespôsobuje tak výrazné spomalenie systému ako predošlá verzia frameworku.

Predošlé články sú viac zamerané na tzv. confused deputy variant privilege escalation útokov. Marforio a kol. sa vo svojej práci [98] venujú druhej variante, tzv. collusion útokom, viď obr. 2.7. V článku sú prehľadne popísané rôzne spôsoby ako sa dá takýto útok vykonať. Autori tiež predstavili niekoľko spôsobov ako týmto útokom zabrániť. Jedná sa hlavne o implementačné vylepšenia a odporúčania pre vývojárov aplikácií. Treba spomenúť, že väčšina z predstavených útokov už bola obmedzená bezpečnostnými aktualizáciami OS Android v novších verziách, viď predošlá kapitola.



Obr. 2.7: Ukážka privilege escalation útoku, varianta collusion [98]

Ďalší výskum od Felt a kol. [103] ukazuje, že veľkým problémom systému povolení ako celku je jeho zložitosť v kontexte bežného používateľa. V prieskume, ktorý autori vykonali je vidieť, že:

- Len 42 % používateľov si je vedomých aké povolenia inštalovaná aplikácia potrebuje, a vie na čo asi tie povolenia slúžia.
- Skoro 70% používateľov sa riadi pri inštalácii aplikácie recenziami od iných používateľov.
- V priemere len 30% používateľov vie správne povedať, aké konkrétne operácie môže aplikácia s danými povoleniami vykonávať.

Na základe týchto výsledkov môžeme povedať, že spoločnosť ešte stále nie je dostatočne vzdelaná v tejto oblasti. Napriek tomuto faktu veľká časť respondentov bola schopná bez vážnych problémov zodpovedať položené otázky.

Na tento výskum nepriamo nadviazali Wei a kol. [104]. Vo svojej práci skúmali evolúciu povolení v ekosystéme OS Android, verziu po verzii. Autori dospeli k záveru,

že povolenia sa menia hlavne z dôvodu pridania nových hardvérových prvkov, alebo kvôli úprave možností OS Android. Z tohoto dôvodu boli pridané povolenia pre prístup k NFC a pod. Ďalej sa povolenia pridávajú (rozširujú) alebo odoberajú na základe bezpečnostnej analýzy doterajšieho stavu OS, napr. zavedenie povolenia pre prácu s prémiovými SMS správami. Záverom však autori konštatujú, že aplikácie obsahujú viac povolení ako by reálne na svoj bezproblémový chod mali. Toto je spôsobené buď lenivosťou vývojárov (kopírovanie manifestu), alebo maskovaním sťahovania citlivých dát "vylepšovaním služieb" a pod. Tieto problémy sa postupom času a vývojom Androidu v novších verziách darí limitovať [48], ale stále je čo zlepšovať.

Apex

Jedným z hlavných problémov systému povolení je nutnosť prijatia všetkých povolení, ktoré si aplikácia vypýta, inak ju nie je možné nainštalovať. Nauman a kol. v r. 2010 predstavili systém Apex [99], ktorý je rozšírením súčasného modelu povolení. Jeho základom je vylepšený inštalátor aplikácií nazvaný Poly, ktorý dovoľuje používateľom upresniť obmedzenia pre každé požadované povolenie:

- Štandardný prístup prijatia všetkých povolení.
- Selektívne zamietnutie vybraných povolení, napr. aplikácia nebude schopná posielať SMS správy.
- Možnosť podmieneného povolenia - aplikácia bude schopná využívať takéto povolenie len určený počet krát za deň, resp. len v určitých časoch.

Tento prístup bol pomerne inovatívny a doteraz neexistuje žiadna jeho praktická implementácia do OS Android. Týmto spôsobom by mohlo byť častokrát zabránené rôznym útokom od aplikácií vyžadujúcich na svoj chod veľké množstvo povolení.

2.2.4 Nezabezpečená komunikácia

Jedným z najväčších problémov androidových aplikácií je, že pre vývoj aplikácií a na používanie bezpečnostných prvkov nie sú pevne stanovené pravidlá čo a ako robiť. Príkladom je nedodržiavanie rôznych odporúčaní (best practices). Niektoré takéto chyby môžu viesť k napadnutiu zariadenia malvérom, resp. odchytávanie komunikácie zariadenia a ďalšieho účastníka.

Nezabezpečenej komunikácii na internete sa vo svojom článku [79] venuje Fahl et.al. Väčšina aplikácií na svoje fungovanie vyžaduje pripojenie na Internet. Skoro všetky tieto aplikácie používajú na zabezpečenie svojej komunikácie protokoly SSL/TLS. SSL a TLS sú kryptografické protokoly slúžiace na ochranu sieťovej komunikácie pred odpočúvaním a manipuláciou. Na ich implementáciu poskytuje Android SDK verzie 4.0 niekoľko balíkov. Povolené je aj modifikovanie (customization) ich použitia, z čoho môžu vyplynúť nasledovné problémy:

- Dôvera vo všetky certifikáty, bez ohľadu na ich podpis.
- Povolenie všetkých hostnames - absencia kontroly pre koho bol certifikát vydaný.
- Dôvera v príliš veľa certifikačných autorít - verzia 4.0 štandardne dôveruje 134 certifikačným autoritám, ďalšie si môže používateľ doplniť alebo odstrániť.
- Zmiešaný mód/žiadne SSL - vývojári majú možnosť kombinovať zabezpečené aj nezabezpečené možnosti spojenia.

Za predpokladu, že je SSL/TLS správne implementované, je odolné voči aktívnemu aj pasívnemu Man-in-the-Middle (MITM) útoku. V dôsledku absencie vizuálnej kontroly a nesprávnej implementácie môžu byť tieto protokoly zneužité na vykonanie tohto útoku. Autori v článku prezentujú aplikáciu MalloDroid a pomocou nej testujú 13500 androidových aplikácií. MalloDroid vykonáva statickú analýzu týchto aplikácií a kontroluje napr. potrebné povolenia, sieťové API volania, použité certifikáty atď. Počiatočná analýza ukázala, že až 43% skúmaných aplikácií nepoužíva bezpečné spojenie k Internetu. Navyše zo všetkých volaní na top 50 serverov len 22% využíva bezpečné pripojenie. Okrem toho takmer 9% použitých certifikátov nebolo overiteľných, podpísaných samým sebou (self-signed) alebo expirovaných. Vyše 17% aplikácií používajúcich bezpečné pripojenie nemá korektne implementované, resp. nebezpečne upravené overovanie certifikátov, čo môže viesť k potenciálnemu zneužitiu.

V ďalšom "kole" autori testovali efektivitu MITM útoku na vyššie spomenuté potenciálne nebezpečné aplikácie. S bežne dostupným zariadením a bezdrôtovým access pointom s MITM SSL proxy simulovali klasický MITM útok na 100 vybraných aplikácií. Výsledky boli alarmujúce:

- 21 aplikácií malo dôveru v ľubovoľný certifikát, vrátane bankových aplikácií.
- 20 aplikácií prijalo certifikát vystavený pre inú aplikáciu, vrátane antivírovej aplikácie.
- Niektoré aplikácie umožňujú prepísanie HTTPS na HTTP a tak uvoľniť citlivé informácie.
- Väčšina z týchto aplikácií vizuálne neupozorňuje na možný útok.

Celkovo sa autorom podarilo týmto útokom získať prihlasovacie mená a heslá, PIN kódy, čísla účtov a podobné citlivé informácie. Navyše po vykonaní analýzy online dotazníka vyplynulo, že ani používatelia smartfónov z IT oblasti nevedia presne určiť, či a kedy sú pomocou svojho zariadenia bezpečne pripojení. Z článku vyplývajú nasledovné možnosti ochrany:

- vynútená kontrola certifikátov
- vynútenie používania HTTPS

- nutnosť zlepšiť systém povolení
- kontrola aplikácií pred inštaláciou - či už vo forme webovej aplikácie alebo priamo po stiahnutí na zariadenie

Podobným smer výskumu zvolili aj Schrittwieser a kol. Vo svojom článku z r. 2012 [80] analyzovali bezpečnosť a možnosti MITM útoku na rôzne komunikačné aplikácie. Tieto aplikácie mali spoločných niekoľko vecí, napr. bezplatnosť, okamžité posielanie správ (instant messaging), prepojenie so zoznamom kontaktov a hlavne pomerne jednoduchý spôsob zabezpečenia pomocou čísla smartfónu. Príkladom takejto aplikácie je WhatsApp využívaný na všetkých platformách okrem Windows Phone. Autori navrhujú päť scenárov ako sa dá na tieto aplikácie zaútočiť:

- Neoprávnené získanie prístupu k účtu - pomerne jednoduchá metóda, lebo používateľ zadáva číslo telefónu manuálne pri inštalácii. Overenie čísla sa deje poslaním SMS na server, ten odošle PIN kód, ktorý používateľ zadá a overenie je kompletné. Komunikácia so serverom môže byť pomerne jednoducho narušená.
- Podvrhnutie inej správy/manipulácia správ - útočník môže posielať správy s falošnou identifikáciou používateľa.
- Nevyžiadané SMS/hovory - generovanie správ na zahltenie používateľa, resp. odchytávanie správ a odpočúvanie hovorov.
- Enumerácia - špeciálny útok zameraný na identifikáciu používateľa pomocou uložených dát na serveri - zoznamu kontaktov, typu zaradenia atd.
- Modifikácia správ o statuse - protokol používaný na zmenu správy nie je bezpečný, navyše sa cez neho útočník môže dostať aj k správam ostatných používateľov, ktorých má obeť v zozname kontaktov.

Na vykonanie útoku autori využili bežne dostupné zariadenia - smartfóny, SSL proxy a HTTPS servery. Vykonané testy viedli k nasledujúcim záverom:

- Všetky aplikácie okrem jednej sú zraniteľné voči MITM útokom.
- Väčšina aplikácií používa XMPP protokol na ochranu proti falošným ID používateľa.
- Všetky testované aplikácie sú zraniteľné voči spamovaniu SMS správami - napadnuté zariadenie je znefunkčnené, lebo stále prijíma správy - vybitie batérie a nemožnosť robiť čokoľvek iné.
- Všetky aplikácie okrem jednej dovoľujú uloženie zoznamu kontaktov na server. Vygenerovaním všetkých kombinácií telefónnych čísel pre danú oblasť, uložením na server a čakaním na odozvu boli autori schopní získať vyše 20000 reálnych čísel zneužiteľných na účely ako spamovanie, nevyžiadaná reklama atď.

Celkovo môžeme skonštatovať, že tieto aplikácie napriek ich rozšíreniu obsahujú značné množstvo zneužiteľných chýb v návrhu alebo celkovej implementácii. Nemusia nutne viesť k finančným škodám, ale narušenie súkromia používateľov je veľmi reálnou hrozbou.

V roku 2011 Chin a kol. publikovali článok [81], v ktorom analyzovali potenciálne zraniteľnosti v komunikácii medzi aplikáciami v rámci jedného zariadenia. Predstavili tu niekoľko spôsobov ako zaútočiť na aplikáciu zneužitím komunikácie cez systém intentov popísaný v časti 2.2.1. Možné útoky rozdelili do dvoch kategórií:

Neoprávnené prijímanie intentov

Pri posielaní implicitných intentov nie je jasné či sa tento intent dostane k očakávanému príjemcovi. Môžu byť veľmi jednoducho odchytávané škodlivými aplikáciami. Patria sem tieto typy útokov:

- Broadcast theft - škodlivá aplikácia môže takýto intent prijať a zablokovať ostatných potenciálnych príjemcov. Navyše môže táto aplikácia spôsobiť Denial of Service (odopretie služieb) pre požadovanú operáciu, resp. sa stať primárnym vykonávateľom, čo môže viesť napr. k úniku dát.
- Activity hijacking - namiesto legitímnej aktivity sa registruje aktivita škodlivej aplikácie a je vykonávaná namiesto očakávaného adresáta, tento útok sa však nemusí vždy podariť (hlavne pri výbere aplikácie pre daný intent).
- Service hijacking - škodlivá služba môže kradnúť dáta a vykonávať rôzne ďalšie útoky.

Podvrhovanie nepravých intentov

Škodlivá aplikácia môže spustiť takýto útok poslaním intentu komponentu, ktorý túto správu nečaká. Vykonaním požadovaného volania sa tento útok môže začať.

- Malicious broadcast injection - za predpokladu, že broadcast receiver dôveruje prichádzajúcemu broadcast intentu, môže vykonať nebezpečnú operáciu. Môže sa tak stať, že sa škodlivá aplikácia dostane k intentom vyhradeným pre systém a tak pôsobiť ďalšiu škodu.
- Malicious activity launch - spúšťaním škodlivých aktivít môže dôjsť k niekoľkým útokom, od otravovania používateľa svojím spúšťaním sa, cez poškodenie dát až po únik citlivých dát.
- Malicious service launch - spustením škodlivej služby sa používateľ vystaví riziku úniku citlivých informácií alebo vykonávaniu nepovolených operácií s jeho dátami.

Z týchto útokov vyplýva, že vývojári by mali pri posielaní citlivých informácií v aplikáciách využívať explicitné intenty. Komponenty by nemali byť exportované (schopné prijímať intenty od iných aplikácií, obsahujú filter intentov, ktoré môžu prijať), pokiaľ sa tomu nedá vyhnúť, nemali by obsahovať vykonateľné akcie kritické pre systém a mala by sa vykonávať kontrola identity odosielateľa intentu.

Na analýzu nebezpečných intentov autori vytvorili aplikáciu ComDroid. Pracuje s disasemblovanými *.dex* súbormi a zaznamenáva potenciálne zraniteľnosti v komponentoch a intentoch. Skúma vytváranie a prenos intentov, týmto spôsobom sa snaží detegovať vyššie spomenuté možnosti útokov. Sleduje sa ich tok, či sú explicitné, či spúšťajú nejakú akciu alebo nesú dáta. Vysiela varovanie ak nájde implicitný intent s povoleniami normal, dangerous alebo žiadnymi, ktorý môže byť zraniteľný útokmi ako broadcast DOS alebo odpočúvané a odchytávané. Okrem toho aplikácia analyzuje komponenty či sú exportované alebo majú filter intentov. ComDroid vyšle varovanie ak nájde komponent chránený povoleniami normal, dangerous alebo nechránený. Taktiež upozorňuje na receivery, ktoré prijímajú systémové broadcast akcie. Nakoniec upozorňuje na implicitné intenty, ktoré by mali byť vysielané ako explicitné.

Pomocou tejto aplikácie boli autori schopní odhaliť viacero zraniteľností v 100 skúmaných aplikáciách. Aplikácia detegovala 401 exportovaných komponentov a 1013 zraniteľných intentov. Napr. 97% aplikácií malo varovanie pred activity hijackingom, no len necelých 28% bolo v skutočnosti zraniteľných. 56% aplikácií bolo zraniteľných voči broadcast injection útoku atď. Analýza ukázala, že väčšina aplikácií je pomerne odolná voči spomenutým typom útokov. Na druhej strane sa našli aplikácie, ktoré boli vo veľkej miere vystavené potenciálnym hrozbám, hlavne v oblasti odpočúvania a úniku citlivých informácií. Vytvorená aplikácia ComDroid navyše môže slúžiť ako pomôcka pre vývojárov na otestovanie zraniteľnosti ich aplikácie.

Ďalším príspevkom do tejto oblasti je práca Ozcana a spol. [105]. Autori v tejto práci predstavili univerzálny šifrovací framework zvaný BabelCrypt. Jeho funkcia spočíva v tom, všetkým nainštalovaným aplikáciám na posielanie správ (messengery) poskytuje možnosť tzv. *end-to-end* šifrovania, bez ohľadu na nainštalované aplikácie. Na používanie tejto funkcionality netreba nič ďalšie inštalovať či konfigurovať, takže s ním môže pracovať aj bežný používateľ. Autorom sa podarilo prepojenie tohto frameworku s najpoužívanejšími messengermi, ako napr. FaceBook Messenger, Skype či WhatsApp. Šifrovanie a derivácia kľúča podľa testov výrazne nezvyšuje záťaž na systémové zdroje, preto môžeme skonštatovať, že tento framework je jedným z vylepšení, ktoré by si mali používatelia OS Android nainštalovať.

Do kategórie nezabezpečenej komunikácie môžeme zaradiť aj tzv. útoky Mobile Evil Twin. Ich skúmaním a detekciou sa zaoberá práca Szongotta a kol. [106]. Vo svojej práci ukázali, ako jednoduché je vykonať útok na nezabezpečenú wi-fi sieť pomocou "naklonovania" príslušného access pointu, resp. ďalších variácií. Autori vo svojej štúdii tieto útoky pomerne jednoducho realizovali a navrhli riešenie tohto problému.

Na zamedzenie takémuto útoku stačí pri pripájaní k wi-fi sieti pridať k overovacím parametrom napr. informácie o aktuálne pripojenej "base station" mobilného operátora či lokáciu tohto access pointu. Úspešnosť závisí od toho, či klient povolí prístup tejto kontrolnej aplikácie k týmto zdrojom - ak poskytne, predstavený systém vykazuje vysokú úspešnosť detekcie tohto typu útoku.

Podobne sem môžeme zaradiť aj ochranu súkromia a možnosti sledovania osôb pomocou mobilných zariadení. Vo svojom článku [107] sa tejto problematike venujú Vanrykel a kol. Autori sa nezaujímajú o zabezpečenie komunikácie v pravom slova zmysle, ale o odchytávanie informácií, ktoré môžu viesť k jednoznačnej identifikácii používateľa. Odhalenia Edwarda Snowdena ukázali, že aplikácie často prenášajú identifikačné údaje o zariadeniach v otvorenej podobe. Medzi tieto údaje patria napr. MAC Adresa zariadenia, IMEI a IMSI čísla, sériové číslo SIM karty či zariadenia. Aplikácie na ich získanie potrebujú povolenie (permission), ale keďže tieto API volania sú v jednom povolení (READ_PHONE_STATE) s bežne potrebnými API volaniami, tak je problém tomu zamedziť. Autori zistili, že jednoduchou analýzou sieťovej prevádzky a kombináciou jej výsledku s TCP časovými pečiatkami je možné veľmi presne určiť identitu používateľa a potom ho sledovať. Vyvodené závery teda potvrdzujú Snowdenove tvrdenia o možnostiach rozsiahleho vládneho špehovania používateľov.

Na záver tejto podkapitoly môžeme spomenúť článok od Dmitrienko a kol. [108]. Autori v článku skúmali bezpečnosť dvojfaktorovej mobilnej autentizácie (M2FA) pre rôzne služby ako napr. Google, Dropbox či Facebook. Autori vo svojom výskume ukazujú, že je pomerne jednoduché M2FA obísť, resp. zrušiť a dostať sa tak k citlivým údajom. Jednoznačným záverom autorov je, že tento model v softvérovej podobe je veľmi zraniteľný voči rôznemu malvéru, či už mobilnému alebo klasickému desktopovému. Odporúčajú preto využívanie hardvérových prvkov na zvýšenie bezpečnosti. Jedná sa hlavne o dedikované hardvérové tokeny, bezpečné elementy a hardvér (napr. na čítačke SIM kariet), ale aj o kontrolu integrity zabezpečenej komunikácie medzi mobilným zariadením a službou, na ktorú sa pripája.

2.2.5 Zabezpečnie Google Play a ostatných zdrojov aplikácií

O zabezpečení Google Play (GP) sme podrobnejšie písali v predošlej kapitole. Jeho podstatou bolo na začiatku zdieľanie a sťahovanie aplikácií, hudby, videí a iného obsahu pre OS Android. Ako pre otvorená platformu mal platiť princíp dôvery v aplikácie, t.j. že nebudú obsahovať žiadny škodlivý kód (podobne ako v Linuxovej komunite). Možno to tak aj spočiatku platilo, ale ako sa platforma rozširovala a zvyšovala počet používateľov, postupne sa zvyšoval aj počet škodlivých aplikácií na GP.

Google sa preto rozhodol proti tomuto bojovať, ale situácia stále nie je ideálna. Kvôli absencii schvaľovacieho procesu ako má Apple AppStore, stále sa môže stať, že napriek prvotnej kontrole môže byť zverejnená škodlivá aplikácia. Taktiež nový

systém Bouncer nie je ešte úplne dokonalý. O jeho zraniteľnosti, resp. obíditeľnosti písal Kassner [34] niekoľko týždňov po jeho uvedení do prevádzky. Už pred jeho uvedením do prevádzky sa rôzni výskumní pracovníci zaoberali problémom detekcie škodlivých aplikácií bez potreby schvaľovacieho procesu. Tieto metódy sa netýkajú len GP, ale aj iných (third-party) aplikačných úložísk (markety). V tejto časti si popíšeme niekoľko návrhov.

Jedným z prvých návrhov na bezpečnostnú kontrolu aplikácií na marketoch bol systém AppInspector od Gilberta a kol. [74] z r. 2011. Tento systém podľa autorov eliminuje nutnosť čakania na schválenie aplikácie a uskutoční sa skôr ako je aplikácia nainštalovaná na zariadenie. Navrhnutý systém je postavený na inštalácii aplikácií na virtuálne zariadenia bežiace v cloudovom prostredí. Tu je testovaná a overovaná ich bezpečnosť, navyše vo virtuálnom prostredí môžu zariadenia bežať paralelne, čím sa rapídne zvýši efektivita celého systému. Aby to však fungovalo, treba vyriešiť tri základné otázky:

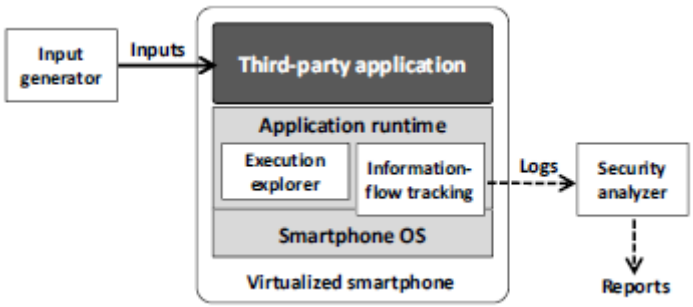
- Ako sledovať a zaznamenávať tok citlivých informácií?
- Ako z nazbieraných dát identifikovať narušenie bezpečnosti resp. súkromia?
- Ako zabezpečiť, že analýza kódu bude dôkladná?

Tieto otázky sú objasnené na nasledujúcom obrázku 2.8. *Generátor vstupov* a *execution explorer* starajúci sa o rôzne možnosti vykonávania kódu pokrývajú tretiu otázku. Beh aplikácie sleduje *information-flow tracking*, čím sa pokrýva prvá otázka a nakoniec *security analyzer* interpretuje výstupné dáta, čím je vyriešená aj druhá otázka.

Pred implementáciou systému autori definovali jednoduché modely bezpečnostného narušenia a narušenia súkromia. Na sledovanie behu aplikácie (informačného toku) použili metódu značkovania tzv. taint popísaný v jednej z ich predošlých prác [97]. Navyše autori zaviedli sledovanie a záznam akcií ako napr. prístupy k GPS, sieti alebo dátovému úložisku. Takéto množstvo záznamov by bolo náročné spracovať v rozumnom čase, preto sa na analýzu vyberajú len záznamy týkajúce sa vopred definovaných citlivých dát alebo operácií.

Na bezpečnostnú analýzu autori navrhujú použiť grafy závislosti (dependency graphs). Tieto grafy ilustrujú cestu od udalosti, ktorá je príčinou napr. zneužitia citlivých dát, cez dátový tok aplikácie až po výstup aplikácie, ktorý bol označený ako bezpečnostná chyba. Na grafoch potom môžeme vykonávať ďalšiu analýzu správania sa príslušnej aplikácie a vytvárať pravidlá použiteľné v ďalšej analýze.

Ďalším prvkom analýzy je sledovanie, či aplikácia využíva všetky povolenia, ktoré má v manifeste deklarované. Toto má slúžiť ako návod pre vývojárov, aby písali bezpečnejšie aplikácie. Najväčším problémom tohto návrhu je, že je problematické generovať všetky možné vstupy aplikácií a následne sledovať príslušné dátové toky. Náhodné generovanie vstupov sa ukázalo ako nedostatočné, autori preto zvolili systematickejší



Obr. 2.8: Schéma AppInspector-a [74]

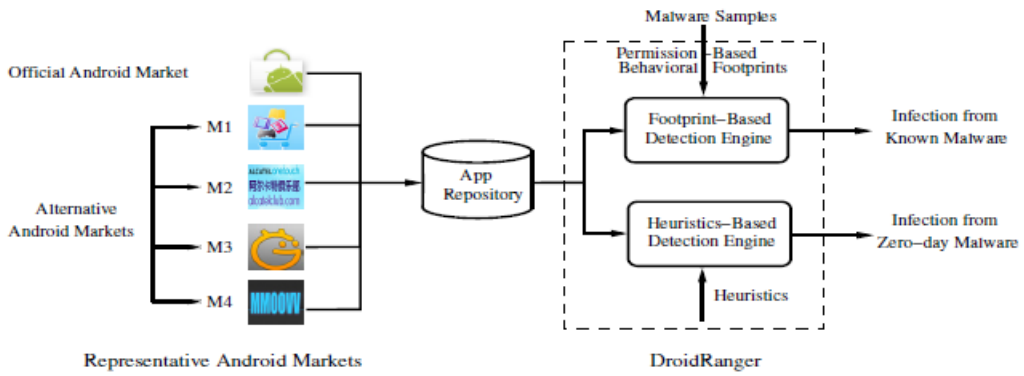
prístup. Využíva sa tzv. concolic vykonávanie, ktoré kombinuje konkrétne (testovanie pomocou náhodných vstupov) a symbolické (systematické prehľadávanie všetkých vetiev kódu) vykonávanie kódu. Správnym nastavením a skombinovaním týchto metód je možné v reálnom čase pokryť veľké množstvo vykonateľných ciest v kóde príslušnej aplikácie. Jeho menšou nevýhodou je nutnosť modifikácie DVM na spustenie aplikácie.

Niektoré prvky z tohto návrhu boli z časti zapracované do kontroly na GP a Bouncera. Väčšina z third-party marketov sa však o takéto opatrenia veľmi nestará, takže bolo nutné vymyslieť spôsob ako kontrolovať aplikácie aj na nich. Veľmi aktívnymi v tomto smere sú výskumníci z North Carolina State University. Vo svojom článku z minulého roku [76] sa zaoberajú detekciou škodlivých aplikácií na oficiálnych aj neoficiálnych marketoch. Autori počas dvoch mesiacov posťahovali všetky bezplatné aplikácie (vyše 200 000) z piatich marketov. Tieto aplikácie následne testovali schémou nazvanou *permission-based behavioral footprinting*, čím boli schopní odhaliť existujúci malvér. Všetky aplikácie, ktoré prešli týmto testom následne išli do filtrovacej schémy nazvanej *heuristics-based filtering scheme*, ktorou sa snažili odhaliť dosiaľ neznámy malvér. Obe schémy boli implementované do systému DroidRanger. Aplikácia funguje nasledovne (ilustrované na obrázku 2.9):

- Zber aplikácií z marketov a ukladanie do lokálneho repozitára.
- Extrakcia údajov o autorovi a potrebných povoleniach, následná organizácia v databáze.
- Prvé testovanie na detekciu známeho malvéru.
- Druhé testovanie na detekciu neznámeho malvéru.

Detekcia známeho malvéru

Na odhaľovanie známeho malvéru museli autori rozdeliť navrhnutú schému na dve časti:



Obr. 2.9: Schéma DroidRanger-a [76]

- filtrovanie na základe povolení
- prideľovanie behaviorálnych odtlačkov

Filtrovanie na základe povolení slúži na vybratie aplikácií, ktoré využívajú len autormi vybrané povolenia (napr. INTERNET, RECEIVE_SMS), resp. ich kombinácie. Týmto spôsobom sa odstráni drvivá väčšina aplikácií (skoro 98%), ktoré nie sú relevantné pre ďalší krok analýzy. Druhým krokom začína manuálnou analýzou správania sa známych rodín malvéru. Týmto spôsobom je možné špecifikovať hľadané metódy pracujúce so žiadanými povoleniami, volania API a analyzovať dátový tok. Navyše autori zapracovali do tohto modelu aj štrukturálnu stavbu aplikácie a takto vytvorili základné typy správania sa malvéru.

Detekcia neznámeho malvéru

Podobne ako predošlá časť, aj táto musela byť rozdelená na dve:

- heuristické filtrovanie
- dynamické monitorovanie činnosti aplikácie

Keďže autori nemali k dispozícii vzorky malvéru na porovnanie, museli použiť na detekciu iné metódy. Zamerali sa teda na použitie dvoch heuristík na sledovanie určitých prvkov OS Android, ktoré by mohli byť potenciálne zneužité. Prvá heuristika súvisí s dynamickým načítaním binárneho Java kódu zo vzdialenej nedôveryhodnej stránky. Pri dynamickom načítavaní kódu z Internetu totiž nie je možné predikovať správanie sa aplikácie - toto sa väčšinou využíva na zobrazovanie rôznych reklám v aplikáciách. Druhá heuristika sa týka dynamického načítavania kódu z lokálneho zdroja. Linuxové jadro stále obsahuje volania priamo viditeľné aplikáciám písaným v natívnom kóde, ktoré tak môžu zneužiť slabiny jadra a rootnuť systém. Okrem toho má Android zvláštne úložisko pre aplikácie v natívnom kóde. Preto je podozrivé, ak sa nejaká aplikácia snaží takýto kód ukryť, resp. inštalovať do iného

adresára. Pomocou nových heuristík boli autori schopní identifikovať nové potenciálne nebezpečné modely správania. Ďalším logickým krokom bolo vyskúšať a popísať ich činnosť a vyhodnotiť z nazbieraných dát ich správanie. Pri vykazovaní podozrivého správania autori vykonali manuálnu kontrolu a pridelili vzorke príslušný odtlačok.

Použitím týchto dvoch testov boli autori medzi stiahnutými aplikáciami schopní odhaliť 10 dovtedy známych rodín malvéru a rôzne ich modifikácie. Celkový čas potrebný na kontrolu vyše 200 000 aplikácií bol 4,5 hodiny. Navrhnutá schéma ma takisto veľkú odolnosť voči false positive vzorkám. Navyše sa pomocou tejto schémy podarilo odhaliť dve rodiny dovtedy neznámeho malvéru - Plankton a DroidKungFu (vzorky boli okamžite poslané vývojárom antivírových aplikácií). Ako autori na záver píšu, tento systém ešte nie je dokonalý - nepokrýva všetky možné varianty chovania sa aplikácií. Poskytuje však potrebný základ, na ktorom sa dá ďalej budovať výskum v tejto oblasti.

2.3 Zhrnutie aktuálneho stavu problematiky

V teoretickej časti dizertačnej práce sme sa venovali popisu operačného systému Android ako celku, komplexnému prehľadu jeho bezpečnostných mechanizmov a bezpečnosti aplikácií pod ním bežiacich.

Z prehľadu aktuálneho stavu problematiky vyplynulo, že výskum v oblasti bezpečnosti OS Android bol sústredený v dvoch hlavných smeroch:

- Bezpečnostné mechanizmy, vrátane kryptografie, postranných kanálov, fyzického prístupu a bezpečnostných rozšírení OS a pod.
- Aplikačná bezpečnosť, vrátane detekcie a prevencie malvéru, zefektívnenie modelu povolení, zavedenie povolení závislých na rolách a iné.

Analýzou problematiky sme zistili, že hlavnými bezpečnostnými problémami boli nedostatočne prepracovaný a neflexibilný systém povolení a široké spektrum malvéru. Náš ďalší výskum preto chceme smerovať práve do oblasti aplikačnej bezpečnosti, v náväznosti na príslušné bezpečnostné mechanizmy. Túto oblasť sme vybrali preto, lebo je dôležitejšie sústredit' sa v prvom rade na všeobecnejšie a praktickejšie otázky, než na príliš špecifické opatrenia závislé na platforme. Preto sme si v rámci dizertačnej skúšky zadefinovali tieto tri rámcové tézy, ktorým sme sa v ďalšej práci venovali:

- **Role-based permission model (Model povolení závislý na rolách).** Základným problémom platformy Android bola nemožnosť výberu povolení, ktoré sú pridelené aplikácii pri inštalácii. Toto často spôsobuje tzv. privilege escalation útoky, pri ktorých dochádza k úniku citlivých dát. Jedným z riešení je systém Apex na dynamické prideľovanie povolení. Systém povolení však môže byť pre používateľa nezrozumiteľný a príliš náročný na konfiguráciu. Naším cieľom bolo

špecifikovať model dynamických povolení, v ktorých používateľ a(alebo) aplikácia vystupuje v nejakej role. Úlohou bolo navrhnúť a implementovať vhodný systém rolí a mechanizmy, ktorými by bolo možné docieliť želané správanie.

- **Malware detection (Detekcia malvéru).** Špecifikovaním a implementovaním modelu rolí by bolo možné lepšie detegovať niektoré typy malvéru a zabrániť niektorým typom útokov. Preto sme sa rozhodli skúmať aj prispôsobenie modelu rolí na detekciu čo najširšieho spektra malvéru.

- **Súvisiace bezpečnostné mechanizmy.** Na zabezpečenie nového systému povolení by bolo potrebné upraviť, resp. zaviesť nové bezpečnostné mechanizmy. Konkrétne sme uvažovali o zásahoch do spôsobu prihlasovania sa do zariadenia, kryptografickej ochrany až po prípadné zásahy do jadra OS.

Vzhľadom na veľmi rýchly vývoj v oblasti bezpečnosti OS Android bolo nutné priebežne meniť ciele dizertačnej práce:

- Hlavným dôvodom boli úpravy v samotnom OS, kde vo verzii 6 došlo k výraznej zmene modelu povolení zo statického na dynamický. Takýto vývoj sme predpokladali nielen my, ale aj veľká časť komunity okolo OS Android. Tým pádom Google vyriešil jeden z dlhotrvajúcich problémov, a s ním aj prvú z našich navrhovaných téz.

- Téze súvisiacich bezpečnostných mechanizmov sme sa venovali v rámci vedenia resp. konzultovania diplomových prác Švandu [60] a Varcholu [61], kde bol vytvorený nový spôsob prihlasovania sa do zariadenia pomocou gesta. Súmárne výsledky sme neskôr publikovali v článku [63], preto považujeme túto tézu za úspešne splnenú, hoci nie ako primárnu časť nášho výskumu.

- Na základe získaných poznatkov sme sa rozhodli preto rozvinúť tézu detekcie malvéru resp. potenciálne nebezpečných aplikácií na základe statickej analýzy požadovaných povolení a analýzy zdrojového kódu aplikácie. Výskum však ukázal, že žiadna metóda na detekciu škodlivých aplikácií nie je stopercentne úspešná. Pri navrhovaní možných vylepšení sme sa inšpirovali predošlým výskumom vykonanom medzi študentami [112]. Rozhodli sme sa zapojiť do procesu detekcie malvéru aj používateľov, keďže detekčné systémy často vyhodnotia škodlivé aplikácie ako bezpečné a naopak. Za cieľ sme si stanovili prepojiť detekčný systém s klientskou aplikáciou, kde si budú môcť používatelia svoje nainštalované aplikácie skontrolovať a pridať aj vlastné hodnotenie, ktoré zasa poslúži nám na zlepšenie fungovania a presnosti detekcie celého systému. Tento systém sme pomenovali "Distribuovaná detekcia malvéru so sociálnymi aspektami", a zvyšok práce sa bude venovať jeho návrhu, technickým detailom a výslednému zhodnoteniu celkovej funkčnosti.

Kapitola 3

Distribuovaná detekcia malvéru so sociálnymi aspektami

Ako sme povedali na konci predošlej kapitoly, existujúce riešenia na detekciu malvéru nie sú stopercentne účinné. Vždy sa nájde nejaká chyba alebo zraniteľnosť v systéme či aplikácii a tvorcovia malvéru ju zneužijú. Ako ukázali testy v rôznych literárnych zdrojoch, veľkým problémom je detekcia nového či "zmutovaného" malvéru. Komerčné riešenia na detekciu malvéru reagujú pomerne flexibilne, ale keďže sú ich algoritmy proprietárne, tak nevieme s istotou povedať, ako vlastne fungujú. V našom výskume sme sa preto rozhodli poskytnúť používateľom jednoduchý a efektívny systém na detekciu mobilného malvéru, ktorého činnosť bude zrozumiteľná a budú ju môcť čiastočne ovplyvniť.

3.1 Pojmy a definície

Predtým než začneme s popisom návrhu nášho riešenia, musíme si zadefinovať pojmy, o ktoré sa v ňom budeme opierať.

- **Malvér** - aktuálne neexistuje presná definícia mobilného malvéru, keďže v každom výskume sa interpretuje trochu inak. Pre naše účely bude malvérom mobilná aplikácia, ktorá:
  - neoprávnene pristupuje k systémovým zdrojom;
  - neoprávnene pristupuje k používateľským dátam, manipuluje s nimi alebo spôsobuje ich únik;
  - vykonáva inú činnosť, ktorá je nežiadúca pre používateľa.
- **Detekcia malvéru** - detekcia vyššie uvedených mobilných aplikácií na základe zvoleného algoritmu (statická, dynamická/behaviorálna atď.)
- **Cieľ detekcie** - cieľom detekcie je úspešne odhaliť škodlivú (resp. podozrivú) aplikáciu nainštalovanú na zariadení.

- **Chyba prvého druhu** - zlyhanie detekcie malvéru - vyhlásenie malvéru za bezpečnú aplikáciu.
- **Chyba druhého druhu** - chybné označenie bezpečnej aplikácie - vyhlásenie bezpečnej aplikácie za malvér.
- **Používateľský vstup** - interakcia používateľa so systémom, napr. používaním klientskej aplikácie na zariadení.

Ďalej si potrebujeme ujasnit' základné požiadavky na celkovú funkcionalitu nášho riešenia:

- **Jednoduchost'** - celé riešenie musí byt' pre používateľa čo najjednoduchšie na inštaláciu a ovládanie. Príliš zložité riešenia odrádzajú používateľov od používania.
- **Zrozumiteľnost'** - riešenie je určené pre laickú verejnost', takže musí byt' jednoduché na ovládanie a jeho funkcionalita zrozumiteľná pre bežného používateľa.
- **Nenáročnost' na zdroje zariadenia** - riešenie nesmie spotrebovávat' veľké množstvo systémových zdrojov zariadenia (batéria, prenesené dáta), lebo by ho používatelia prestali používat'.
- **Spoľahlivost'** - je potrebné, aby bolo riešenie dostatočne spoľahlivé, aby mu používatelia dôverovali.
- **Kompatibilita** - riešenie musí byt' schopné fungovania na čo najširšom spektre zariadení.
- **Minimalizácia poškodenia používateľom** - riešenie by malo byt' odolné voči neprimeranému zaobchádzaniu zo strany používateľa.

3.1.1 Technické príznaky detekcie malvéru

Dôležitou súčast'ou návrhu je identifikácia technických príznakov, ktoré chceme použit' na detekciu škodlivých (podozrivých) aplikácií. Na základe výskumu popísanom v predošlej kapitole sme sa rozhodli pre analýzu povolení, ktoré vyžadujú aplikácie pri inštalácii. Tieto povolenia samy o sebe nemusia byt' škodlivé. Každé z nich však zastrešuje niekoľko (až desiatok) API volaní, ktoré však pri inštalácii vôbec nevidiet'. Tieto API volania znova nemusia byt' samé o sebe škodlivé či nebezpečné. "Vhodnou" kombináciou API volaní z viacerých povolení však môže vzniknút' bezpečnostná diera, ktorú môže zneužit' buď priamo táto aplikácia, alebo nejaká iná.

Detekciu pomocou analýzy povolení doplníme analýzou zdrojového kódu aplikácií, kde okrem vyššie uvedených API volaní môžeme nájst' aj d'alšie metódy (funkcie), ktoré môžu indikovat' neželanú činnost'. Takýmito funkciami môže byt' napríklad pripájanie sa na vzdialený server (tu vieme skontrolovat' aj IP adresu, či nie je v

zozname škodlivých), sťahovanie a nahrávanie rôznych dát atď.

Na tomto mieste však treba znova veľmi dôrazne podotknúť, že pri detekcii malvéru sa nejedná o čisto technický problém. Ako bolo spomenuté v predošlých kapitolách, autori škodlivého kódu sa ho snažia rôznym spôsobom maskovať. Môže ísť o repackované (wrapnuté) aplikácie, kde bol pridaný škodlivý kód do pôvodne neškodnej aplikácie. Škodlivá funkcionalita môže byť maskovaná legitímnou činnosťou a vykonávaná na pozadí. Zdrojové kódy môžu byť obfuskované, či ich analýza je inak znemožnená.

Problémom je, že existuje už tak veľké množstvo aplikácií, že jednotlivec resp. skupina analytikov nie je schopná ich v reálnom čase zanalyzovať. Preto si myslíme, že je potrebné do procesu analýzy aplikácií zapojiť aj ich používateľov. Každý používateľ má v zariadení nainštalované iné aplikácie a inak ich aj používa. Existuje teda predpoklad, že väčšina z nich vie, ako sa majú aplikácie správať a k akým zdrojom (povoleniam) majú mať prístup. Ich názor by teda mohol byť jedným z faktorov, ako vylepšiť úspešnosť detekcie škodlivých (podozrivých) aplikácií.

3.1.2 Kritériá na posúdenie riešenia

Pri testovaní riešenia budeme dbať na splnenie stanovených testovacích kritérií:

- **Jednoduchosť a zrozumiteľnosť pre používateľov** - riešenie, resp. práca s ním, musí byť rýchlo osvojiteľné pre bežného používateľa.
- **Nenáročnosť na zdroje zariadenia** - riešenie nesmie počas svojej činnosti výrazne čerpať zdroje zariadenia. Pri nečinnosti má byť toto čerpanie minimalizované.
- **Rýchlosť vyhodnotenia analýzy** - výsledok analýzy aplikácie musí byť používateľovi zobrazený v dostatočne krátkom čase, aby bol zachovaný používateľský komfort.
- **Spoľahlivosť a kompatibilita** - riešenie musí byť spustiteľné na čo najširšom spektre zariadení a musí byť stabilné.
- **Presnosť** - riešenie musí čo najpresnejšie vyhodnocovať aplikácie na škodlivé (podozrivé) a neškodné.
- **Odolnosť voči chybám prvého druhu** - riešenie musí primárne minimalizovať chyby pri označení malvéru ako bezpečnej aplikácie.
- **Odolnosť voči chybám druhého druhu** - sekundárne, riešenie musí minimalizovať chyby pri označení bezpečnej aplikácie ako malvéru.

3.2 Návrh riešenia

Pri návrhu sme zobrali do úvahy všetky vyššie uvedené požiadavky na riešenie a obmedzenia, a nechali si priestor na dodatočné úpravy riešenia. Rozhodli sme sa pre riešenie postavené na princípe klient-server architektúry. Táto architektúra je dostatočne flexibilná pre naše potreby, pretože vieme zabezpečiť:

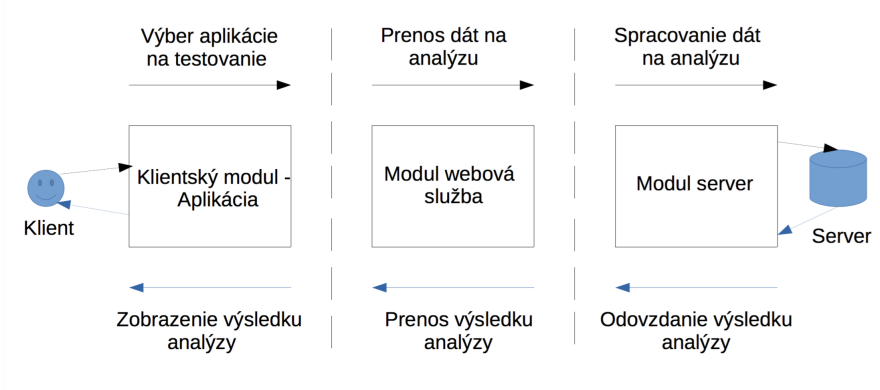
- **Jednoduchosť a zrozumiteľnosť riešenia** - na klientskej strane nám postačuje jednoduchá mobilná aplikácia, ktorá slúži len ako grafické rozhranie pre používateľa a nevykonáva sa na ňom žiadna ďalšia činnosť. Takto vieme zabezpečiť aj **nenáročnosť na zdroje zariadenia** a **kompatibilitu** s čo najväčším počtom mobilných zariadení. Navyše, čím jednoduchšia táto aplikácia bude, tým viac sa zníži riziko **poškodenia systému používateľom**.

- Na strane servera môžeme vykonávať všetku dôležitú činnosť, teda hlavne analýzu aplikácií. Dosiahneme tak **spoľahlivosť** riešenia, keďže nebudeme výraznejšie obmedzení nedostatkom výpočtových zdrojov. Taktiež tu vieme zabezpečiť ukladanie veľkého množstva dát v databáze a ich zálohovanie.

- Klientskú a serverovú časť vieme jednoducho prepojiť pomocou webovej služby, ktorá poskytuje rýchly a zabezpečený prenos dát z jednej strany na druhú a späť. Webové služby sú veľmi flexibilné a vieme si ich podľa potreby nakonfigurovať tak, aby presne vyhovovali našim požiadavkám.

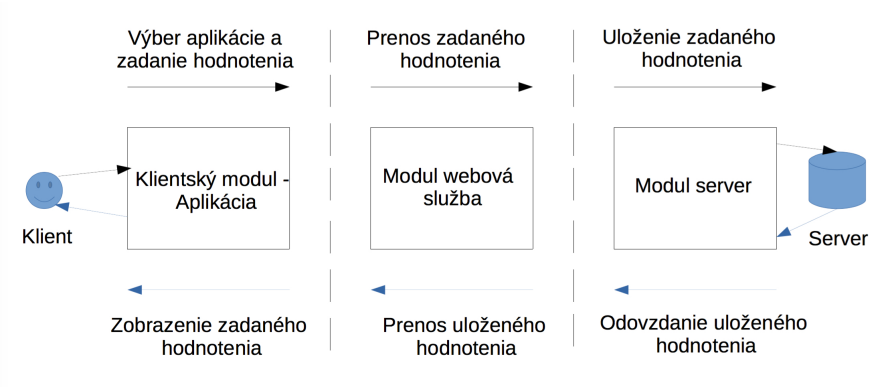
- K serverovej časti vieme ľahko pridať aj administrátorské rozhranie, cez ktoré môžeme celú serverovú časť jednoducho spravovať.

Na obrázku 3.1 sa nachádza ilustračný vysokoúrovňový pohľad na riešenie. Tento pohľad je upresnený v ďalších častiach práce pojednávajúcich o funkcionalite jednotlivých modulov a ich technických detailoch.



Obr. 3.1: Vysokoúrovňová architektúra riešenia

Používateľský vstup do riešenia je načrtnutý na obrázku 3.2. Ako pri predošlom obrázku, aj tento pohľad je upresnený v ďalších častiach práce.



Obr. 3.2: Vysokoúrovňový pohľad na používateľský vstup

Ako je vidieť z týchto diagramov, naše riešenie sa skladá z troch hlavných komponentov - klientskej Android aplikácie, serverovej časti a prepojené sú webovou službou. Okrem toho sme sa rozhodli implementovať aj administrátorské rozhranie vo forme jednoduchej webovej aplikácie pripojenej na serverovú časť.

3.2.1 Modul Klientská aplikácia

Pre klienta - používateľa - najdôležitejšiu časť riešenia tvorí jednoduchá Android aplikácia. Aplikácia nazvaná FEIDroid plní hlavne prezentačnú funkciu, okrem toho však má aj ďalšie využitie načrtnuté v use-case diagrame, viď obr. 3.3. Jednotlivé komponenty si v krátkosti predstavíme a podrobnejšie sa im budeme venovať v implementačnej časti práce.

- Zber informácií o aplikáciách.
- Ich odosielanie na serverovú časť pomocou webovej služby.
- Ich zobrazenie, spolu s výsledkom analýzy používateľovi.
- Možnosť používateľa zadať vlastné bezpečnostné ohodnotenie konkrétnej aplikácie.

Zber informácií o aplikáciách

Na to, aby sme mohli vykonať analýzu aplikácií nainštalovaných na zariadení, potrebujeme vedieť, aké aplikácie na zariadení vôbec sú prítomné. V našej práci predpokladáme, že systémové a vstavané aplikácie od výrobcov sú bezpečné a preto ich neberieme

Obr. 3.3: Use-case diagram aplikácie

do úvahy. Aplikácia na pozadí prehľadá zariadenie a vráti zoznam nainštalovaných aplikácií, ktorý zobrazí používateľovi. Tieto informácie si môže používateľ prezrieť keď klikne na detail aplikácie. Medzi tieto informácie patrí napr. výrobca, verzia aplikácie, SHA-1 odtlačok a iné.

Odosielanie informácií na server pomocou webovej služby

Vyššie spomenuté informácie o aplikáciách sa pri používateľskej požiadavke na analýzu aplikácie odošlú na server. Na základe týchto informácií má byť serverový modul schopný vykonať analýzu zvolenej aplikácie. Informácie sú na server odosielané pomocou zabezpečenej webovej služby, aby sme eliminovali riziko manipulácie s nimi.

Zobrazenie informácií spolu s výsledkom analýzy používateľovi

Po vykonaní analýzy zvolenej aplikácie serverová časť pošle výsledok analýzy naspäť do klientskej aplikácie. Prenos výsledku analýzy je znova vykonaný pomocou webovej služby. Výsledok analýzy je po prijatí zobrazený používateľovi v klientskej aplikácií. Zobrazenie výsledku je veľmi intuitívne a ľahko pochopiteľné pre laického aj skúseného používateľa.

Možnosť používateľa zadať vlastné bezpečnostné ohodnotenie konkrétnej aplikácie

Keďže sme sa rozhodli zapojiť do procesu analýzy aplikácií aj používateľov, musíme im poskytnúť jednoduchý spôsob ako to urobiť. Po zobrazení výsledku analýzy aplikácie sa používateľovi zobrazí možnosť pridať vlastné hodnotenie aplikácie. Zadať hodnotenie je možné vo forme kliknutia na tlačidlo reprezentujúce bezpečnostné hodnotenie aplikácie podľa uváženia používateľa. Zaznamenanie tohto hodnotenia je takmer identické s odoslaním informácií o aplikácii na serverovú časť pri požiadavke na jej analýzu. Odpoveď servera takisto, teda pomocou webovej služby odošle prijaté hodnotenie znova na klientskú aplikáciu, kde sa zobrazí používateľovi. Toto hodnotenie môže používateľ vždy zmeniť, pokiaľ nadobudne dodatočné informácie o aplikácii, či v prípade, že táto aplikácia bude vykazovať neštandardné správanie.

3.2.2 Modul Webová služba

Najjednoduchším a najčastejšie používaným spôsobom prenosu dát v klient-server architektúre je webová služba [115]. Je to softvér na sprostredkovanie komunikácie medzi dvomi strojmi. Dáta sú prenášané v niektorom zo strojovo čitateľných formátov, najčastejšie je to XML alebo JSON. Vybrali sme si formát JSON - JavaScript Object Notation [118]. Ide o spôsob zápisu dát, ktorý je nezávislý na platforme a je primárne určený na prenos dát, ktoré môžu byť organizované v poliach alebo zoskupené v objektoch. V našom riešení webová služba slúži len na prenos informácií medzi mobilným zariadením a serverom. S dátami nijako inak nemanipuluje, jednoducho len prenáša to, čo jej odovzdá buď jedna alebo druhá strana komunikácie. Ako sme spomenuli vyššie, túto komunikáciu sme zabezpečili pred manipuláciou tretej strany počas prenosu dát. Zabezpečenie pozostáva z implementácie šifrovaného spojenia cez HTTPS [117] protokol.

3.2.3 Modul Serverová časť

Na strane servera sa nachádza viacero komponentov vykonávajúcich logickú časť riešenia, t.j. analýzu rizika používania nainštalovaných aplikácií. Podobne ako klientská aplikácia obsahuje serverová časť viacero menších častí vykonávajúcich jednotlivé činnosti. Medzi tieto časti patria:

- Podmodul na sťahovanie *.apk* súborov.
- Podmodul na dekompiláciu *.apk* súborov.
- Databázový podmodul.
- Podmodul vykonávajúci analýzu aplikácií.
- Podmodul administrátorského rozhrania.

Podmodul na sťahovanie *.apk* súborov

Keď serverová časť obdrží dáta aplikácie na analýzu, najprv skontroluje, či už táto aplikácia nebola v minulosti analyzovaná. Pokiaľ už bola, tak len vyhľadá výsledok a pošle ho naspäť klientskej aplikácii. Pokiaľ nebola, je potrebné stiahnuť jej *.apk* inštalačný súbor. Sťahovanie sa vykonáva v tomto module využívajúcom knižnicu tretej strany na stiahnutie konkrétnej aplikácie z GooglePlay úložiska aplikácií. Stiahnutý súbor je následne poslaný do ďalšieho modulu.

Podmodul na dekompiláciu *.apk* súborov

Aby bolo možné analyzovať aplikáciu, je potrebné dostať sa k jej manifestu a zdrojovým súborom. Keďže máme dostupné len inštalačné *.apk* súbory, je potrebné ich najprv dekompilovať. Na túto činnosť slúži práve tento modul. Výsledkom jeho činnosti je zoznam požadovaných povolení z manifestu a čitateľný zdrojový kód aplikácie. Podrobný popis činnosti tohto podmodulu je v ďalšej podkapitole venujúcej sa technickým detailom riešenia. Výstupy podmodulu vstupujú do časti venujúcej sa samotnej analýze aplikácií.

Podmodul vykonávajúci analýzu aplikácií

V najdôležitejšej časti serverového modulu sa vykonáva analýza aplikácií. Do tohto procesu vstupujú dáta z dekompilovaného *.apk* súboru. Samotný proces analýzy sa vykonáva vo viacerých fázach, ktorých detaily si priblížime v ďalšej podkapitole. Ich výstupmi sú normalizované hodnoty, ktoré vstupujú do agregátora. Tu sa z týmto čiastkovým hodnoteniam priradia váhy na základe ich dopadu na celkový výsledok analýzy a vytvorí sa z nich jedno finálne hodnotenie. Toto hodnotenie sa ukladá do databázy a posiela sa webovou službou naspäť do klientskej aplikácie na zobrazenie používateľovi.

Databázový podmodul

Databázový modul uchováva všetky informácie o analyzovaných aplikáciách, t.j. ich názov, vydavateľa, verziu, SHA-1 odtlačok či výsledok analýzy našim algoritmom. Okrem uchovávania týchto dát slúži aj na ich rýchle sprístupnenie v prípade, že bola prijatá požiadavka na analýzu aplikácie, ktorá už bola analyzovaná v inej požiadavke a jej hodnotenie bolo uložené. Okrem týchto dát sú tu uložené aj nastavenia pre čiastkové procesy vykonávané pri analýze, či aktuálne nastavenie váh agregátora čiastkových hodnotení.

Podmodul administrátorského rozhrania

Na zlepšenie a zjednodušenie práce administrátora - na správu serverovej časti riešenia - sme sa rozhodli vytvoriť grafické používateľské rozhranie (GUI). Medzi základné možnosti správy serverovej časti patrí prezeranie záznamov - aplikácií - v databáze, ich

detailov, hodnotení naším algoritmom či používateľmi. Takisto je pomocou neho možné manuálne pridávať *.apk* súbory na analýzu, či vytvárať grafy z výsledkov analýz. Na ukladanie vlastných údajov, ako sú napr. informácie o používateľských účtoch, používa vlastnú databázu. So serverovou časťou riešenia je prepojené na úrovni databázy aj HTTP API.

3.3 Technické detaily riešenia

V implementačnej časti práce si bližšie predstavíme najdôležitejšie komponenty, ktoré sme použili pri vypracovaní nášho riešenia. Jednotlivé podkapitoly sú rozdelené na klientskú aplikáciu a serverovú časť. Časť pojednávajúcu o webovej službe sme z dôvodu väčšej prehľadnosti rozdelili a zakomponovali do zvyšných dvoch podkapitol. Treba podotknúť, že riešenie bolo dlhodobo vyvíjané za pomoci tímov študentov našej fakulty v rámci predmetu Tímový projekt a čiastočne publikované v práci [114]. Riešenie je dostupné cez administrátorské rozhranie serverovej časti, resp. aplikácia FEIDroid je voľne stiahnuteľná z úložiska Google Play.

3.3.1 Modul Klientská aplikácia

Klientská aplikácia je napísaná v programovacom jazyku Java [116], ktorý sa štandardne používa pri vývoji Android aplikácií. Pri vývoji sme použili prostredie Android Studio. V klientskej aplikácii sme nepoužili žiadne neštandardné knižnice, alebo súčasti iných aplikácií tretích strán. V tejto podkapitole práce popíšeme niektoré podstatné implementačné detaily spojené s funkčnými časťami klientskej aplikácie popísanými v návrhovej časti.

Zber informácií o aplikáciách

Táto funkcionalita je implementovaná ako podkomponent nazvaný *updater*. Má na starosti len posielanie a aktualizáciu dát na serveri. Neobsahuje žiadne GUI a funguje len na pozadí. Hlavným komponentom je Android služba (service), ktorá spĺňa tieto požiadavky. Túto službu môžu spustiť rôzne komponenty aplikácie a bude bežať na pozadí, až kým nevykoná svoju úlohu a nevypne sa potrebnou metódou. Do databázy sa ukladajú dáta o všetkých aplikáciách nainštalovaných na zariadení. Dáta sú posielané v dvoch prípadoch:

- Pri nabootovaní systému sa pošlú informácie o všetkých aplikáciách v zariadení.
- V druhom prípade, ktorý nastane pri nainštalovaní novej aplikácie, alebo aktualizácii nejakej aplikácie sa posielajú dáta len o tejto aplikácii.

V tomto prípade sa dosiahne automatická aktualizácia bez potreby zásahu používateľa. K detekcii zmeny v zozname aplikácií sa používa komponent *BroadcastReceiver*. Tento komponent odchytáva špeciálne udalosti v systéme - intenty, ktoré nesú informáciu o vykonávanej akcii. Na základe toho si vieme vyfiltrovať konkrétne aktivity systému.

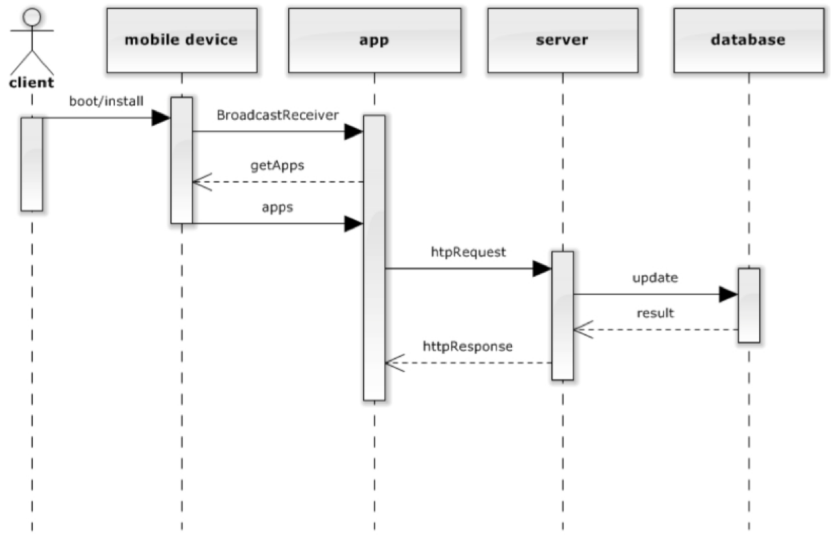
Aby mohla aplikácia korektne vykonávať túto funkcionalitu, je potrebné jej pridať povolenia RECEIVE_BOOT_COMPLETED, INSTALL_PACKAGES a INTERNET. Najdôležitejšie triedy tejto časti aplikácie sú:

- **OnBootReceiver.java** - jedná sa o triedu využívanú ako *BroadcastReceiver*. Má na starosti odchytenie spustenia zariadenia. Po odchytení tejto udalosti sa trieda pomocou metódy *onReceive* pokúsi spustiť službu, ktorá má za úlohu na pozadí aktualizovať dáta na serveri. V prípade úspešnej komunikácie so serverom sa informácie uložia do databázy a následne sú použité pri bezpečnostnej analýze aplikácií.

- **OnInstallReceiver.java** - trieda podobná predošlej, odchytáva inštaláciu, reinštaláciu alebo aktualizáciu ľubovoľnej aplikácie v zariadení. Taktiež spúšťa službu, ktorá aktualizuje dáta na serveri. Tu však musí zo zachyteného intentu zistiť, o ktorú aplikáciu sa jedná, aby nemuseli byť aktualizované informácie o všetkých aplikáciách.

- **ServiceUpdate.java** - tu sa nachádza celková funkcionalita - je to samotná služba bežiaca na pozadí vždy, keď je spustená predošlými triedami a vypne sa po zaslaní dát na server.

Služba získava informácie o aplikáciách pomocou triedy *PackageManager*, ktorá vie získať rozličné informácie o aplikáciách. V našom prípade sú to verzia a názov aplikácie. Keďže neskúmame systémové aplikácie, prehľadávajú sa len aplikácie v adresári */data/app/*, do ktorého sa inštalujú nesystémové aplikácie. Na obr. 3.4 je možné vidieť sekvenčný diagram aktualizácie dát.



Obr. 3.4: Sekvenčný diagram aktualizácie dát

Odosielanie informácií na server

Získané dáta je pred poslaním na server potrebné zorganizovať a uložiť do formátu vhodného na prenos pomocou webovej služby. Najčastejšími formátmi pri prenose dát sa pri webovej službe používajú XML a JSON. V našom riešení sme zvolili formát JSON (JavaScript Object Notation) [118] štruktúr. Ide o spôsob zápisu dát, ktorý je nezávislý na platforme a je primárne určený na prenos dát, ktoré môžu byť organizované v poliach alebo zoskupené v objektoch. Na odosielanie údajov do webovej služby sa využíva štandardná metóda *POST*. Tabuľka 3.1 ilustruje štruktúru odoslaného vzorového JSON objektu na server.

Tabuľka 3.1: Štruktúra JSON objektu

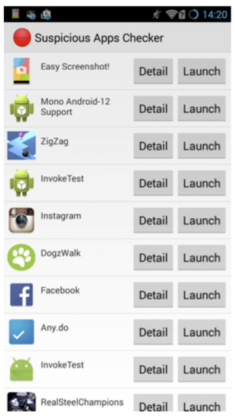
{	
”name”:	”Sample application”
”package”:	”com.example.sampleapplication”
”version”:	”1.0”
”fingerprint”:	”81:51:32:5D:CD:BA:E9:E0:FF:95:F9:F9”
”description”:	”Sample application”
}	

Zobrazenie informácií používateľovi

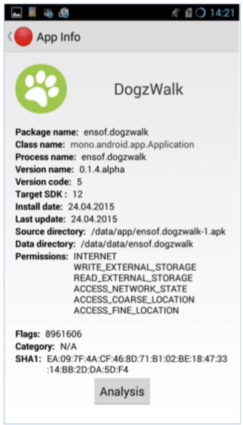
Ďalšou významnou funkciou aplikácie je zobrazenie informácií o nainštalovaných aplikáciách a bezpečnostného rizika aplikácií používateľovi. Po spustení aplikácia zobrazí zoznam nainštalovaných aplikácií. Zo zoznamu si používateľ môže zvoliť zobrazenie detailov aplikácie (tlačidlo *Detail*) alebo spustenie aplikácie (tlačidlo *Launch*), viď obr. 3.5. Po stlačení tlačidla pre detail sa na obrazovke zobrazia dodatočné informácie o aplikácii, viď obr. 3.6. Medzi tieto informácie patria napr. názov balíka, názov procesu, verzie (slovná aj číselný kód), cieľové SDK, dátumy inštalácie, poslednej aktualizácie, zoznam povolení, kategória či SHA-1 odtlačok.

Okrem kategórie sú všetky informácie získané lokálne zo zariadenia. Pre získanie kategórie aplikácie sa vytvára štandardná HTTPS *GET* požiadavka na server. Požiadavka smeruje na adresu $https : // < adresaservera > /api/application/ < idaplikacie > /categories$. V odpovedi sa vráti zoznam kategórií, do ktorých aplikácia patrí. Atribút ID aplikácie sa získa pri prechode zo zoznamu aplikácií na detail aplikácie a to volaním *GET* na adresu $https : // < adresaservera > /api/application/find?name =< nazovaplikacie >$. Údaje sú prijaté znova vo formáte JSON, štruktúra objektu závisí od volanej metódy.

Ďalšou možnosťou používateľa je zobrazenie bezpečnostného rizika, ktoré aplikácia predstavuje. To je možné stlačením tlačidla *Analysis*. Spôsob určenia rizika je popísaný v serverovej časti riešenia. Pre získanie tohto údaju z databázy aplikácia opäť vytvorí



Obr. 3.5: Zoznam nainštalovaných aplikácií

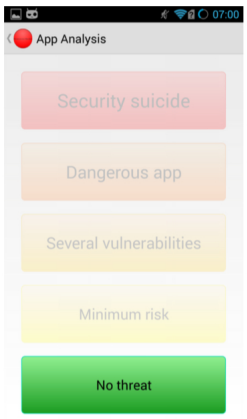


Obr. 3.6: Rozšírené informácie o aplikácii

štandardnú HTTPS *GET* požiadavku na server. V odpovedi sa nachádza percentuálne vyjadrená hodnota rizika, pričom 0 % predstavuje žiadne riziko a 100 % naopak maximálne riziko. Následne podľa získanej hodnoty sa prispôsobí zobrazenie výsledkov - vysvieti sa príslušné tlačidlo, viď obr. 3.7. Po kliknutí na vysvietené tlačidlo sa používateľovi zobrazí popis konkrétneho stupňa ohrozenia. Okrem toho sa zobrazí aj zoznam požadovaných povolení spolu s popisom ich potenciálneho rizika.

Používateľský vstup

Jedným z hlavných cieľov práce bolo umožniť používateľovi nejakým spôsobom ohodnotiť zvolenú aplikáciu z hľadiska bezpečnostného rizika. Náš algoritmus môže totiž aplikáciu vyhodnotiť ako nebezpečnú a tá pritom nebezpečná nebude, alebo naopak. Tento prípad je veľmi častý pri bankových aplikáciách resp. rôznych komunikačných aplikáciách. Tieto aplikácie využívajú pre svoj správny chod veľké množstvo povolení



Obr. 3.7: Zobrazenie výsledku analýzy

a náš algoritmus ich preto môže vyhodnotiť ako rizikové. Používateľ ich však môže považovať za bezpečné, resp. je o ich bezpečnosti presvedčený. Preto pokladáme za vhodné, aby sa naše hodnotenie do istej miery dalo korigovať používateľským hodnotením. Naše hodnotenie pritom poslúži ako "návod" pre používateľov. Navyše týmto spôsobom získame množinu dôveryhodných vývojárov, čo sa znova môže premietnuť do nastavenia hodnotiaceho algoritmu. A nakoniec sú pre nás spätnou väzbou o tom, či je nastavenie parametrov hodnotiaceho algoritmu správne a znova ho vieme upraviť. Implementácia tohto vstupu nie je zložitá, na klientskej strane zahŕňa úpravu používateľského rozhrania a časti webovej služby.

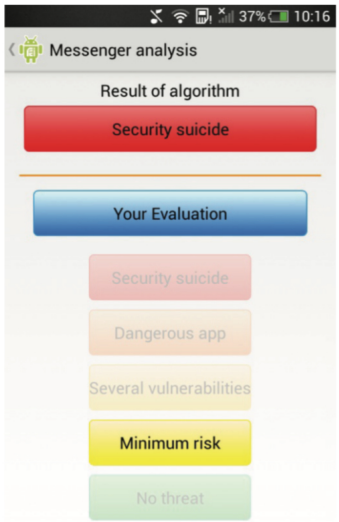
V záujme čo najväčšej prehľadnosti sme sa rozhodli upraviť existujúce rozhranie zobrazujúce výsledok analýzy, viď obr. 3.7. Odstránili sme preto neaktívne tlačidlá, ktoré mali len informatívny charakter a "zaberali miesto" na obrazovke. Pridali sme tlačidlo *Your Evaluation* na zadanie hodnotenia pre používateľa a nakoniec ďalších päť tlačidiel, ktorými môže používateľ zadať vlastné hodnotenie, viď obr. 3.8.
Proces zadávania hodnotenia je veľmi jednoduchý:

- Po kliknutí na tlačidlo *Analysis* sa zobrazí obrazovka s jedným tlačidlom, ktoré predstavuje výsledok analýzy naším algoritmom a s tlačidlami používateľského vstupu, z ktorých je aktívne len tlačidlo s názvom *Your Evaluation*. Ostatné tlačidlá sú neaktívne, viď obr. 3.9 prvé zľava.

- Po kliknutí na toto tlačidlo sa odblokujú ostatné tlačidlá s jednotlivými stupňami hodnotenia, viď obr. 3.9 druhé zľava.

- Po kliknutí na vybrané tlačidlo sa odošle hodnotenie používateľa na server o tejto akcii ho informuje aj hlásenie "FeiDroid is posting Data", viď obr. 3.9 druhé sprava.

- Ak sa teraz používateľ vráti na zoznam aplikácií a znova zobrazí detail tejto apli-



Obr. 3.8: Výsledné grafické rozhranie

kácie, už sa mu zobrazí jeho predošlé hodnotenie, ktoré môže aj zmeniť rovnakým spôsobom, viď obr. 3.9 prvé sprava.

Obr. 3.9: Zadávanie hodnotenia

Pri odoslaní hodnotenia do webovej služby bolo potrebné zistiť, či daná aplikácia už má hodnotenie od konkrétneho používateľa. Vytvorili sme preto ďalší *GET* dopyt, do ktorého sme zahrnuli ako identifikátor ID aplikácie v našej databáze a ID používateľa, resp. zariadenia, kde je táto aplikácia nainštalovaná. ID zariadenia sme zvolili ako anonymný spôsob rozlišovania medzi jednotlivými používateľmi, keďže aj pri viacerých profiloch na jednom zariadení má každý profil toto ID unikátne. Pokiaľ sa v databáze nájde hodnotenie konkrétnej aplikácie od konkrétneho používa-

teľa, je mu zobrazené v GUI. Pokiaľ neexistuje, tak služba čaká na zadanie hodnotenia.

Zadanie hodnotenia začína v triede *AnalysisFragment.java*, kde si uložíme hodnotenie aplikácie od používateľa do *POST* požiadavky. Odtiaľto zavoláme triedu *ServiceUpdate.java*, ktorej pošleme toto hodnotenie spolu s ID hodnotenej aplikácie a spolu s výsledkom analýzy vykonanej na serveri. Triede zadáme, že má pokračovať v *POST* požiadavke pre používateľský vstup. Získame ID zariadenia a zavoláme metódu *execute()* z triedy *PostData.java*. Do tejto metódy pošleme všetky predošlé hodnoty a reťazec $*user_input*, aby metóda vedela, čo má vykonať. Implementácia *POST* požiadavky sa končí v triede *PostData.java*, kde sa všetky hodnoty uložia do JSON formátu, skonvertujú sa do reťazca a odošlú na server pomocou HTTPS *POST* metódy.

Dodatočné úpravy

V skorších verziách riešenia bolo niekoľko drobných nedostatkov, ktoré sme postupne odstraňovali. Neboli to chyby ovplyvňujúce funkcionalitu aplikácie, skôr sme sa snažili o zlepšenie používateľského komfortu. Medzi tieto vylepšenia patria napr.:

- Prechod na Let's encrypt [120] - certifikačnú autoritu, ktorá poskytuje bezplatné X.509 certifikáty pre TLS šifrovanie prostredníctvom automatizovaného procesu. Proces je navrhnutý tak, aby eliminoval súčasný zložitý proces manuálnej tvorby, validácie, podpisovania, inštalácie a obnovy certifikátov pre bezpečné webové stránky a webové služby. Navyše nemusíme uchovávať certifikát lokálne a celková práca s ním je jednoduchšia.

- Oprava kódovania - nedopatrením sme nepoužili štandardné *utf8_general* kódovanie a nesprávne sa preto pracovalo s názvami aplikácií, ktoré obsahovali diakritiku. Okrem toho sme pridali funkciu URL encoding v API, ktoré slúži na vyhľadanie aplikácie podľa názvu, čo vyriešilo nefunkčné dopyty pre aplikácie, ktoré majú v názve medzeru.

- Kontrola internetového pripojenia - k správnemu chodu klientskej aplikácie je potrebné mať pripojenie k internetu, aby mohla komunikovať so serverom a vymieňať si potrebné dáta. V predošlej verzii nebola ošetrená situácia kedy sa prerušilo spojenie a aplikácia tak nečakane skončila. Na elimináciu takýchto situácií sme zapracovali kontrolu internetového pripojenia v každom stave, kedy je potrebné. Ak kontrola vyhodnotí, že je používateľ pripojený, nič sa nestane a môže pokračovať v práci. V opačnom prípade sa zobrazí vyskakovacie okno, ktoré ho upozorní na to, že je potrebné mať internetové pripojenie. Následne vráti používateľa do predošlého stavu, odkiaľ vykonával príslušnú operáciu.

- Emailová notifikácia vývojárov v prípade nedostupného záznamu aplikácie - v prípade, že sa z nejakého dôvodu v databáze nenachádza záznam k aplikácii, ktorú používateľ chcel analyzovať, ho aplikácia len upozornila, že sa jej nepodarilo načítať aplikáciu. Z dôvodu urýchlenia hľadania prípadných chýb a zlepšenia

používateľského prostredia sme zvolili upozornenie vývojárov pomocou emailu spojenú s klientskou aplikáciou prostredníctvom Google Play. Pri upozornení na neúspešné načítanie dát k aplikácii umožníme používateľovi odoslať vopred vyplnený email na túto adresu.

3.3.2 Serverová časť

V tejto časti implementácie si predstavíme najdôležitejšiu časť nášho riešenia - serverovú. V serverovej časti sa vykonávajú všetky operácie súvisiace s analýzou aplikácií, či už ide o dekompiláciu inštalačných súborov, práca s databázou alebo samotná analýza. Väčšina týchto komponentov je znova písaná v jazyku Java.

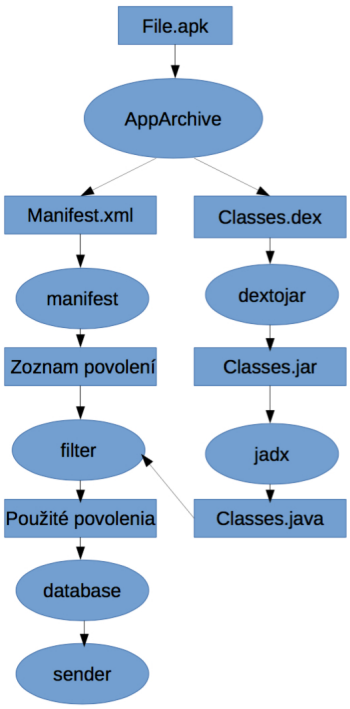
Dekompilácia *.apk* súborov

Android aplikácie sú inštalované z *.apk* súborov, čo sú v podstate *.zip* archívy založené na báze *.jar* súborov. Takýto súbor obsahuje bytecode aplikácie uložený v súboroch spustiteľných v DVM (v *.dex* formáte), zdroje, rozloženie používateľského rozhrania a manifest (*AndroidManifest.xml)*). Manifest je povinný súbor a bez jeho informácií nie je možné aplikáciu nainštalovať ani spustiť. Komponent nazvaný *ReverseApk* umožňuje tento archív rozbaliť a analyzovať meta informácie z manifestu, ako napr. požadované povolenia, služby, aktivity, názov balíka, verziu SDK či podpis. Následne sa v aktuálne dekompilovanom bytecode pokúšame extrahovať úplný zoznam dostupných Java objektov a metód. Tieto informácie sa využívajú pri statickej analýze, počas ktorej sa snažíme odhaliť nevyužívané povolenia, ktoré potenciálne zvyšujú riziko použitia aplikácie. Okrem toho môžu aplikácie žiadajúce nebezpečnú kombináciu povolení svedčiť o nebezpečnom správaní, čo sa využíva v nasledujúcom komponente. Celková štruktúra komponentu je zobrazená na obr. 3.10.

Jednotlivé podkomponenty majú nasledovnú funkciu:

- **APPARCHIVE** je počiatočnou fázou programu, ktorá zabezpečuje prepojenie modulu s ostatnými časťami aplikácie. Webová služba sa dopytuje na dané *.apk*, ktoré ma byť analyzované. Ak už daná aplikácia bola analyzovaná, znovu sa tá istá operácia nevykonáva, avšak ak skript zistil, že dané *.apk* ešte nebolo analyzované, snaží sa ho stiahnuť z externého zdroja. V prípade úspešného stiahnutia v tomto module, dochádza k rozbaleniu *.apk* súboru, získaniu súborov *classes.dex* a čitateľného súboru *AndroidManifest.xml*. Pri implementácii tohto komponentu sme použili už existujúci nástroj Apktool [121]. Pomocou neho sme zabezpečili nielen rozbalenie pôvodného *.apk* súboru, ale aj získanie čitateľného *AndroidManifest.xml* a súborov *classes.dex*, ktoré sú potrebné pri následnom spracovaní.

- **MANIFEST** slúži na spracovanie súboru *AndroidManifest.xml*, ktorý je súčasťou každej aplikácie pre platformu Android. Z manifestu sa získa zoznam povolení, ktoré aplikácia požaduje. Vstupom programu je čitateľný súbor *AndroidManifest.xml*. Zoznam povolení sa získa pomocou vyhľadávania reťazca "an-



Obr. 3.10: Štruktúra komponentu ReverseApk

droid.permission." v texte vstupného súboru *AndroidManifest.xml*. Výstup je zoznam všetkých povolení, ktoré aplikácia v manifeste požaduje.

- **DEXTOJAR** je použitý na prevod *classes.dex* do súboru *classes.jar*. Následne sa *classes.jar* dekompiluje pomocou nástroja *jadx* [122], čím získavame zdrojový kód aplikácie (napísaný v jazyku Java), vhodný na ďalšie spracovanie.

- **FILTER** hľadá povolenia v zdrojovom kóde. Nastavenie filtra je priamo závislé na súbore. Vstupom tohto komponentu je zoznam povolení zo súboru *AndroidManifest.xml* a tiež čitateľné *.java* súbory, ktoré sme získali konverziou a dekompiláciou *classes.dex*. Hlavnou funkciou tohto komponentu je zisťovanie, či sa jednotlivé povolenia získané z manifestu používajú v zdrojovom kóde aplikácie. Keďže sa jednotlivé povolenia v zdrojovom kóde nevolajú priamo menom, ale formou funkcií (metód), bolo potrebné vytvoriť komponent s názvom DATABASE. V tomto komponente sa uchováva, pre všetky povolenia, zoznam funkcií (metód), ktoré môžu byť volané v zdrojovom kóde. Aby sa zistilo, či sa dané povolenie v aplikácii používa, musí sa v jej zdrojovom kóde volať aspoň jedna funkcia (metóda) súvisiaca s daným povolením. Zdrojový kód sa prehľadáva na základe parametrov z komponentu DATABASE.

- **DATABASE** obsahuje zoznam povolení a pre každé povolenie zoznam funkcií

(metód), ktoré pre svoju činnosť potrebuje. Myšlienkou vytvorenia je nutnosť prepojenia jednotlivých modulov pomocou databázy.

- **SENDER** podkomponent, ktorý odosiela, vo forme JSON, použité a aj nepoužité povolenia, ktoré sme získali počas statickej analýzy.

Odosielanie informácií klientskej aplikácii

Na vkladanie dát zo servera a komunikáciu medzi klientskou aplikáciou a serverom používame webovú službu postavenú na Jersey [123] frameworku v Jave, ktorý je spustený na aplikačnom serveri Apache Tomcat 7 [124]. Na objektovo-relačné mapovanie databázy sme zvolili Java Persistent API (JPA) [125] technológiu implementovanú v EclipseLink [126]. Komunikácia s webovou službou prebieha cez HTTPS a dáta sa serializujú do JSON formátu. Samotnú konverziu objektov do JSON formátu vykonáva Jackson Marshaller [127]. Štruktúra služby sa skladá z viacerých podkomponentov, ktoré sú popísané v nasledujúcej časti:

- **Entity** sú dátové štruktúry zodpovedajúce entitám v databáze Triedy, ktoré sú mapované pomocou JPA sú **Application**, **Permission**, **PermissionUsage** a **ApplicationCategory**. Inštancie týchto tried dostaneme pomocou *EntityManagera* (urobíme dopyt na databázu). Využívame ich na prácu s dátami pri výpočtoch. Nepoužívame ich na prezentáciu a prenos cez HTTPS. Na tento účel slúži vrstva *Resource*.

- **Resource** - na prenos dát cez webovú službu ku klientovi používame *Resource* triedy, ktoré sa následne serializujú do JSON formátu. Pre všetky entity existuje aspoň jedna *Resource* trieda. Môžeme však definovať ľubovoľne veľa *Resource* tried, ak potrebujeme dáta prezentovať rôznymi spôsobmi. Tieto triedy používajú anotáciu @XmlRootElement(name="názov resource") prípadne iné XML anotácie podľa potreby. Takto definované triedy vie Jackson Marshaller automaticky serializovať/deserializovať do/z JSON formátu pri prenose. Na konverziu medzi entitami a *Resource* triedami používame službu pre jednotlivé entity.

- **Service** - celá webová služba je rozdelená na viacero častí, podľa hlavnej entity s ktorou práve pracujeme. Toto rozdelenie je pre koncového používateľa skryté. Máme zvlášť službu pre aplikácie, povolenia, využitie povolení, kategórie, ktoré môžu navzájom využívať svoje služby, ak je to potrebné. Štruktúra jednotlivých služieb je podobná, pretože všetky dedia od abstraktnej generickej služby *BasicService*. Každá služba má určenú hlavnú entitu, nad ktorou pracuje a hlavnú *Resource* triedu, na ktorú bude konvertovať entitu pred prenosom. *BasicService* má implementované niektoré metódy, ktoré sú rovnaké pre všetky rozšírenia s jediným rozdielom, ktorým je entita nad ktorou pracuje. Každá entita musí mať definované tzv. *Named Queries* pomocou @NamedQueries anotácií. Názvy týchto dopytov musia byť v tvare <nazov_triedy>.<nazov_dopytu>. Každá trieda, ktorá má svoju službu, musí mať aspoň tieto dopyty: *findAll*, *findById*, *findByIds*.

- **Basic Service** - ukážkou metódy z triedy *BasicService* je napr. **public Entity findById(Long id)**, ktorá vyhľadá v databáze inštanciu s ID podľa parametra. Metódy dostupné ako API webovej služby (metódy majú príponu "Resource" kvôli odlíšeniu od predchádzajúcich metód a zdôraznenie, že pracujú s resourceami a nie entitami), napr. **public Response findAllResource()** vráti všetky záznamy entity v databáze. Metódy, ktoré sú prístupné na komunikáciu, majú v kóde nastavenú cestu, pomocou ktorej sa k metóde pristupuje a metódu HTTP protokolu (*GET, POST, PUT, DELETE)*. K metódam služby pristupujeme cez URL v tvare: <adresa_servera>/<nazov_modulu>/api/<service>/<metoda+parametre>.

- **ApplicationService** - je rozšírením *BasicService*, ktoré operuje nad aplikáciami a obsluhuje analýzu, napr. **public Application findMatchingApplication** vráti aplikáciu z databázy, ktorá sa zhoduje podľa názvu a verzie s aplikáciou v parametri. Webová služba obsahuje ešte samostatné služby *ApplicationCategoryService*, *PermissionService* a *PermissionUsageService*. Tieto rozšírenia *BasicService* obsahujú iba implementácie konverzných metód.

- **Criteria** - ako bolo spomenuté pri opise metód služieb, vyhľadávanie entít v databáze môže byť filtrované pomocou kritérií. Na tento účel slúži generické rozhranie *Criteria*. Triedy, ktoré metódy využívajú, sú definované v *javax.persistence* balíku a tvoria súčasť JPA technológie. Metóda *applyTo* je zodpovedná za vykonanie filtrácie nad entitou. Implementácia tohto rozhrania by mala obsahovať iba špecifické kritérium a každé ďalšie by malo mať samostatnú implementáciu.

- **Metamodel** - na uľahčenie tvorby kritérií s využitím *CriteriaBuildera* z JPA frameworku a na lepšiu prehľadnosť kódu využívame tzv. metamodel, ktorý mapuje atribúty entít na ich dátové štruktúry. Pomocou tried metamodelu vieme ľahko pracovať s atribútami do *CriteriaBuildera*. Metamodel je generovaný pri builde projektu pomocou EclipseLink JPA Metamodel generátora.

- **Maven Build** - na vytvorenie WAR súboru, ktorý vieme nahrať na aplikačný server používame Maven [128]. Jedná sa o robustný nástroj na tvorbu (buildovanie) predovšetkým Java projektov. Stará sa zároveň o sťahovanie závislostí na iných Java balíkoch a umožňuje vykonávať rôzne podprocesy počas tvorby alebo iného procesu nad projektom. Celé nastavenie projektu, doplnkov (pluginov) a procesov je zhrnuté v súbore **pom.xml**. Týmto sa zároveň uľahčuje importovanie projektu do iného vývojového prostredia. Stačí pristúpiť k tomuto súboru a automaticky sa vytvorí celá projektová štruktúra. Náš projekt je nastavený tak, aby sa pri tvorbe (okrem štandardných operácií) automaticky vygeneroval spomínaný metamodel.

Konfiguračná služba

Ako rozhranie na konfiguráciu webovej služby používame samostatnú službu *AppConfigService*, pomocou ktorej vieme z databázy zisťovať potrebné nastavenia a meniť

správanie webovej služby bez priameho zásahu do kódu. Na ukladanie konfigurácie používame tabuľku 3.2. Na mapovanie tejto tabuľky na Java entitu používame JPA. *AppConfigService* je rozšírením *BasicService*, takže máme prístup k jeho metódam. Táto služba však nie je prístupná cez HTTP požiadavky. Modifikácie konfigurácií sa dajú robiť iba v kóde alebo priamo v databáze. Na získanie hodnoty konfigurácie slúži metóda **public AppConfig getAppConfig(String key)**.

Tabuľka 3.2: Tabuľka pre ukladanie konfigurácií.

ID	INT	primárny kľúč
KEY_NAME	VARCHAR	kľúč, pomocou ktorého pristupujeme ku konfigurácii
VALUE	VARCHAR	hodnota konfigurácie
DESCRIPTION	VARCHAR	popis

Analýza aplikácií

Hlavnou časťou webovej služby je analýza aplikácií. Na analýzu môžeme používať ľubovoľne veľa analyzérov. Každý analyzér musí implementovať rozhranie popísané v algoritme 1. Implementácia metódy **analyze** musí vrátiť výsledok analýzy. Formát a obsah výsledku analýzy je ľubovoľný a závisí od implementácie. Jediná požiadavka je, aby implementovala/rozširovala rozhranie *AnalysisResult*. Na spustenie analýzy nad aplikáciou môžeme použiť metódu aplikačnej služby - **analyze**. Prístup k metóde cez HTTP je nasledovný: *api / application /<id>/analyze*.

Algoritmus 1: Implementácia rozhrania analyzéra

```
1: public interface ApplicationAnalyzer<T extends AnalysisResult> {
2: T analyze(Application application);
3: AnalysisResultResource convertResultToResource(T result);
4: }
```

Samotná implementácia analýzy používa agregovaný analyzér, ktorý obsahuje niekoľko modulov. Každý modul obsahuje iný analyzér. Agregovaný analyzér spustí analýzu nad aplikáciou pre každý modul a výsledky analýzy spojí do agregovaného výsledku. Každému modulu je priradená hodnota váhy, ktorá určuje aký podiel na výsledku má mať výsledok analýzy modulu. Pomocou *AppConfigService* vieme nastavovať, ktoré moduly sa v analýze použijú a aké budú mať váhy. Implementáciou agregovaného analyzéru je trieda **public class AggregatedApplicationAnalyzer implements ApplicationAnalyzer<AggregatedAnalysisResult>**. Výsledok každého modulu sa normalizuje na hodnotu z intervalu <0, 1>. Následne pomocou váh modulov sa agregovaný výsledok tiež normalizuje na hodnotu z intervalu <0, 1>.

Každý modul musí implementovať rozhranie **public interface AnalysisModule<T extends AnalysisResult>**, respektíve rozširovať abstraktnú triedu **abstract public class AbstractAnalysisModule<T extends AnalysisResult> implements**

AnalysisModule<T>. Táto trieda implementuje základnú obsluhu modulu. Aby bol modul plnohodnotný, musí mať nastavený analyzér a typ výsledku, ktorý analýza produkuje a musí implementovať metódu na normalizáciu výsledku. V našej analýze používame tieto moduly: *PermissionAnalysisModule, PermissionUsageAnalysisModule, SimpleApplicationAnalyzer, MethodAnalysisModule*. Aktuálna konfigurácia modulov je v tabuľke 3.3.

Tabuľka 3.3: Aktuálna konfigurácia modulov

PERMISSION_ANALYSIS_MODULE_ENABLED	TRUE
PERMISSION_ANALYSIS_MODULE_WEIGHT	20
PERMISSION_ANALYSIS_MODULE_GROUPS	3.5
SIMPLE_ANALYSIS_MODULE_ENABLED	TRUE
SIMPLE_ANALYSIS_MODULE_WEIGHT	2
PERMISSION_USAGE_ANALYSIS_MODULE_ENABLED	TRUE
PERMISSION_USAGE_ANALYSIS_MODULE_WEIGHT	2
METHOD_ANALYSIS_MODULE_ENABLED	TRUE
METHOD_ANALYSIS_MODULE_WEIGHT	1

Algoritmus na analýzu aplikácií

Postupným skúmaním systému povolení a metód statickej analýzy sme dospeli k štyrom metódam analýzy, ktoré sme v našej práci implementovali:

PermissionAnalysisModule
Základom našej analýzy je modul, ktorý vykonáva analýzu povolení, ktoré aplikácia požaduje pri inštalácii. Analyzujeme podobnosť s inými - škodlivými - aplikáciami. Tu vychádzame z práce [82]. Od autorov práce sa nám podarilo získať rozsiahlu vzorku malvéru, ktorú zozbierali. Zanalyzovali sme zloženie a distribúciu požadovaných povolení týchto aplikácií. Ďalej sme vytvorili zoznam aktív nachádzajúcich sa na mobilných zariadeniach a povolení, ktoré s nimi priamo súvisia. Kombináciou týchto aktív a predošlej analýzy dostaneme skupiny "nebezpečných" povolení, ktoré najčastejšie využíva malvér. Tieto skupiny majú rôznu početnosť, vzhľadom na výpočtovú náročnosť sa nám najlepšie osvedčilo kontrolovanie trojíc, štvoríc a pätíc "nebezpečných" povolení. Tento spôsob detekcie sa ukázal ako veľmi účinný a jednoduchý na implementáciu. Poskytuje tak základ pre ďalšie, doplňujúce, moduly. Tento modul využíva vlastnú tabuľku v databáze 3.4.

Tabuľka 3.4: PERMISSION_ANALYSIS

ID	INT	primárny kľúč
PERMISSIONS	VARCHAR	zoznam ID povolení, ktoré testujeme
SCORE	FLOAT	dosiahnuté skóre po analýze
GROUP_ID	INT	ID skupiny, do ktorej patrí analýza

PermissionAnalysis je pomocná entita, pomocou ktorej hodnotíme bezpečnosť aplikácie. Hodnota *permissions* je reťazec, ktorý obsahuje čiarkami oddelené hodnoty ID povolení, ktoré chceme otestovať. Hodnota *score* určuje mieru "nebezpečia", ktoré zoznam v *permissions* predstavuje. Hodnota v *groupId* určuje, do ktorej skupiny analýzy patrí záznam. Analýza pomocou tohto modulu prebieha nasledovne:

1. Určíme hodnoty *groupId*, ktoré chceme testovať.
2. Z databázy zistíme všetky záznamy *permissionAnalysis*, ktoré patria medzi hodnoty *groupId*.
3. Zistíme zoznam povolení, ktorý obsahuje aplikácia, ktorú testujeme.
4. Testujeme zoznam povolení aplikácie, či obsahuje povolenia, ktoré obsahujú záznamy *permissionAnalysis*.
5. Spočítame skóre všetkých záznamov *permissionAnalysis*, ktoré vyhoveli testu v bode 4.
6. Výsledok jednoducho normalizujeme. Výslednú sumu skóre vydelíme maximálnou možnou dosiahnuteľnou hodnotou, ktorá sa mohla v analýze dosiahnuť.

Treba podotknúť, že úspešnosť a výpovedná hodnota analýzy spočíva v nastavení skupín povolení a ich hodnotenia v tabuľke PERMISSION_ANALYSIS. Na určovanie týchto hodnôt slúži pomocný analyzér *PermissionDistributionAnalyzer*.

SimpleApplicationAnalyzer
Toto je pôvodný modul na analýzu aplikácií, ktorý bol nasadený priamo na zariadení. Poskytuje len základnú analýzu aplikácií. Tento modul bol samostatne predstavený v práci [113] spolu s proof-of-concept aplikáciou. Jeho podstatou je rozdelenie aplikácií na kategórie podľa ich funkcionality, napr. hry, sociálne siete, financie, nástroje atď. Takisto aj povolenia sme rozdelili do štyroch skupín:

- ČERVENÉ (r) - vysoké až kritické riziko - tieto povolenia sú použiteľné len vybranou skupinou aplikácií, ktoré majú legitímny dôvod na ich použitie.
- ORANŽOVÉ (o) - stredné riziko - tieto povolenia by mali byť používané len aplikáciami, ktoré ich potrebujú na svoj korektný chod.
- ŽLTÉ (y) - malé riziko - tieto povolenia by mohli byť zaradené do skupiny ZELENÉ, ale v niektorých aplikáciách by mohli byť zneužiteľné.
- ZELENÉ (g) - malé až žiadne riziko - typické a bežne požadované povolenia, ktoré sú bezpečné, pokiaľ nie sú v kombinácii s niektorými ďalšími povoleniami.

V prvom kroku sme každú aplikáciu ohodnotili na základe tohto rozdelenia. Prvá časť hodnotenia R_1 sa vypočíta ako súčet hodnotení požadovaných povolení na základe vyššie spomenutého rozdelenia. Zelená skupina zvyšuje skóre o 0.5, žltá o 1, oranžová o 3 a červená o 5 bodov. Tento súčet vydelíme maximálnym možným skóre, t.j. hodnotou ak by všetky požadované povolenia boli z červenej skupiny. Tento výsledok nakoniec vynásobíme číslom 100, aby sme dostali mieru rizika v percentách, viď (3.1):

$$R_1 = \frac{g \cdot k_g + y \cdot k_y + o \cdot k_o + r \cdot k_r}{(g + y + o + r) \cdot k_r} \cdot 100, \tag{3.1}$$

kde g, y, o, r - je počet povolení zo skupín ZELENÉ, ŽLTÉ, ORANŽOVÉ a ČERVENÉ a
k_g, k_y, k_o, k_r - sú koeficienty rizika pre povolenia z každej skupiny.

V druhom kroku sme ku každej kategórii pridelili povolenia, ktoré pre ňu považujeme za bezpečné a tiež tie, ktoré považujeme za podozrivé, resp. nebezpečné. Za každé podozrivé povolenie sa zvýši skóre o hodnotu kategórie, v ktorej sa nachádza. Tento súčet znova vydelíme maximálnym možným skóre, podobne ako v prvom kroku, s tým rozdielom, že berieme do úvahy povolenia rozdelené pre konkrétnu kategóriu. Tento výsledok nakoniec vynásobíme číslom 100, aby sme dostali mieru rizika R_2 v percentách, viď (3.2):

$$R_2 = \frac{p_s \cdot k_s + p_n \cdot k_n}{p \cdot k_s} \cdot 100, \tag{3.2}$$

kde p - celkový počet povolení,
p_s - počet podozrivých povolení pre aktuálnu kategóriu,
k_s - koeficient rizika pre podozrivé povolenia z tejto kategórie,
p_n - počet bezpečných povolení pre aktuálnu kategóriu,
k_n - koeficient rizika pre bezpečné povolenia z tejto kategórie.

Výsledky týchto testov sa normalizujú, sčítajú a vypočíta sa hodnota aritmetického priemeru, keďže obe časti algoritmu sú rovnocenné. Takto dosiahneme finálne hodnotenie R, viď (3.3):

$$R = \frac{R_1 + R_2}{2}, \tag{3.3}$$

kde R - finálne skóre,
R_1 - skóre z prvej časti algoritmu,
R_2 - skóre z druhej časti algoritmu.

Po bližšom preskúmaní a testovaní modulov **PermissionAnalysisModule** a **SimpleApplicationAnalyzer** sme dospeli k záveru, že je možné ich čiastočne obísť, resp. znížiť ich presnosť. Táto situácia nastane, ak testovanej aplikácii zapíšeme do *manifestu* veľké množstvo povolení. Pri ich vhodnej kombinácii môžeme oklamať prvý (neškodné povolenia) aj druhý (zo zelenej, resp. bezpečnej množiny) modul.

Nebezpečné povolenia sa takto vo veľkom množstve povolení "stratia". V takomto prípade môže aplikácia dostať nízke hodnotenie napriek škodlivej funkcionalite. Preto sme sa rozhodli pridať ďalšie dva moduly, ktoré by mali podobné situácie čiastočne eliminovať.

PermissionUsageAnalysisModule
Tento modul slúži na analýzu nepotrebných povolení a je prvým z dvoch pomocných modulov. Pri vložení aplikácie do databázy používame pomocný komponent, ktorý stiahne inštalačný *.apk* súbor aplikácie, rozbalí ho a dekompiluje na čitateľný zdrojový kód. Následne analyzuje či systémové volania nachádzajúce sa v kóde reálne zodpovedajú povoleniam, ktoré sú deklarované v *manifeste* aplikácie. Ak sa nájdu povolenia, ktoré aplikácia nepoužíva, môže to indikovať tri možnosti:

- Vývojár skopíroval *manifest* z predošlého projektu do aktuálneho a prípadne pridal ďalšie potrebné povolenia. V tomto prípade ide o neúmyselné zvýšenie bezpečnostného rizika aplikácie. Tento stav znižuje dôveryhodnosť vývojára a zvyšuje pravdepodobnosť ďalších bezpečnostných nedostatkov aplikácie z dôvodu nedôslednosti vývojára.

- V druhom prípade tento fakt môže indikovať, že nevyužívané povolenia tam boli pridané z nejakého dôvodu. Aplikácia môže byť o nejaký čas aktualizovaná a "obohatená" o škodlivú funkcionalitu. V takomto prípade by už nemusela žiadať používateľa o potvrdenie potrebných povolení. Bolo by tak možné nevedomky si stiahnuť škodlivú funkcionalitu v nainštalovanej repacknutej aplikácii.

- V poslednom prípade útočník pozná systém hodnotenia aplikácií v našom riešení. Povolenia pridal do aplikácie za účelom oklamania analýzy a umelého zníženia výsledného hodnotenia svojej aplikácie.

Ani jeden z prípadov by sa v legitímnej a bezpečnej aplikácii nemali vyskytovať. Preto sme považovali potrebné na tento fakt aspoň v krátkosti upozorniť. Normalizácia výsledku sa vypočíta ako (počet nepotrebných (nepoužitých) povolení) / (počet povolení v manifeste).

MethodAnalysisModule
Toto je druhý pomocný modul určený na statickú analýzu kódu aplikácie. Pomocou skriptu bežiacom na serveri sa rozbalí a dekompiluje *.apk* súbor aplikácie a vykoná sa statická analýza kódu, či neobsahuje nebezpečné metódy. Príkladom takejto metódy je metóda *invoke*. Ďalej sledujeme napr. či sa aplikácia nepripája na neznámy server, alebo nečaká na aktualizáciu. Väčšina moderných rodín malvéru sa snaží uniknúť pred analýzou a jedna z prvých vecí, ktoré sa zvyčajne snažia urobiť, je skontrolovať, či sú spustené v kontrolovanom prostredí. Riadené prostredie v tomto kontexte znamená emulátor. Ak malvér beží na emulátore, znamená to, že je s najväčšou pravdepodobnosťou vyšetrovaný, prípadne analyzovaný. Na detekciu, či malvér beží v kontrolovanom prostredí, vývojári malvéru zvyknú požívať nasledujúce

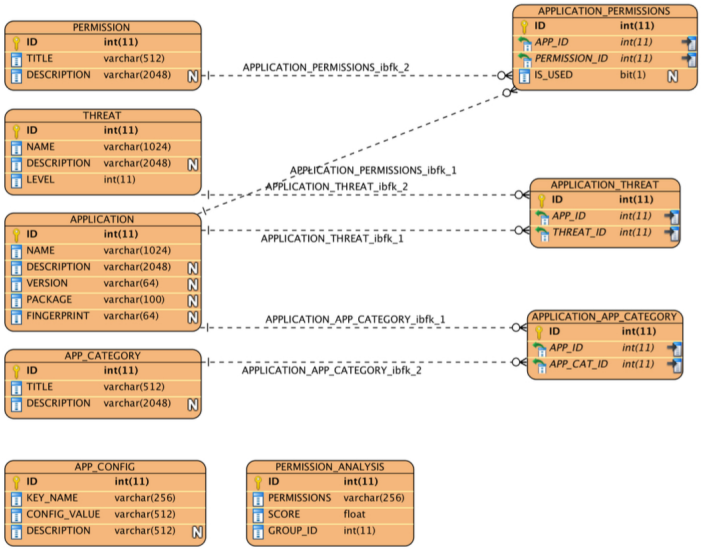
metódy: *Build.PRODUCT.contains("sdk"), Build.MODEL.contains("sdk"), localTelephonyManager.getSimOperatorName().equals("Android"), find_qemud_process()*. Po dekompilácii sa snažíme hľadať aj tieto metódy, ktorých prítomnosť naznačuje, že by sa mohlo jednať o malvér, čo sa zohľadňuje pri vyhodnocovaní rizika v algoritme popísanom v podkapitole o module *AnalysisModule*. Oba pomocné moduly sú aktívne len v prípade, že je aktívny aspoň jeden z hlavných modulov.

V neskoršej fáze práce sme na začiatok celej analýzy zapracovali funkcionalitu na overovanie podpisov aplikácií. Všetky Android aplikácie, ktoré sú publikované v oficiálnom obchode Google Play, musia byť podpísané certifikátom vývojára. Tieto podpisy je možné získať priamo z *.apk* súborov. Hlavným dôvodom tohto doplnenia je upozorňovanie na tzv. wrapnuté/repacknuté aplikácie - populárne aplikácie, ku ktorým bol pridaný nebezpečný kód a následne boli podpísané certifikátom útočníka. Náš systém porovná SHA-1 odtlačok certifikátu získaný za behu aplikácie s odtlačkom z referenčného zdroja - Google Play. Ak sa odtlačky nerovnajú, je vysoko pravdepodobné, že daná aplikácia je wrapnutá/repacknutá a obsahuje škodlivý kód. Takúto aplikáciu automaticky hodnotíme ako podozrivú.

Databázový model

Všetky dáta sú ukladané do MySQL [119] databázy, odkiaľ k nim na analýzu pristupujú jednotlivé komponenty. Databázový model použitý v našej práci prešiel viacerými úpravami počas prác. Konečný model sa skladá z desiatich tabuliek, ktoré majú medzi sebou príslušné vzťahy. Model je zobrazený na obr. 3.11. Popis jednotlivých tabuliek sa nachádza pod obrázkom. V modeli sa nachádza len deväť tabuliek, desiatu pre používateľský vstup sme popísali v samostatnej časti.

- **PERMISSION** - zoznam povolení v OS Android. Každá aplikácia má povolenia na prístup iba k určitým zdrojom a miestam v systéme. V tabuľke sú vymenované jednotlivé povolenia a ich charakteristika.
- **THREAT** - zoznam možných hrozieb ktoré ohrozujú zariadenie a súkromie používateľa. Názov, popis, úroveň - ako veľmi je dané ohrozenie vážne (strata financií, únik osobných údajov atď.).
- **APPLICATION** - záznamy tabuľky obsahujú základné informácie o aplikáciach ako sú názov aplikácie, krátky opis a verzia danej aplikácie.
- **APP_CATEGORY** - zoznam kategórií aplikácií. Do týchto kategórií sú aplikácie zaradené aj v Google Play obchode (šport, zdravie, cestovanie, hry).
- **APPLICATION_PERMISSION** - spojovacia tabuľka k tabuľkám APPLICATION a PERMISSIONS. Ide o priradenie povolení k jednotlivým aplikáciám.
- **APPLICATION_THREAT** - spojovacia tabuľka k tabuľkám APPLICATION a THREAT. Ide o priradenie hrozieb k jednotlivým aplikáciám.



Obr. 3.11: Databázový model

- **APPLICATION_APP_CATEGORY** - spojovacia tabuľka k tabuľkám APPLICATION a APP_CATEGORY. Ide o priradenie kategórie k jednotlivým aplikáciám.
- **APP_CONFIG** - v tejto tabuľke je uložená aktuálna konfigurácia modulov používaných na analýzu aplikácií.
- **PERMISSION_ANALYSIS** - pomocná tabuľka na analýzu povolení.

Používateľský vstup

Najpodstatnejšou časťou používateľského vstupu na serverovej časti bolo vytvorenie ďalšej tabuľky USER_INPUT na uchovávanie tohoto hodnotenia. Tabuľka je veľmi jednoduchá, obsahuje 5 riadkov:

- ID - identifikačné číslo záznamu v tabuľke.
- APP_ID - je to identifikačné číslo aplikácie, prepojenie na tabuľku APPLICATION.
- ID_DEVICE - je identifikačné číslo Android_ID, získané z Android zariadenia používateľa.
- USER_VALUE - je hodnota z rozsahu 1-5 pričom 1 je najmenej nebezpečné a 5 predstavuje najvyšší stupeň hrozby.
- ALGORITHM_VALUE - je výsledok automatického hodnotiaceho algoritmu.

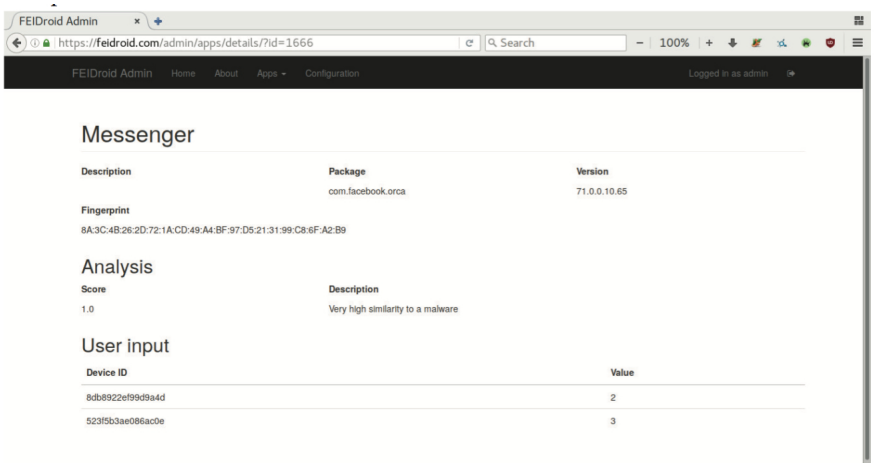
Na to, aby sme mohli spracovávať vstup od používateľa sme museli do webovej služby implementovať spracovanie metódy *POST*, ktorá zapísala tieto údaje do tabuľky a metódy *GET*, ktorá dokáže tieto údaje z tabuľky načítať. K USER_INPUT tabuľke sme vytvorili pomocou generátora entít v Eclipse entitnú triedu *UserInput.java*, ktorá je generovaná automaticky do priečinka */entities*. K tejto triede sme museli vytvoriť *UserInputResource* v priečinku */resources*, aby mohla samotná webová služba pristupovať k metódam v tejto triede. Na spracovanie *POST* požiadaviek sme upravili triedu *ApplicationService.java* umiestnenej v priečinku */services*. K tejto triede sme museli doplniť triedu *UserinputService.java*, ktorá nám zabezpečila vyhľadanie jedinečného spojenia aplikácie a zariadenia pomocou metódy *findByAppAndIdDevice(app, idDevice)*. V triede *UserinputService.java* sme vytvorili taktiež metódy *convertResourceToEntity(userinputResource)* a *convertEntityToResource (userinput)* na správnu konverziu medzi entitami a resourcami.

Administrátorské rozhranie

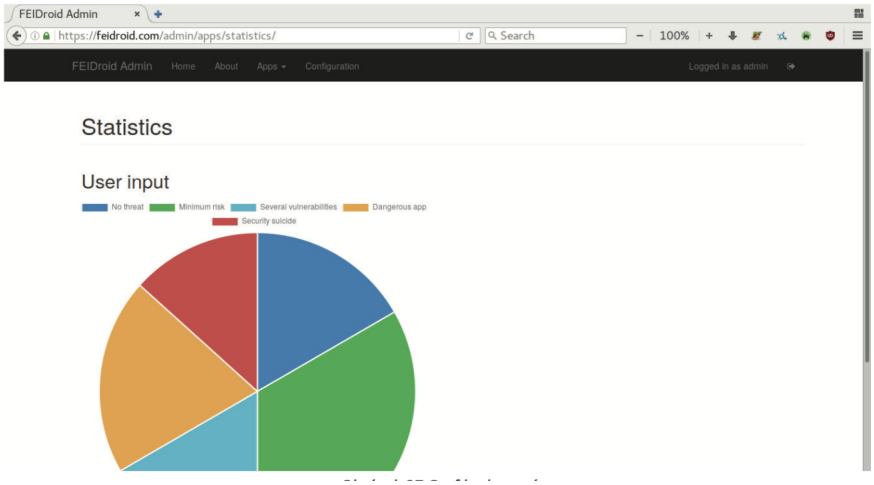
Implementačne ide o najjednoduchšiu časť nášho riešenia. Pôvodne sme nemali v pláne pridávať webové rozhranie. Z dôvodu prehľadnosti a lepšej práce s nastaveniami hodnotiacich algoritmov sme sa ale napokon rozhodli vytvoriť veľmi jednoduché webové administrátorské rozhranie. Je napísané v jazyku Python 2 [129], s použitím webového frameworku Flask [130] a niekoľkých knižníc. Rozhranie splňa všetky potrebné funkcionality, t.j. umožňuje nasledovné:

- Prezerať si zoznam analyzovaných aplikácií.
- Zobraziť detail konkrétnej aplikácie spolu s hodnotením používateľov, viď obr. 3.12.
- Nahrávať *.apk* súbory na analýzu.
- Zobraziť v grafe všetky používateľské hodnotenia, viď obr. 3.13.
- Meniť nastavenia parametrov pri analýze aplikácií, viď obr. 3.14.

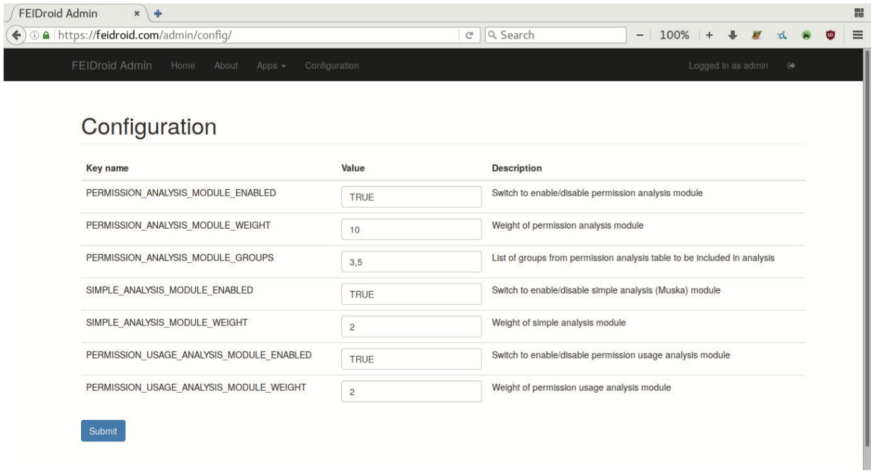
Na to, aby mal k jednotlivým častiam rozhrania prístup, musí byť používateľ registrovaný a prihlásený. Aby mohol vykonávať zmeny na serveri, musia mu byť pridelené administrátorské práva.



Obr. 3.12: Detail aplikácie v administrátorskom rozhraní



Obr. 3.13: Zobrazenie používateľských hodnotení na grafe



Obr. 3.14: Nastavenie konfigurácie na analýzu aplikácií

Kapitola 4

Výsledky a vyhodnotenie

V poslednej časti dizertačnej práce sa budeme venovať testovaniu nášho riešenia. V kapitole 3.1.2 sme si zadefinovali kritériá na posúdenie úspešnosti riešenia. Nie všetky z nich sa dajú exaktne číselne vyjadriť, pretože záleží na subjektívnom dojme používateľa z používania našej aplikácie. Preto sme sa zamerali na číselne vyjadriteľné aspekty testovania a pridali sme niekoľko ďalších analýz. V našej databáze sa nachádza 1294 aplikácií evidovaných ako malvér rôznymi databázami, či portálmi ako napr. VirusTotal. Okrem malvéru sa nám podarilo získať 870 inštalačných súborov rôznych neškodných legitímnych aplikácií. Nie so všetkými vzorkami sme však vedeli pracovať na minimálnej úrovni, t.j. dostať sa aspoň k dekompilácii a manifestu. Preto v každej ďalšej časti práce budeme pracovať len s konkrétnymi počtami vzoriek viažúcich sa k príslušnej časti.

4.1 Rýchlosť vyhodnotenia analýzy

Pod pojmom rýchlosť analýzy rozumieme čas potrebný na vyhodnotenie analýzy vybranej aplikácie. V najpomalšom prípade je potrebné aplikáciu stiahnuť, dekompilovať a zanalyzovať. Vzhľadom na fakt, že nedokážeme ovplyvniť aktuálnu prenosovú rýchlosť dát, bude táto hodnota subjektívna k času vykonania merania. Ďalej sme sa zamerali na rýchlosť otestovania vzoriek dostupných v našej databáze pri prvom spustení, kedy tieto hodnoty môžu byť referenčné v prípade, že používateľ bude prvý, kto odošle na server požiadavku na kontrolu takejto aplikácie. Vtedy musí celý proces prebehnúť od začiatku, keďže o aplikácii nie je záznam v databáze. V prípade, že aplikácia už bola niekedy testovaná, naše riešenie len vyhľadá záznam vo svojej databáze a výsledky kontroly pošle používateľovi okamžite. Pri tejto sérii testov sme pracovali s objemom dát, nie s jednotlivými aplikáciami. Tento postup považujeme za vhodnejší, keďže viaceré aplikácie majú veľkosť rádovo v desiatkach až stovkách MB, a v porovnaní s oveľa menšími vzorkami by boli výsledky do značnej miery skreslené. Pri testovaní sme aplikácie vyberali náhodne z databázy, aby sme dosiahli čo najvyššiu variabilitu testovaných dát.

Tabuľka 4.1: Rýchlosť sťahovania vzoriek

Objem dát v MB	Priemerný potrebný čas v s na 1MB dát
1	6
10	5.5
30	4
50	3.3
100	3
500	2.8
1000	2.5

Tabuľka 4.2: Rýchlosť dekompilácie a analýzy vzoriek

Objem dát v MB	Priemerný potrebný čas v s na 1MB dát
1	11
10	10.5
30	3
50	1.8
100	1.95
500	1.93
1000	1.91

Z tabuľky 4.1 vidíme, že so zväčšujúcim sa objemom dát potrebným na analýzu klesá priemerný čas, potrebný na stiahnutie 1MB dát. Predpokladáme však, že aplikácie posielané na analýzu budú menšie ako 100MB, keďže pri väčších aplikáciách by používateľov odradzoval objem prenesených dát potrebný na analýzu.

Rýchlosť dekompilácie a samotnej statickej analýzy vzoriek už závisí nielen od objemu analyzovaných dát, ale aj od počtu analyzovaných aplikácií, viď tabuľka 4.2. Pri väčšom počte aplikácií sa musí vykonať väčšie množstvo operácií potrebných na dekompiláciu *.apk* súborov. V priemere sa však neobjavili žiadne výrazné odchýlky. Ďalším významným prvkom aplikácií, ktorý ovplyvňuje rýchlosť analýzy je zložitosť zdrojového kódu aplikácie. Nemusí totiž platiť, že čím je aplikácia "objemovo" väčšia, musí byť aj zložitejšia. Príkladom môže byť aplikácia na prezeranie rôznych vtipných obrázkov - "objemovo" obsahuje síce veľa dát kvôli obrázkom, ale funkcionálne je veľmi jednoduchá a jej zdrojový kód vieme pomerne rýchlo zanalyzovať. Posledným faktorom, ktorý môže analýzu výrazne spomaliť je prítomnosť obfuskovaného kódu v aplikácii. V takomto prípade trvá analýza aj malej aplikácie rádovo desiatky sekúnd a nemusí byť v konečnom dôsledku úspešná. Pokles času potrebného na spracovanie pri objeme dát 50MB preto pripisujeme skladbe množiny testovaných aplikácií a ďalším, vyššie uvedeným faktorom.

Ako sme už spomenuli vyššie, predpokladáme, že postupne naplníme našu databázu otestovaných aplikácií do takej miery, že na vykonanie vyššie spomenutých úkonov ne-

budú musieť používatelia vôbec čakať, resp. len vo výnimočných prípadoch. V súčasnom stave naše riešenie poskytuje používateľovi pri malých aplikáciách vysokú rýchlosť odozvy na jeho požiadavku. Pri väčších aplikáciách rýchlosť odozvy záleží hlavne na rýchlosti pripojenia a zložitosti analyzovanej aplikácie.

4.2 Rozdelenie vzoriek do kategórií

Ešte pred samotným testovaním presnosti detekcie malvéru sme sa rozhodli zmapovať celkovú distribúciu malvéru do jednotlivých kategórií, resp. aplikácie z akých kategórií sa najčastejšie používajú na ukrytie malvéru napr. pri repackingu. Výsledky sú prehľadne spracované v tabuľke 4.3. Všetky aplikácie, resp. vzorky malvéru sme manuálne nainštalovali na testovacie zariadenie a spustili. Nepodarilo sa nám nainštalovať všetky vzorky (1193 z 1294), hlavne kvôli nekompatibilite so zariadením alebo verziou OS. V týchto prípadoch sa nám ani z názvov aplikácií nepodarilo určiť ich kategóriu. Podobne sme pre potrebu ďalšieho zohľadnenia výsledkov spracovali aj legitímne aplikácie v našej databáze, viď tabuľku 4.4. Pri legitímnych aplikáciách sme z názvu vedeli určiť ich kategórie.

Záznamy v tejto tabuľke zodpovedajú zoznamu aplikácií, ktoré sa nám podarilo stiahnuť na testovacie účely. Zastúpenie v jednotlivých kategóriách preto môže byť viac naklonené ku viac dostupným kategóriám. Čísla v zátvorkách zodpovedajú počtom aplikácií, pri ktorých sa nám podarilo dostať sa k manifestu a získať požadované povolenia. Rozdiely sú spôsobené hlavne dvomi faktormi: prvým je fakt, že viacero legitímnych aplikácií, hlavne z kategórií Games, Tools a Multimedia vôbec nevyžadujú na svoju činnosť akékoľvek povolenia. Druhým je, že staršie verzie aplikácií na svoju činnosť nemuseli vyžadovať žiadne povolenia a ich následné vyžadovanie je spôsobené pridaním funkcionality do novšej verzie aplikácie.

Podľa očakávania najväčšia časť vzoriek spadala do kategórie Games, teda rôznych hier. Je to pochopiteľné, keďže aplikácie z tejto kategórie na svoju činnosť potrebujú často rôzne povolenia, ako napr. prístup ku kontaktom či internetu kvôli zdieľaniu výsledkov a úspechov v hre s priateľmi. Ďalej môžu vyžadovať prístup s prémiovým SMS správam, aby mohli používatelia vykonávať tzv. in-app purchases, teda nakupovať rôzne prídavky do hry (nové úrovne, kostýmy, zbrane atď.). Ďalšou kategóriou v poradí, pre nás pomerne prekvapivou, boli multimédiá. Do tejto kategórie spadajú rôzne prehrávače hudby, videí a hlavne čítačky na e-knihy. Knihy rôzneho žánru sú upravené do formy aplikácie, ktorú si používateľ stiahne a číta na svojom zariadení. Tu sme si pri drvivej väčšine aplikácií všimli, že požadujú povolenia, ktoré na svoju deklarovanú činnosť nepotrebujú, ako napr. prístup k Internetu, kontaktom, SMS správam a pod.

V porovnaní s predošlou kategóriou sú už pri bližšom preskúmaní pri inštalácii označiteľné za minimálne podozrivé. Pokiaľ však používatelia len odklikávajú potvrdenia

Tabuľka 4.3: Celková distribúcia malvéru vzhľadom na kategóriu

Kategória	Počet vzoriek v kategórii	%
Sports	0	0
Health	6	0.46
Lifestyle	23	1.78
Games	358	27.68
Travel	4	0.31
Multimedia	322	24.88
Entertainment	110	8.50
Social	6	0.46
Tools	276	21.33
Communication	19	1.47
Shopping	2	0.15
Productivity	7	0.54
Themes	58	4.48
Finance	2	0.15
Nezaradené	101	7.81
Celkovo	1294	100

a nečítajú, čo aplikácia reálne požaduje, tak sa škodlivý kód šíri veľmi jednoducho. Treťou najväčšou kategóriou sú Tools, teda rôzne aplikácie na správu zariadenia, ako napr. na zvýšenie výdrže batérie, čistenie úložiska, operačnej pamäte a pod. Podobne ako pri hrách, aj tieto vyžadujú veľké množstvo povolení, ktoré sú potrebné na vykonávanie ich legitímnej funkcionality, ale sú rovnako využiteľné aj na škodlivú činnosť. Na rozdiel od multimédií je ich detekcia "voľným okom" pri inštalácii takmer nemožná.

Podobne ako multimédiá, čo sa týka požadovaných povolení, vyzerajú aj ďalšie dve kategórie Entertainment a Themes, teda zábava a témy. Zábavné aplikácie ležia na pomedzí hier a multimédií, príkladom môže byť aplikácia na vydávanie nejakého zvuku, napr. trúbka, klaxón a pod. Do tém zasa spadajú aplikácie na úpravu pozadia na zariadení, zmenu výzoru tlačidiel a pod. Rovnako ako pri multimédiách aj tieto požadujú povolenia na prvý pohľad podozrivé pre tento typ aplikácií. Ostatné kategórie sú oproti týmto piatim zastúpené minimálne. Očakávali sme väčšie zastúpenie v kategóriách Communication, Social, Shopping, Finance či Travel, lebo tieto na svoje fungovnie nutne požadujú povolenia na prístup k Internetu, kontaktom, telefónu, emailu, SMS správam a pod.

Pri tomto vyhodnotení sme si všimli, že pri početnejších rodinách malvéru môžu nastať dve situácie v distribúcii vzoriek do kategórií. Prvou je, keď jedna rodina patrí len do jednej kategórie aplikácií, prípadne len s minimálnymi odchýlkami. Príkladom môžu byť Kmin (52 v Themes), BaseBridge (88 v Multimedia) či AnserverBot

Tabuľka 4.4: Celková distribúcia legitímnych aplikácií vzhľadom na kategóriu

Kategória	Počet vzoriek v kategórii	%
Sports	18 (16)	2.07
Health	3 (3)	0.34
Lifestyle	34 (24)	3.91
Games	87 (70)	10.00
Travel	70 (62)	8.05
Multimedia	107 (90)	12.30
Entertainment	14 (14)	1.61
Social	95 (77)	10.92
Tools	259 (56)	29.77
Communication	64 (55)	7.36
Shopping	24 (24)	2.76
Productivity	68 (64)	7.82
Themes	5 (0)	0.57
Finance	22 (20)	2.53
Celkovo	870 (575)	100

(157 v Multimedia). Druhou je rozdelenie vzoriek do viacerých kategórií. ako napr. DroidKungFu4 (40 v Games, 17 v Multimedia, 34 v Tools) či DroidKungFu3 (36 v Tools, 10 v Multimedia, 175 v Games a 57 v Entertainment). Dôvodov na jedno či druhé správanie je viacero. V prvom prípade ide o špecializovaný malvér vyrábaný akoby podľa zopár vizuálnych šablón a mení sa len obsah, napr. e-knihy. Najpravdepodobnejšie vysvetlenie bude asi to, že autorom išlo o rýchle vytvorenie čo najväčšieho počtu rôznych škodlivých aplikácií pre "jednorazovú akciu" za účelom rýchleho zisku a ďalej sa už funkcionalita nemodifikovala.

V druhom prípade je situácia presne opačná. Autori malvéru svoje riešenia pravidelne modifikovali, aby sa vyhli antivírovej kontrole. Preto sa objavujú v rôzne pozmenenej forme vo viacerých kategóriách, za ktoré sa tieto vzorky maskujú. Odhalenie týchto modifkácií záleží hlavne na miere podobnosti s už odhalenými variantami. Napr. pri menej početných variantoch rodiny DroidKungFu nazvaných DroidKungFuSapp a DroidKungFuUpdate bola doba odhalenia niekoľko mesiacov, keďže boli výrazne pozmenené oproti DroidKungFu3 či DroidKungFu4.

V skupine legitímnych neškodných aplikácií, tabuľka 4.4 sú najvýraznejšie zastúpené aplikácie z kategórie Tools. Sem patria rôzne aplikácie na správu zariadenia či antivírové aplikácie. Na ďalšom mieste sú Multimedia, hlavne aplikácie na prehrávanie videí (YouTube), prehrávače hudby (Spotify) a iného multimediálneho obsahu. Nasledujú aplikácie rôznych sociálnych sietí ako FaceBook, Twitter či SnapChat. Až na štvrtom mieste sú aplikácie z kategórie Games, nasledované kategóriou Travel, kam spadajú rôzne navigačné aplikácie, rezervácie ubytovania a turistickí sprievodcovia.

Významnejšie sú zastúpené aj kategórie Productivity (napr. kalendáre, plánovače) či Communication (WhatsApp, Messenger).

V percentuálnom pomere vidíme viaceré rozdiely medzi tabuľkami 4.3 a 4.4. Pri legitímnych aplikáciách výrazne stúpol výskyt aplikácií z kategórií Sports, Travel, Social, Communication, Shopping, Productivity či Finance. Naopak, klesli Games, Multimedia či Entertainment. Predpokladáme, tieto štatistiky vyplývajú z vyššie uvedenej distribúcie malvérových vzoriek do kategórií, v ktorých sa môžu ľahšie zamaskovať za legitímne aplikácie. Naopak, je pre ne zjavným problémom maskovať sa za aplikácie od overených vývojárov, resp. overených značiek ako FaceBook, Spotify či WhatsApp. Takéto aplikácie sú dôsledne kontrolované a zverejňované na takmer každom aplikačnom úložisku (markete). Ich rozšírenosť má navyše za dôsledok to, že ich dizajn - od farieb po rozloženie ovládacích prvkov (layout) - je tak známy, že podvrh je na prvý pohľad viditeľný aj pre bežného používateľa. Takže aj keď by bolo pre útočníka lákavé takúto aplikáciu napodobniť a zneužiť, je vysoko pravdepodobné, že tento pokus by bol nebol výrazne úspešný.

4.3 Presnosť výsledkov analýzy

Pred prezentovaním výsledkov si v krátkosti pripomenieme jednotlivé moduly hodnotiaceho algoritmu:

- *PermissionAnalysisModule* - prvý základný modul, ktorý určuje "škodlivosť" aplikácie na základe podobnosti požadovaných povolení a ich skupín s povoleniami obsiahnutými vo vzorkách malvéru.
- *SimpleApplicationAnalyzer* - druhý základný modul, ktorý hodnotí aplikácie na základe ich príslušnosti do skupín (hry, multimédiá), kde pre každú skupinu máme povolenia rozdelené do skupín podľa ich rizika.
- *PermissionUsageAnalysisModule* - prvý pomocný modul - kontroluje, či sa povolenia žiadané v *manifeste* aplikácie využívajú aj v zdrojovom kóde, alebo nie.
- *MethodAnalysisModule* - druhý pomocný modul - kontroluje, či zdrojový kód aplikácie neobsahuje nebezpečné metódy alebo reťazce.

Každý modul vypočíta svoje skóre pre konkrétnu aplikáciu a tieto skóre sa v agregátore váhujú v závislosti od nami zadaného nastavenia, znormalizujú a nakoniec sú prezentované používateľovi. V tejto časti sme na testovanie využili všetky dostupné aplikácie, aj tie ktoré sme nevedeli zaradiť do kategórie. Takéto aplikácie sú zaradené do špeciálnej kategórie, kde sú povolenia rovnomerne rozdelené do skupín. Výsledky z týchto 101 aplikácií tak môžu mierne skresľovať celkový výsledok.

Presnosť analýzy v našom ponímaní znamená vyjadrenie veľkosti chyby prvého druhu a chyby druhého druhu. Ich závislosť od nastavenia váh jednotlivým modulom

Tabuľka 4.5: Úspešnosť detekcie malvéru a správneho zadelenia legit. aplikácií pri nastavení 1-0-0-0

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	451	34.85	762	87.59
$< 0.2; 0.4)$	259	20.02	48	5.52
$< 0.4; 0.6)$	135	10.43	27	3.10
$< 0.6; 0.8)$	27	2.09	9	1.03
$< 0.8; 1 >$	422	32.61	24	2.76
Celkovo	1294	100	870	100

Tabuľka 4.6: Úspešnosť detekcie malvéru a správneho zadelenia legit. aplikácií pri nastavení 0-1-0-0

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	8	0.62	343	39.43
$< 0.2; 0.4)$	99	7.65	211	24.25
$< 0.4; 0.6)$	415	32.07	201	23.10
$< 0.6; 0.8)$	257	19.86	93	10.69
$< 0.8; 1 >$	515	39.80	22	2.53
Celkovo	1294	100	870	100

ilustrujú tabuľky 4.5 až 4.8, kde je aktívny vždy len jeden z hlavných modulov, alebo sú všetky moduly, vrátane pomocných, nastavené na rovnakú váhu. Finálne, najlepšie nastavenie váh pre všetky moduly, vyjadruje tabuľka 4.9. Jednotlivé riadky tabuľky vyjadrujú, koľko vzoriek malvéru dostalo hodnotenie v nami zadefinovaných rozsahoch podobnosti s malvérom. Čím bližšie bolo hodnotenie vzoriek malvéru k 1, tým presnejšie bola škodlivá vzorka správne označená ako malvér, a čím bližšie bolo hodnotenie legitímnej vzorky k 0, tým presnejšie bola táto vzorka označená ako legitímna. Počet vzoriek malvéru so skóre bližším k 0 ako k 1 vyjadruje veľkosť chyby prvého druhu. Analogicky, počet legitímnych aplikácií so skóre bližším k 1 ako k 0 vyjadruje veľkosť chyby druhého druhu.

Pri prvej základnej konfigurácii vidíme pomerne rovnomerné rozdelenie testovaných vzoriek do nami definovaných kategórií, viď tabuľka 4.5. Viac ako polovica vzoriek malvéru by v tomto prípade bola označená len ako mierne riziko. Len niečo vyše tretina celkového počtu vzoriek by bola správne označená ako škodlivé aplikácie. Je to spôsobené tým, že nie všetky chybne označené aplikácie vyžadujú na svoju činnosť veľké množstvo povolení. Pokiaľ je požadovaných povolení málo, napr. $3 - 5$, tak sa nevytvoria dostatočne veľké skupiny, resp. nevytvorí sa ich dostatočný počet, a preto dostanú nižšie skóre podobnosti s malvérom. Pomocou tohto istého nastavenia sme otestovali aj legitímne aplikácie z našej databázy. Vyše 93% týchto aplikácií bolo správne určených ako legitímne.

Tabuľka 4.7: Úspešnosť detekcie malvéru a správneho zadelenia legit. aplikácií pri nastavení 1-1-0-0

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	104	8.04	579	66.55
$< 0.2; 0.4)$	375	28.98	187	21.49
$< 0.4; 0.6)$	306	23.65	49	5.63
$< 0.6; 0.8)$	96	7.42	35	4.02
$< 0.8; 1 >$	413	31.91	20	2.30
Celkovo	1294	100	870	100

Druhá základná konfigurácia využíva presne zadefinované zoznamy nebezpečných povolení pre jednotlivé kategórie aplikácií. Tieto kritériá sú prísnejšie ako hodnotenie na základe výskytu skupín povolení, čo sa pozitívne odrazilo aj vo výsledkoch testu. Z tabuľky 4.6 vidíme, že takmer 60% vzoriek malvéru bolo s vysokou spoľahlivosťou vyhodnotených korektne. Ďalších 32% vzoriek bolo určených nejednoznačne, kde by bola potrebná dodatočná kontrola. Prísnosť týchto kritérií však negatívne ovplyvnila úspešnosť posúdenia legitímnych aplikácií. Správne bolo posúdených len 63.7% takýchto aplikácií. Zvyšné legitímne aplikácie vyžadujú povolenia, ktoré pokladáme za podozrivé, resp. priamo nebezpečné pre dané kategórie, a tým pádom je ich skóre podobnosti s malvérom výrazne vyššie ako pri predošlom nastavení.

Tretia základná konfigurácia pozostáva zo zapojenia oboch hlavných modulov s rovnakými váhami. Z tabuľky výsledkov 4.7 vidíme, že oproti prvej konfigurácii výrazne klesol počet nesprávne posúdených škodlivých aplikácií (prvé dva riadky v tabuľkách) o 18%. Počet nejednoznačne ohodnotených aplikácií (stredný riadok) vzrástol o 13% a počet správne označených vzoriek (posledné dva riadky) stúpol na 39%. Môžeme teda skonštatovať, že oproti prvej konfigurácii sa úspešnosť detekcie malvéru zvýšila. V porovnaní s druhou základnou konfiguráciou sú však výsledky horšie. Počet správne označených vzoriek malvéru klesol o približne 20%. Naopak, počet nesprávne vyhodnotených vzoriek stúpol o bezmála 30%. Počet nejednoznačne určených vzoriek síce klesol o takmer 10%, ale považujeme to za redistribúciu vzoriek do iných riadkov v rámci výsledkov. Zaujímavý je pohľad na hodnotenie legitímnych aplikácií pri tomto nastavení. Až 88% z nich je správne určených, čo je len minimálna odchýlka od prvého nastavenia. V porovnaní s druhým je to nárast o 25%. Počet nesprávne (6.3%), či nejednoznačne (5.6%) vyhodnotených vzoriek takisto výrazne nestúpol pri porovnaní s prvou konfiguráciou. V porovnaní s druhou je to pokles o 7 resp. o takmer 17%.

Posledná základná konfigurácia zapojila do analýzy všetky hlavné aj pomocné moduly, všetky s rovnakou váhou. Výsledky sú prezentované v tabuľke 4.8. Ako vidíme z tabuľky, výsledky analýzy škodlivých aplikácií s takouto konfiguráciou sú nevyhovujúce. Je tu badať výrazný konflikt hlavných a pomocných modulov. Výsledky hlavných

Tabuľka 4.8: Úspešnosť detekcie malvéru a správneho zadelenia legit. aplikácií pri nastavení 1-1-1-1

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	618	47.76	784	90.11
$< 0.2; 0.4)$	336	25.96	66	7.59
$< 0.4; 0.6)$	339	26.20	19	2.18
$< 0.6; 0.8)$	1	0.08	1	0.11
$< 0.8; 1 >$	0	0.00	0	0.00
Celkovo	1294	100	870	100

Tabuľka 4.9: Úspešnosť detekcie malvéru a správneho zadelenia legit. aplikácií pri nastavení 10-30-1-1

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	47	3.63	431	49.54
$< 0.2; 0.4)$	268	20.71	281	32.30
$< 0.4; 0.6)$	363	28.05	93	10.69
$< 0.6; 0.8)$	247	19.09	44	5.06
$< 0.8; 1 >$	369	28.52	21	2.41
Celkovo	1294	100	870	100

modulov sú do veľkej miery potlačené výsledkami pomocných modulov, ktoré vo svojej podstate dávajú len odpoveď "áno" alebo "nie", resp. "1" alebo "0". Preto je potrebné tieto výsledky výraznejšie potlačiť, teda znížiť ich váhu na celkovom výsledku, aby ich prínos bol relevantný. Ďalej predpokladáme, že pridaním váhy prvému, či druhému modulu, resp. nájdením vhodného pomeru váh medzi dvomi hlavnými modulmi, sa dostaneme k optimálnemu nastaveniu. Toto nastavenie by malo korektne vyhodnocovať legitímne aplikácie a v dostatočnej miere správnosti vyhodnocovať aj vzorky malvéru.

Pri hľadaní optimálneho nastavenia sme vychádzali z testov systému s vyššie uvedenými štyrmi základnými nastaveniami. Ich kombináciou a následnými experimentami s ich váhovaním sme sa postupne prepracovali až k podľa nás najvhodnejšiemu nastaveniu váh jednotlivých modulov. Toto nastavenie má váhy jednotlivých modulov "10-30-1-1" a dosiahnuté výsledky sú v tabuľke 4.9.

Väčšina moderných systémov na detekciu mobilného malvéru používa oveľa sofistikovanejšie riešenia, resp. využíva kombináciu viacerých postupov. Na našich výsledkoch vidíme, že aj pri použití veľmi jednoduchých metód môžeme dosiahnuť pomerne presné výsledky. 47.61% vzoriek malvéru náš systém s vysokou presnosťou (skóre vyššie ako 0.6) určil ako malvér. Ak sa pozrieme do tabuľky 4.10, kde sme skupinu malvéru so skóre od 0.4 do 0.599 predelili na hranici 0.5. Tu vidíme, že už vyše 60% vzoriek malvéru bolo správne určených ako malvér. Nastavenie systému by sme vedeli

Tabuľka 4.10: Podrobnejší pohľad na "stredný" riadok v tabuľke 4.9

Skóre	Počet vzoriek mal.	%	Počet vzoriek leg.	%
$< 0; 0.2)$	47	3.63	431	49.54
$< 0.2; 0.4)$	268	20.71	281	32.30
$< 0.4; 0.5)$	198	15.30	64	7.36
$< 0.5; 0.6)$	165	12.75	29	3.33
$< 0.6; 0.8)$	247	19.09	44	5.06
$< 0.8; 1 >$	369	28.52	21	2.41
Celkovo	1294	100	870	100

upraviť tak, aby toto percento bolo ešte vyššie, ale tým by sme zhoršili úspešnosť správneho rozpoznania legitímnych aplikácií.

Podobne ako v prípade malvéru, vzorky neškodných aplikácií boli analyzované rovnakým systémom s rovnakými nastaveniami, a rozdelené do zadefinovaných rozsahov podľa dosiahnutého hodnotenia, viď tabuľka 4.9. Naše riešenie správne určilo 81.84% legitímnych aplikácií ako neškodné, pokiaľ nahliadneme do tabuľky 4.10, toto percento vzrastie na 89.2%. Toto finálne nastavenie 10-30-1-1 dosiahlo pri testoch najlepší pomer vzoriek malvéru označených ako nebezpečné a vzoriek legitímnych aplikácií označených ako neškodné. V našej databáze však existuje niekoľko vzoriek - či už malvéru, alebo legitímnych aplikácií, ktoré boli z nejakého dôvodu chybne vyhodnotené, t.j. malvér ako bezpečná aplikácia a legitímna aplikácia ako malvér. Na tieto prípady sa pozrieme v nasledujúcich podkapitolách.

4.3.1 Vyhodnotenie chyby prvého druhu

Viac ako 60% vzoriek malvéru sa nášmu systému podarilo korektne označiť ako malvér, ak zoberieme do úvahy aj všetky vzorky so skóre z intervalu $< 0.4; 0.6)$, tak toto číslo narastie na 75.66%. Stále však v našej množine ostáva vyše 300 aplikácií, ktoré nevieme správne označiť ako malvér. Z tohoto dôvodu sme sa rozhodli bližšie preskúmať skupinu vzoriek s hodnotením z intervalu $< 0; 0.2)$. Táto množina obsahuje 47 vzoriek, resp. viacero verzií jednej aplikácie. Tieto chybne ohodnotené vzorky tvoria tzv. chybu prvého druhu. Chybu prvého druhu sme si pri návrhu definovali ako situáciu, kedy je malvér chybne označený ako legitímna aplikácia, resp. jeho číselné ohodnotenie nie je dostatočne vysoké na to, aby bol takto označený. Rozhodli sme sa preto zistiť, čo je príčinou tejto situácie a nájsť spôsob, ako ju eliminovať. Tieto aplikácie sme rozdelili do skupín podľa príbuznosti, kategórie, rodiny malvéru či požadovaných povolení:

- Skupina "zábavných" aplikácií - Mosquito Repellent (4 vzorky), Easy Button (2), Airhorn (2), Whoopee (2) - všetky vzorky vyžadujú len tri povolenia: INTERNET, GET_ACCOUNTS a READ_PHONE_STATE. Na kradnutie informácií o zariadení a používateľovi to aplikáciám postačuje. Na detekciu je však tento po-

čet príliš malý a nevytvorí sa dostatočný počet nebezpečných skupín povolení, takže výsledné skóre bude nízke.

- Skupina aplikácií na prezeranie erotického obsahu - Hot Girls 3, Beauty Breasts, Beauty Girl, Beauty Leg - podobne ako vzorky vyššie, aj táto rodina požaduje len tri povolenia - INTERNET, READ_PHONE_STATE a SET_WALLPAPER. Dôsledkom je ako vyššie nedostatočne vysoké skóre a preto sú tieto aplikácie označené ako neškodné, čo na prvý pohľad aj vyzerajú.

- Skupina repacknutých hier (Car Mechanic Simulator, Dokkan Battle, Drag Toilet Paper, Minishot Basketball) - skupina hier z jednej rodiny malvéru, požadujúca len prístup k dátovému úložisku, sieťovým pripojeniam, informáciám o zariadení a v prípade poslednej aj k vibráciám. Znovu pri analýze nedosiahnu potrebnú hodnotu skóre, aby mohli byť označené ako malvér.

- Repacknuté hry Falldown (3) a Basketball Shot (4) - obe vyžadujú prístup k internetu a práci s dátovým úložiskom. Okrem nich požadujú ešte aj ACCESS _NETWORK_STATE, READ_PHONE_STATE, resp. VIBRATE a WAKE_LOCK. Oproti prípadom vyššie, tu vidíme kombináciu nebezpečných povolení pre kategóriu Games v kombinácii s bezpečnými povoleniami ako VIBRATE či "relatívne bezpečnými" ako je READ_PHONE_STATE. Znova tak dochádza k zníženiu výsledného skóre a vyhnutie sa detekcii.

- Repacknutý Download Manager (3) - systémová aplikácia, ktorá bola súčasťou nadstavby pre zariadenie Samsung Galaxy S3. V tomto prípade aplikácia síce požadovala viaceré povolenia na prácu so sieťami alebo dátovým úložiskom, ale bola zaradená do správnej kategórie (Tools). Preto tieto povolenia spadali do menej nebezpečných kategórií (zelená a žltá), a následne aj výsledné skóre bolo výrazne nižšie.

- Repacknutá aplikácia na obnovu dát Recovery Deluxe - pri tejto aplikácii je zaujímavé, že podľa názvu by mala vedieť pristupovať k dátovému úložisku, ale vyžaduje len povolenia INTERNET, GET_ACCOUNTS a ACCESS_NETWORK_STATE. Toto by malo používateľa upozorniť, že aplikácia vyžaduje úplne iné povolenia, ako by podľa názvu mala. Podobne ako vyššie, počet povolení je na detekciu príliš malý, takže výsledné skóre bude znova nízke. Na zisťovanie a odosielanie informácií o používateľovi to však aplikácii postačuje.

- Repacknuté aplikácie na správu zariadenia (Super App Manager, App Installer Manager) - tieto aplikácie vyžadujú podobné povolenia ako repacknuté hry Falldown a Basketball Shot. Znova ide o kombináciu povolení na kradnutie používateľských dát a neškodných povolení ako VIBRATE na sťaženie prípadnej analýzy a zamaskovanie svojej činnosti.

- Repacknutá aplikácia Google Play - malvér z rodiny Basebridge maskujúci sa za regulárnu systémovú aplikáciu. Okrem prístupu k dátovému úložisku a informáciám o zariadení nevyžaduje žiadne ďalšie povolenia. Žiadané povolenia by boli

v prípade legitímnej aplikácie v poriadku, keďže ide o aplikáciu primárne zameranú na sťahovanie obsahu a iných aplikácií. Z tohoto dôvodu je aj výsledné skóre nízke.

- Aplikácia vibrujúca po ukončení hovoru Call End Vibrate - ako názov aplikácie hovorí, požaduje povolenie na ovládanie vibrácií zariadenia, k tomu aj informácie o stave telefónu a či je nabootovaný OS. Povolenie na prístup k Internetu sa medzi legitímnymi povoleniami "stratí" a tým sa znemožní jej detekcia.

- Aplikácia na úpravu fotografií Quick Photo Grid - multimediálna aplikácia na spracovanie fotografií. Podobne ako vyššie, aj ona vyžaduje prístup na Internet, ktoré nepostačuje medzi legitímnymi povoleniami na dosiahnutie dostatočne vysokého skóre.

- Skupina rôznych "unlockerov" k hre Angry Birds (4) - ako ich názov napovedá, jedná sa o rôzne aplikácie slúžiace na odomykanie obsahu v populárnej sérii mobilných hier Angry Birds. Funkcie na odomykanie úrovní, postavičiek či kostýmov sa v tejto sérii vyskytujú ako súčasť aplikácií. Našlo sa však viacero verzií (hlavne na ruských a čínskych úložiskách), kde bola táto funkcionalita prerobená do samostatných aplikácií. Tieto aplikácie požadujú viac ako 20 rôznych povolení, vrátane prístupu ku kontaktom, SMS správam a pod. V dobe ich vydania bolo platenie SMS správami za obsah pomerne bežné, preto boli často repackované. Napriek jasne viditeľnej nebezpečnosti tieto aplikácie náš systém nevie korektne zaradiť, keďže nebezpečných povolení je medzi ostatnými príliš málo.

- Klient služby Netflix a zľavová aplikácia CN Coupon - dve aplikácie obsahovo a funkcionálne veľmi rozdielne, ale požadovanými povoleniami veľmi podobné. Obe vyžadujú prístup k Internetu, resp. Wi-Fi, k lokalizačným službám, či vykonávaniu finančných operácií. Tieto povolenia však patria v rámci týchto kategórií k oprávneným, resp. nie výrazne podozrivým. Navyše, nevytvorí sa dostatočné množstvo nebezpečných skupín povolení, takže ich detekcia je znova neúspešná.

- Revolution Day - aplikácia slúžiaca ako organizér, navyše pripomínajúca výročia rôznych revolúcií. Všetky jej požadované povolenia patria medzi oprávnené pre túto kategóriu, či už ide o ukladanie poznámok na dátové úložisko, alebo prístup k Internetu slúžiaci na sťahovanie informácií k výročiu zvolenej revolúcie. Zo šiestich potrebných povolení len dve a ich kombinácia môže byť považovaná za hrozbu pre používateľa.

- Repacknutá hra Happy Lord - vyžaduje prístup na Internet a k dátovému úložisku. Hoci sú niektoré kombinácie povolení pre túto kategóriu nebezpečné, nepostačuje to na dosiahnutie dostatočne vysokého skóre, lebo ostatné (menej podozrivé) kombinácie skóre znížia na príliš nízku hodnotu.

- Cestovateľská aplikácia Wan Yue Apartment - slúžiaca primárne na rezerváciu ubytovania a komunikáciu s ubytovacím zariadením prostredníctvom SMS správ.

Podľa dostupných informácií bola súčasťou botnetu, čo by vysvetľovalo požiadavky na povolenia WRITE_CALL_LOGS a WRITE_CONTACTS, ktoré sú pre túto kategóriu neobvyklé. Aplikácia sa len tesne nedostala do vyššie ohodnotenej kategórie, pri prísnejších nastaveniach už figurovala v skupine $< 0.2; 0.4)$.

- Skupina aplikácií s rôznym zameraním - Light Grid, Love Meter, Happy New Year a repacknutá hra Robodefense - tieto aplikácie vyžadujú prístup k Internetu, k dátovému úložisku a informáciám o zariadení. Pri analýze sme si všimli, že tieto aplikácie požadujú aj dve povolenia, ktoré nemáme v našej databáze. Jedná sa o MODIFY_PHONE_STATE a WRITE_APN_SETTINGS. Tieto povolenia sú vyhradené pre systémové aplikácie, a teda by sa do bežných aplikácií tretích strán vôbec nemali dostať. Ich využitie, resp. zneužitie je možné len na "rootnutom" zariadení, navyše musí byť aplikácia označená ako systémová a uložená v priečinku pre systémové aplikácie. Pri skúmaní týchto vzoriek sme nenašli žiadne indície, ktoré by nasvedčovali ich príslušnosť k systémovým aplikáciám, takže môže ísť len o aplikácie, ktoré takto budú označené v budúcnosti, resp. až po získaní prístupu k "rootnutému" zariadeniu. V našom systéme však absencia týchto povolení spôsobila vytvorenie skupín povolení s nízkymi hodnotami skóre a preto bola aj celková detekcia neúspešná.

Podľa našich očakávaní majú najväčšie zastúpenie aplikácie zo skupín Games, Entertainment a Tools (viď tabuľka 4.11), ktoré sú aj podľa výskumu popísaného v Kapitole 2 najčastejšie zneužívanými kategóriami. Z podrobnejšej analýzy vyplýva, že neúspešná detekcia malvéru sa môže udiať v dvoch prípadoch. V prvom aplikácia požaduje príliš málo povolení, ktoré nie sú výrazne nebezpečné pre jej kategóriu, čo zníži skóre a navyše sa nevytvorí potrebné množstvo dostatočne početných skupín povolení, čo tiež zníži výsledné skóre. V druhom prípade sa nebezpečné povolenia "zamiešajú" medzi menej nebezpečné a nastane opačná situácia, keď sa vytvorí príliš veľa skupín povolení. Nebezpečné skupiny sú tak prevýšené bezpečnými a skóre nedosiahne potrebnú výšku. Tieto nedostatky je možné odstrániť zvýšením váhy druhého hlavného analyzačného modulu *SimpleApplicationAnalyzer*, čo by však malo negatívny dopad na úspešnosť správneho posúdenia legitímnych aplikácií. Ďalším spôsobom by bolo pridanie modulu na dynamickú analýzu aplikácií, kde by mohli byť sledované ďalšie charakteristiky aplikácií ako napr. sieťová komunikácia.

4.3.2 Vyhodnotenie chyby druhého druhu

Takmer 90% legitímnych aplikácií bolo správne vyhodnotených ako neškodných, teda používatelia boli uistení, že pri ich používaní im hrozí len minimálne, resp. žiadne riziko. Zvyšných 10% naše riešenie označuje ako podozrivé, či priamo nebezpečné, čím sa dostávame z tzv. chybe druhého druhu. Chybu druhého druhu sme si na začiatku predošlej kapitoly definovali ako prípad, keď je legitímna aplikácia nesprávne označená ako malvér. Táto chyba je dôležitým parametrom pri hodnotení presnosti riešenia. V absolútnych číslach je vyjadrená v tabuľke 4.9 ako počet legitímnych aplikácií s hodnotením z rozsahu $< 0.8; 1 >$, resp. $< 0.6; 0.8)$. V rozsahu $< 0.6; 0.8)$ sa nachádza

Tabuľka 4.11: Počet vzoriek malvéru určených ako legitímne aplikácie vzhľadom na kategóriu

Kategória	Počet vzoriek v kategórii	Počet vzoriek určených ako legitímne	%
Sports	0	0	0
Health	6	0	0
Lifestyle	23	0	0
Games	358	17	4.75
Travel	4	1	25
Multimedia	322	2	0.62
Entertainment	110	15	13.63
Social	6	0	0
Tools	276	8	2.90
Communication	19	0	0
Shopping	2	1	50
Productivity	7	1	14.29
Themes	58	1	1.72
Finance	2	1	50
Nezaradené	101	0	0

44 aplikácií, ale len 7 z nich má skóre vyššie ako 0.7. Podrobnejšie sme preto rozobrali aplikácie spadajúce do vyššieho rozsahu, kde sa nachádza 21 aplikácií, resp. viacero verzií niekoľkých aplikácií. Konkrétne sa jedná o tieto:

- Facebook aplikácia, zoznamka Badoo, Facebook Messenger (4 verzie) a Google Hangouts - aplikácie k sociálnym sieťam vyžadujúce prístup ku kontaktom, pripojeniu na Internet, lokalizačným službám, fotografiám a iným multimédiám nachádzajúcim sa na zariadení.

- Google Aplikácia (5 verzií) - univerzálna Google aplikácia, okrem mailového a chatového klienta vie pracovať aj s multimédiami a rôznymi systémovými nastaveniami zariadenia.

- Hra 2048 - jednoduchá hra, ktorá v tejto verzii vyžaduje prístup k Internetu či kontaktom za účelom zdieľania skóre medzi používateľmi, pozývanie ďalších ľudí k hraniu alebo na odomykanie ďalších možností hry. V neskorších verziách bola väčšina z týchto povolení a funkcionalít odstránená.

- Samsung Health, Gear Fit Plugin - vyžadujú prístup k lokalizačným službám, či Internetu, hlavne kvôli ukladaniu dát na externej lokalite. Kvôli svojmu zameraniu pristupujú k väčšiemu množstvu citlivých dát, napr. zdravotným záznamom, ktoré môžu byť po odcudzení zneužité na poškodenie používateľa.

- Nástroje na správu zariadenia - AirDroid, SYNCit, LinkSharing (2 verzie) - tieto aplikácie slúžia na zlepšenie používateľského komfortu, prepájanie s inými zaria-

Tabuľka 4.12: Počet legitímnych aplikácií z každej kategórie chybne určených ako malvér

Kategória	Počet vzoriek v kategórii	Počet vzoriek určených ako malvér	%
Sports	18	0	0
Health	3	1	33.33
Lifestyle	34	2	5.88
Games	87	1	1.15
Travel	70	0	0
Multimedia	107	0	0
Entertainment	14	0	0
Social	95	1	1.05
Tools	259	11	4.25
Communication	64	5	7.81
Shopping	24	0	0
Productivity	68	0	0
Themes	5	0	0
Finance	22	0	0

deniami (napr. s TV) cez Bluetooth alebo po sieti. Dokážu tiež meniť nastavenia sieťových pripojení a zdieľať pomocou nich aj multimediálne dáta.

- ESET Mobile Security & Antivirus - antivírový nástroj, vyžaduje na svoje korektné fungovanie prístup k širokému spektru systémových nastavení, prístup k multimédiám či externému pamäťovému médiu.

Tieto aplikácie spadali do kategórií Tools (Google, SYNCit, ESET antivírus atď.), Communication (Facebook Messenger a Google Hangouts), Lifestyle (zoznamka Badoo) a po jednej z Games, Health a Social, viď tabuľka 4.12. Tieto aplikácie skutočne potrebujú veľké množstvo povolení na svoje korektné fungovanie (s možnou výnimkou hry 2048), sú správne zaradené do kategórií, kde sú tieto povolenia označené ako menej podozrivé a pochádzajú od overených vývojárov. Napriek tomu boli na základe vyššej podobnosti s malvérom vyhodnotené ako potenciálne nebezpečné. Práve kvôli takýmto "falošným poplachom" sme sa rozhodli implementovať možnosť ohodnotenia aplikácií používateľmi, viď kapitola 3. Pokiaľ používatelia týmto aplikáciám dôverujú, majú možnosť vyjadriť túto dôveru svojim hodnotením a tak poskytnúť spätnú väzbu nášmu hodnotiacemu algoritmu. O testovaní používateľského hodnotenia pojednáva podkapitola 4.5.

4.4 Výskyt povolení v malvéri a legitímnych aplikáciách

V ďalšej časti vyhodnocovania výsledkov analýzy sme sa pozreli na najčastejšie sa vyskytujúce povolenia, resp. ich dvojice a trojice v malvéri a legitímnych aplikáciách.

Táto oblasť je zaujímavá z toho dôvodu, že z týchto výsledkov môžeme pomerne presne určiť vektory útoku škodlivých aplikácií. Napr. prístup k Internetu a fotografiám či citlivým dokumentom môže viesť k ich úniku a následnému vydieraniu poškodenej osoby. Na základe týchto výsledkov je možné určiť kritické oblasti na zariadeniach (práca s SMS správami, zoznam kontaktov či interné úložisko dát), ktorým treba venovať zvýšenú pozornosť pri návrhu a implementácii aplikácie. Z databázy malvéru sa nám podarilo dekompilovať a zanalyzovať všetkých 1294 vzoriek. Podobne aj pri legitímnych aplikáciách sa nám podarilo spracovať všetky vzorky, tým rozdielom, že aspoň jedno povolenie vyžaduje len 575 vzoriek z 870.

4.4.1 Singles

V prvom kroku tejto fázy testovania sme sa zamerali na výskyt jednotlivých povolení (singles) v aplikáciách z našej databázy. Výsledky sú prehľadne spracované v tabuľkách 4.13 a 4.14. Pri bližšom pohľade na tabuľku 4.13 vidíme, že skoro každá vzorka malvéru vyžaduje pripojenie na Internet a povolenie READ_PHONE_STATE, ktorým je možné získať rôzne údaje o zariadení a jeho používateľovi (IMEI číslo, telefónne číslo), ako aj meniť správanie sa aplikácie (napr. stlmenie zvukov pri prichádzajúcom hovore). Tieto výsledky nemusia byť na prvý pohľad podozrivé, ale keď ich dáme do kontextu s kategóriami týchto aplikácií, viď tabuľka 4.3, resp. ich zoznamom, zistíme, že veľké ich veľká časť by tieto povolenia logicky na svoje korektné fungovanie nemala vyžadovať.

Veľkej časti aplikácií z kategórií Entertainment, Multimedia či Tools vôbec netreba pripojenie na Internet, keďže by sa po správnosti malo jednať o aplikácie určené na fungovanie v offline režime. Ďalej viac ako 60% týchto vzoriek vyžaduje prístup k všeobecným sieťovým nastaveniam (ACCESS_NETWORK_STATE) a nastaveniam Wi-Fi sietí (ACCESS_WIFI_STATE, CHANGE_WIFI_STATE). Tieto povolenia môžu spôsobiť samovoľné pripájanie sa zariadenia na rôzne nezabezpečené siete a spôsobovať tak únik rôznych informácií o požívateľovi, resp. jeho zariadení. Pokiaľ sa vyššie spomenuté povolenia skombinujú s ďalším v tabuľke, WRITE_EXTERNAL_STORAGE, ktoré umožňuje zápis dát na pamäťovú kartu (napr. rôzne logy), môže nastať situácia, kedy škodlivá aplikácia zapisuje dáta na pamäťovú kartu a odosiela ich na riadiaci server útočníka.

Ďalším vektorom možného útoku by mohlo byť zneužívanie povolení na prácu s SMS správami (povolenia READ_SMS, WRITE_SMS, SEND_SMS, RECEIVE_SMS). K nim ešte môžeme pridať aj čítanie kontaktov pomocou READ_CONTACTS. Tento útok spočíva v odosielaní správ na prémiové čísla za poplatok výrazne vyšší ako za štandardnú textovú správu a následné odchytávanie a mazanie potvrdzovacích správ pred tým, ako ich zaregistruje používateľ. Takmer 40% vzoriek z našej databázy požaduje aspoň minimálne možnosti práce s SMS správami, čo je znova v rozpore s našimi očakávaniami, keďže v zozname vzoriek máme len niekoľko aplikácií, ktoré by mali na svoje korektné fungovanie tieto povolenia vyžadovať. Jedná sa o aplikácie z

Tabuľka 4.13: 20 najčastejších povolení v malvéri

Názov	Počet v malvéri	%
INTERNET	1270	98.22
READ_PHONE_STATE	1210	93.58
ACCESS_NETWORK_STATE	1049	81.13
WRITE_EXTERNAL_STORAGE	866	66.98
ACCESS_WIFI_STATE	825	63.81
READ_SMS	812	62.80
RECEIVE_BOOT_COMPLETED	729	54.83
WRITE_SMS	682	52.75
SEND_SMS	574	44.39
RECEIVE_SMS	509	39.37
VIBRATE	498	38.52
ACCESS_COARSE_LOCATION	484	37.43
READ_CONTACTS	473	36.58
CALL_PHONE	434	33.57
WAKE_LOCK	434	33.57
ACCESS_FINE_LOCATION	431	33.33
CHANGE_WIFI_STATE	412	31.86
WRITE_CONTACTS	387	29.93
RESTART_PACKAGES	341	26.37
DISABLE_KEYGUARD	256	19.80

kategórií Communication, Social, Finance či Lifestyle.

Tretím významným vektorom útoku môže byť sledovanie používateľa prostredníctvom lokalizačných povolení (ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION). Tieto povolenia vyžaduje minimálne tretina vzoriek, ktoré sme testovali. Spolu s prístupom k Internetu môže byť sledovaná poloha zariadenia a teda aj jeho používateľa. To by mohlo viesť buď k prepadu používateľa a jeho okradnutia, resp. fyzickej ujmy, alebo naopak k vykradnutiu jeho obydlia, keďže útočník by vedel, že používateľ nie je doma.

Posledným zaujímavým povolením je DISABLE_KEYGUARD. Týmto povolením je možné zrušiť zamykanie obrazovky zariadenia. Vzdialeným zavolaním škodlivej aplikácie tak môže útočník odomknúť zariadenie a získať nad ním kontrolu.

Pri legitímnych aplikáciách bola tabuľka výsledkov 4.14 veľmi podobná predošlej, čo sa týka druhov povolení. Podstatný rozdiel je však v ich percentuálnom zastúpení v kombinácii so zohľadnením kategórií do ktorých tieto aplikácie spadajú, viď tabuľka 4.4. Avšak do výslednej tabuľky 4.14 sme zahrnuli len tie vzorky, ktoré vyžadujú aspoň jedno povolenie, teda počty v zátvorkách v tabuľke 4.4. V tejto časti sa pri

legitímnych aplikáciách budeme opierať o tieto počty.

Podobne ako pri malvéri, aj tu je najviac zastúpená potreba prístupu k Internetu v zmysle potreby povolení INTERNET, ACCESS_NETWORK_STATE či ACCESS_WIFI_STATE. Vzhľadom na tabuľku 4.4 je toto percentuálne zastúpenie pomerne presné, keď zrátame počty aplikácií v kategóriách nutne vyžadujúcich prístup k Internetu na ich korektné fungovanie. Významne zastúpené sú povolenia na prácu s externým pamäťovým médiom - WRITE_EXTERNAL_STORAGE, čo je však zaujímavé, pri legitímnych aplikáciách sa vo väčšej miere vyskytuje aj párové povolenie READ_EXTERNAL_STORAGE. Toto by korešpondovalo s multimediálnymi aplikáciami na prehrávanie hudby či prezeranie fotografií. Taktiež ich môžu využívať aplikácie na správu súborov, komunikačné aplikácie či aplikácie z kategórie Productivity.

Prvým výrazným rozdielom oproti malvéru je používanie povolení na prihlasovanie sa k používateľským účtom, overovanie identity používateľa a správu používateľských účtov - povolenia GET_ACCOUNTS, USE_CREDENTIALS, MANAGE_ACCOUNTS a AUTHENTICATE_ACCOUNTS. Predpokladáme, že je to spôsobené nárastom počtu aplikácií z kategórií Shopping, Social, Finance, Lifestyle a Communication, ktoré všetky využívajú aspoň jedno z nich (prípadne kombináciu) na zabezpečenie používateľského konta. Zastúpenie povolení lokalizačných služieb ACCESS_FINE_LOCATION a ACCESS_COARSE_LOCATION sú v očakávanej početnosti zodpovedajúcej súčtu kategórií Travel, Social a Shopping. Rovnako aj v prípade READ_CONTACTS, využívané kategóriami Communication, Social či Lifestyle.

Povolenie READ_PHONE_STATE môže byť problematické, ale jeho využitie predpokladáme pri aplikáciách na správu zariadení (Tools) a aj pri Shopping či Finance, kde informácie o zariadení môžu byť použité ako druh hardvérového tokenu používaného pri prihlasovaní sa k používateľskému účtu. Prístup k fotoaparátu (CAMERA) vyžadujú aplikácie na fotografovanie a úpravu fotografií (Multimedia), sociálne siete či Lifestylové aplikácie (napr. zoznamka Tinder).

Poslednou zaujímavou dvojicou sú WRITE_SYNC_SETTINGS a READ_SYNC_SETTINGS. Tieto povolenia sa používajú pri synchronizácii webového obsahu, najčastejšie vo forme REST služby. Príkladom môže byť obnovovanie (refresh) mailovej aplikácie či obsahu sociálnej siete (FaceBook). Môžeme si všimnúť, že v tabuľke chýbajú povolenia na prácu s SMS správami. Je to spôsobené hlavne tým, že povolenie na prácu s nimi má len veľmi málo legitímnych aplikácií (WhatsApp), keďže táto funkcionalita je priamo zabudovaná v operačnom systéme. Navyše, tzv. in-app nákupy sa vo väčšine legitímnych prípadov riešia pomocou zabezpečenej služby a nie pomocou prémiových SMS správ.

Celkový percentuálny rozdiel u desiatich vybraných povolení ilustruje tabuľka 4.15. Dobre sú tu vidieť rozdiely v miere využívania niektorých kritických povolení medzi

Tabuľka 4.14: 20 najčastejších povolení v legitímnych aplikáciách

Názov	Počet v legit.	%
INTERNET	350	60.87
ACCESS_NETWORK_STATE	347	60.35
WRITE_EXTERNAL_STORAGE	307	53.39
WAKE_LOCK	296	51.48
READ_EXTERNAL_STORAGE	269	46.78
VIBRATE	244	42.43
GET_ACCOUNTS	217	37.74
ACCESS_FINE_LOCATION	213	37.04
ACCESS_WIFI_STATE	195	33.91
RECEIVE_BOOT_COMPLETED	190	33.04
READ_CONTACTS	186	32.35
ACCESS_COARSE_LOCATION	180	31.30
READ_PHONE_STATE	154	26.78
USE_CREDENTIALS	149	25.91
CAMERA	144	25.04
MANAGE_ACCOUNTS	144	25.04
BLUETOOTH	113	19.65
WRITE_SYNC_SETTINGS	106	18.43
READ_SYNC_SETTINGS	105	18.26
AUTHENTICATE_ACCOUNTS	99	17.22

Tabuľka 4.15: Porovnanie výskytov vybraných 10 single povolení

Názov	% v malvéri	% v legit.	Rozdiel v %
READ_PHONE_STATE	93.58	26.78	66.80
READ_SMS	62.80	7.30	55.50
WRITE_SMS	52.75	1.91	50.84
INTERNET	98.22	60.87	37.35
SEND_SMS	44.49	11.30	33.19
ACCESS_WIFI_STATE	63.81	33.91	29.90
RECEIVE_SMS	39.37	14.09	25.28
RECEIVE_BOOT_COMPLETED	54.83	33.04	21.79
ACCESS_NETWORK_STATE	81.13	60.35	20.78
WRITE_EXTERNAL_STORAGE	66.98	53.39	13.59

malvérom a legitímnymi aplikáciami. Najmarkantnejšie rozdiely sú viditeľné pri získavaní informácií o zariadení (READ_PHONE_STATE) s rozdielom až takmer 67%. Podobne sú na tom aj povolenia na čítanie a písanie SMS správ s vyše 50%-ným rozdielom. Menej výrazným, ale stále vysokým rozdielom medzi legitímnymi aplikáciami a malvérom je aj potreba internetového pripojenia (vyše 37%), či pripájanie sa k Wi-Fi sieťam (takmer 30%). Na druhej strane, povolenie k zápisu na externé médium požaduje takmer rovnaké percento aplikácií z oboch skupín (rozdiel len 13.6%).

4.4.2 Dvojice

Podobne ako samostatné povolenia sme spracovali aj najčastejšie dvojice povolení v malvéri a legitímnych aplikáciách. Výsledky pre malvér sme prehľadne spracovali do tabuľky 4.16. Skoro polovica dvojíc je kombináciou povolenia INTERNET a niektorého z ďalších. Pri vzorkách malvéru to s veľkou pravdepodobnosťou môže znamenať posielanie rôznych dát von zo zariadenia. Jedná sa napríklad o kombinácie s READ_PHONE_STATE (únik informácií o zariadení a používateľovi), READ_SMS (kradnutie používateľových SMS správ) či ACCESS_COARSE_LOCATION (informovanie o polohe používateľa). Podobnú situáciu vidíme aj v prípade ACCESS_NETWORK_STATE a ACCESS_WIFI_STATE, kde aplikácie môžu byť schopné vykonávať rovnakú podvratnú činnosť po bezdrôtovej sieti.

Zaujímavé sú aj kombinácie s povolením RECEIVE_BOOT_COMPLETED, teda oznámením o úspešnom spustení operačného systému. Nie sú až tak početné, ale môžu slúžiť na informovanie útočníka, že zariadenie sa úspešne zaplo a môže vydať pokyny malvéru na vykonanie špecifickej úlohy. Tesne mimo tabuľky skončili dvojice povolení na prácu s SMS správami, READ_SMS, SEND_SMS a RECEIVE_SMS, SEND_SMS v počte 471 resp. 462 výskytov. Toto nám znovu ukazuje, že vyše tretina malvéru je schopná práce s SMS správami, hoci by tieto aplikácie podľa správnosti takúto funkcionalitu vôbec nemali mať. Na vytváranie spamových sietí a botnetov môže slúžiť

Tabuľka 4.16: 20 najčastejších dvojíc povolení v malvéri

Dvojica	Počet v malvéri	%
INTERNET, READ_PHONE_STATE	1175	90.80
ACCESS_NET_STATE, INTERNET	1040	80.37
ACCESS_NET_STATE, READ_PHONE_STATE	1010	78.05
ACCESS_WIFI_STATE, INTERNET	825	63.76
ACCESS_WIFI_STATE, READ_PHONE_STATE	804	62.13
INTERNET, READ_SMS	796	61.51
ACCESS_NET_STATE, ACCESS_WIFI_STATE	784	60.59
READ_PHONE_STATE, READ_SMS	783	60.51
ACCESS_NET_STATE, READ_SMS	730	56.41
INTERNET, RECEIVE_BOOT_COMP	696	53.79
READ_PHONE_STATE, RECEIVE_BOOT_COMP	685	52.94
ACCESS_NET_STATE, RECEIVE_BOOT_COMP	650	50.23
INTERNET, SEND_SMS	551	42.58
ACCESS_WIFI_STATE, RECEIVE_BOOT_COMP	543	41.96
READ_PHONE_STATE, SEND_SMS	543	41.96
ACCESS_WIFI_STATE, READ_SMS	538	41.58
INTERNET, RECEIVE_SMS	501	38.72
READ_PHONE_STATE, RECEIVE_SMS	482	37.25
ACCESS_COARSE_LOCATION, INTERNET	477	36.86
INTERNET, VIBRATE	476	36.76

kombinácia INTERNET, READ_CONTACTS, kde je možnosť ukradnutia zoznamu kontaktov a jeho následné využitie na rozposielanie spamu pomocou SMS správ, emailu alebo iných kontaktov. Dvojica READ_PHONE_STATE, CALL_PHONE v počte výskytov 432 zasa môže slúžiť na telefonické hovory na rôzne čísla. Ako v predošlom prípade, aj tu je možnosť jej využitia na zapojenie zariadenia do botnetu alebo na volania na prémiové čísla patriace útočníkovi.

Štatistiky dvojíc povolení v legitímnych aplikáciách sme spracovali do tabuľky 4.17. Podobne ako pri jednotlivých povoleniach, aj tabuľka dvojíc je obsahovo veľmi podobná tabuľke s malvérom. Znova preto musíme brať do úvahy kontext aplikácií, teda kategórie, do ktorých sú svojou funkcionalitou zaradené. Prvé tri dvojice, ACCESS_NETWORK_STATE, INTERNET; ACCESS_NETWORK_STATE, WAKE_LOCK; INTERNET, WAKE_LOCK; ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE a ACCESS_WIFI_STATE, INTERNET početnosťami zodpovedajú počtom aplikácií zaradeným do kategórií Social, Communication, Travel, Lifestyle a patrí sem aj niekoľko nástrojov - Tools a online multimediálnych aplikácií - Multimedia.

Početnosť dvojíc INTERNET, READ_EXTERNAL_STORAGE a AC-

CESS_NETWORK_STATE, READ_EXTERNAL_STORAGE korešponduje súčtu kategórí Social, Multimedia a Communication, kde sa stará o sťahovanie multimediálneho obsahu, hlavne hudby. Pri Social a Communication ide navyše aj o upload, hlavne fotografií, na sociálne siete. READ_EXTERNAL_STORAGE, WAKE_LOCK zasa zabraňuje hlavne pri prehrávaní multimédií automatickému vypínaniu obrazovky zariadenia. Podobnú funkcionalitu vyžadujú aj aplikácie z kategórie Travel, keďže v skoro všetkých vzorkách sú do určitej miery zakomponované aj navigačné služby. Dvojice ACCESS_NETWORK_STATE, GET_ACCOUNTS; GET_ACCOUNTS, INTERNET a GET_ACCOUNTS, WAKE_LOCK sú najčastejšie využívané v aplikáciách, kde je potrebné overenie používateľa, resp. jeho konta - Social, Communication, Finance a Shopping.

Poslednou zaujímavou skupinou v tabuľke sú ACCESS_FINE_LOCATION, ACCESS_NETWORK_STATE; ACCESS_FINE_LOCATION, INTERNET a ACCESS_FINE_LOCATION, WAKE_LOCK sú okrem navigačných aplikácií z kategórie Travel aj niektorými aplikáciami z kategórií Shopping, Lifestyle, Sports a Social. Ich využitie spočíva napr. vo vyhľadávaní obchodov (Shopping), hľadaní partnerov v zoznamkách (Tinder z kategórie Lifestyle), zaznamenávaní zabehnutej vzdialenosti (Endomondo zo Sports) či populárne označovanie seba alebo priateľov na rôznych miestach a následné zverejnenie na sociálnych sieťach (FaceBook). Do tabuľky sa tesne (186 výskytov) nezmestili dvojice ACCESS_NETWORK_STATE, READ_CONTACTS a INTERNET, READ_CONTACTS využívané komunikačnými aplikáciami, sociálnymi sieťami a Lifestyle aplikáciami (vyššie spomenutý Tinder).

Rozdiely vo využívaní niekoľkých vybraných kritických dvojíc povolení sa nachádzajú v tabuľke 4.18. Možnosť úniku informácií o zariadení (INTERNET, READ_PHONE_STATE - rozdiel vyše 64%), úniku SMS správ (INTERNET, READ_SMS - rozdiel vyše 54%) a všeobecne manipulácie s nimi (READ_PHONE_STATE, READ_SMS a ACCESS_NETWORK_STATE, READ_SMS - rozdiely 50% a viac) sú oproti legitímnym aplikáciám v malvéri vysoko zastúpené. Podobne aj pri samostatných povoleniach, aj tu nájdeme výnimku - rozdiel vo využívaní povolení pracujúcich s polohou používateľa (ACCESS_COARSE_LOCATION, INTERNET) je len minimálny.

4.4.3 Trojice

V poslednej časti tejto testovacej fázy sme spracovali najčastejšie sa vyskytujúce trojice povolení aj v malvéri - tabuľka 4.19 aj v legitímnych aplikáciách 4.20. Ako prvé sme spracovali vzorky malvéru. Dosiahnuté výsledky vo veľkej miere kopírujú výsledky dosiahnuté pri dvojiciach a samostatných povoleniach. Na prvých miestach s vysokými početnosťami sa znova objavujú kombinácie povolení pracujúce s prístupom k Internetu resp. iným sieťam (ACCESS_NETWORK_STATE, INTERNET, ACCESS_WIFI_STATE) spolu s povolením READ_PHONE_STATE. Potvrdzuje sa nám tak možnosť exfiltrácie používateľských dát smerom k útočníkovi. Podobne môžeme

Tabuľka 4.17: 20 najčastejších dvojíc povolení v legitímnych aplikáciách

Dvojica	Počet v legit.	%
ACCESS_NET_STATE, INTERNET	342	59.49
ACCESS_NET_STATE, WAKE_LOCK	274	47.65
INTERNET, WAKE_LOCK	274	47.65
INTERNET, READ_EXT_STO	265	46.09
ACCESS_NET_STATE, READ_EXT_STO	262	45.57
ACCESS_NET_STATE, VIBRATE	240	41.74
INTERNET, VIBRATE	240	41.74
READ_EXT_STO, WAKE_LOCK	237	41.22
VIBRATE, WAKE_LOCK	219	38.09
ACCESS_NET_STATE, GET_ACCOUNTS	216	37.57
GET_ACCOUNTS, INTERNET	216	37.57
ACCESS_FINE_LOC, ACCESS_NET_STATE	213	37.04
ACCESS_FINE_LOC, INTERNET	212	36.87
ACCESS_NET_STATE, ACCESS_WIFI_STATE	195	33.91
ACCESS_WIFI_STATE, INTERNET	195	33.91
GET_ACCOUNTS, WAKE_LOCK	195	33.91
READ_EXT_STO, VIBRATE	194	33.74
ACCESS_FINE_LOC, WAKE_LOCK	191	33.22
INTERNET, RECEIVE_BOOT_COMP	189	32.87
ACCESS_NET_STATE, RECEIVE_BOOT_COMP	187	32.52

Tabuľka 4.18: Porovnanie výskytov vybraných 10 dvojíc povolení

Dvojica	% v mal.	% v leg.	Rozdiel v %
INTERNET, READ_PHONE_STATE	90.80	26.78	64.02
INTERNET, READ_SMS	61.51	7.30	54.21
READ_PHONE_STATE, READ_SMS	60.51	7.13	53.38
ACCESS_NET_STATE, READ_SMS	56.41	7.30	49.11
READ_PH_STATE, REC_BOOT_COMP	52.94	18.78	34.16
INTERNET, SEND_SMS	42.58	11.30	31.28
READ_SMS, SEND_SMS	36.40	6.96	29.44
RECEIVE_SMS, SEND_SMS	35.70	10.61	25.09
ACCESS_NET_STATE, INTERNET	80.37	59.49	20.88
ACCESS_COARSE_LOC, INTERNET	36.86	31.13	5.73

vidieť, že aj povolenie na čítanie SMS správ - READ_SMS - sa vo vysokom počte vyskytuje s týmito povoleniami, čím sa nám znova potvrdzuje možnosť čítania SMS správ a ich možného kradnutia útočníkom.

Častá prítomnosť povolenia RECEIVE_BOOT_COMPLETED môže, podobne ako je vyššie uvedené, signalizovať malvéru, že sa operačný systém úspešne spustil a malvér môže začať vykonávať svoju činnosť. Neoprávnená práca s SMS správami sa potvrdila prítomnosťou povolení SEND_SMS a RECEIVE_SMS vo vyše tretine vzoriek malvéru. Do tabuľky sa tesne (467 výskytov) nezmestila trojica READ_PHONE_STATE, READ_SMS, SEND_SMS a tiež aj INTERNET, SEND_SMS, RECEIVE_SMS (460 výskytov) a potvrdzujú neoprávnenú manipuláciu a prístup k SMS správam. Podobne ako pri dvojiciach, aj tu predpokladáme, že niektoré vzorky malvéru majú možnosť odchytávať potvrdzovacie SMS správy z prémiových čísiel a vykonávať svoju činnosť bez povšimnutia používateľa. Trojice INTERNET, READ_CONTACTS, READ_PHONE_STATE (456) a ACCESS_COARSE_LOCATION, INTERNET, READ_PHONE_STATE (453) môžu byť zneužité na vytváranie botnetov z ukradnutých zoznamov kontaktov, resp. na sledovanie používateľa, keďže útočník má k dispozícii jeho polohu spárovanú s jeho zariadením.

Tabuľka 4.19: 20 najčastejších trojíc povolení v malvéri

Trojica	Počet v malvéri	%
ACCESS_NETWORK_STATE, INTERNET, READ_PHONE_STATE	1010	78.05
ACCESS_WIFI_STATE, INTERNET, READ_PHONE_STATE	804	62.13
ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, INTERNET	784	60.59
INTERNET, READ_PHONE_STATE, READ_SMS	781	60.36
ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, READ_PHONE_STATE	772	59.66
ACCESS_NETWORK_STATE, INTERNET, READ_SMS	730	56.41
ACCESS_NETWORK_STATE, READ_PHONE_STATE, READ_SMS	716	55.33
INTERNET, READ_PHONE_STATE, RECEIVE_BOOT_COMPLETED	685	52.94
ACCESS_NETWORK_STATE, INTERNET, RECEIVE_BOOT_COMPLETED	650	50.23
ACCESS_NETWORK_STATE, READ_PHONE_STATE, RECEIVE_BOOT_COMP	643	49.69
ACCESS_WIFI_STATE, INTERNET, RECEIVE_BOOT_COMPLETED	543	41.96
ACCESS_WIFI_STATE, READ_PHONE_STATE, RECEIVE_BOOT_COMP	542	41.89
INTERNET, READ_PHONE_STATE, SEND_SMS	541	41.81
ACCESS_WIFI_STATE, INTERNET, READ_SMS	538	41.58
ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, READ_SMS	534	41.27
ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, RECEIVE_BOOT_COMP	531	41.04
ACCESS_WIFI_STATE, READ_PHONE_STATE, READ_SMS	530	40.96
INTERNET, READ_PHONE_STATE, RECEIVE_SMS	480	37.09
INTERNET, READ_SMS, SEND_SMS	469	36.24
INTERNET, READ_PHONE_STATE, VIBRATE	468	36.17

Výskyt trojíc povolení v legitímnych aplikáciách sme prehľadne spracovali do tabuľky 4.20. Podobne ako pri dvojiciach, obsahovo je tabuľka podobná tabuľke s trojicami v malvéri. Pokiaľ však zoberieme do úvahy kontext aplikácií, teda ich zaradenie do príslušných kategórií. Povolenia ACCESS_NETWORK_STATE, INTERNET, ACCESS_WIFI_STATE sa najčastejšie vyskytujú v trojiciach s povoleniami WAKE_LOCK a READ_EXTERNAL_STORAGE. Kombinácia s povolením WAKE_LOCK početnosťou korešponduje s počtom aplikácií zaradenými do kategórií Social, Communication, Travel, Lifestyle a patrí sem aj niekoľko nástrojov - Tools a online multimediálnych aplikácií - Multimedia. Pri týchto aplikáciách je nevyhnutné, aby sa nevypínala obrazovka zariadenia počas behu aplikácie - napr. pri streamovaní videa, navigácii a pod.

Aplikácie vyžadujúce kombináciu prvých troch povolení a READ_EXTERNAL_STORAGE sú zaradené do kategórií Multimedia (streamovanie hudby, práca s fotografiami), Travel (práca s offline mapami) a Social (upload fotografií na sociálne siete). Spolu s týmito povoleniami sa často vyskytuje aj povolenie GET_ACCOUNTS, slúžiace na správu používateľských účtov. Podobne ako pri dvojiciach aj tu sa jedná o aplikácie z kategórií Social, Communication, Finance a Shopping. GET_ACCOUNTS sa vyskytuje aj v kombinácii s povolením READ_CONATCTS buď s povolením INTERNET a ACCESS_NETWORK_STATE (obe 170 výskytov), ktoré sa do tabuľky nevošli. Tu vidíme správny príklad použitia povolení na prístup k používateľskému účtu, výberu kontaktu zo zoznamu kontaktov a následnej komunikácii pomocou povolenia INTERNET alebo ACCESS_NETWORK_STATE.

Tabuľka 4.20: 20 najčastejších trojíc povolení v legitímnych aplikáciách

Trojica	Počet v legit.	%
ACCESS_NETWORK_STATE, INTERNET, WAKE_LOCK	274	47.65
ACCESS_NETWORK_STATE, INTERNET, READ_EXT_STORAGE	262	45.57
ACCESS_NETWORK_STATE, INTERNET, VIBRATE	240	41.74
ACCESS_NET_STATE, READ_EXT_STORAGE, WAKE_LOCK	237	41.22
INTERNET, READ_EXTERNAL_STORAGE, WAKE_LOCK	237	41.22
ACCESS_NETWORK_STATE, VIBRATE, WAKE_LOCK	219	38.09
INTERNET, VIBRATE, WAKE_LOCK	219	38.09
ACCESS_NETWORK_STATE, GET_ACCOUNTS, INTERNET	216	37.57
ACCESS_FINE_LOCATION, ACCESS_NET_STATE, INTERNET	212	36.87
ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, INTERNET	195	33.91
ACCESS_NETWORK_STATE, GET_ACCOUNTS, WAKE_LOCK	195	33.91
GET_ACCOUNTS, INTERNET, WAKE_LOCK	195	33.91
ACCESS_NETWORK_STATE, READ_EXT_STORAGE, VIBRATE	194	33.74
INTERNET, READ_EXTERNAL_STORAGE, VIBRATE	194	33.74
ACCESS_FINE_LOCATION, ACCESS_NET_STATE, WAKE_LOCK	191	33.22
ACCESS_FINE_LOCATION, INTERNET, WAKE_LOCK	191	33.22
ACCESS_NET_STATE, INTERNET, RECEIVE_BOOT_COMPLETED	187	32.52
ACCESS_NETWORK_STATE, INTERNET, READ_CONTACTS	186	32.35
READ_EXTERNAL_STORAGE, VIBRATE, WAKE_LOCK	184	32.00
ACCESS_FINE_LOCATION, ACCESS_NET_STATE, READ_EXT_STORAGE	181	31.45
ACCESS_FINE_LOCATION, INTERNET, READ_EXT_STORAGE	181	31.45

Tabuľka 4.21: Porovnanie výskytov vybraných 10 trojíc povolení

Názov	% v mal.	% v leg.	Rozdiel v %
INTERNET, READ_PH_ST, READ_SMS	60.36	7.13	53.23
ACC_NET_ST, INT, READ_PH_ST	78.05	26.78	51.27
ACC_WIFI_ST, INT, READ_PH_ST	62.13	22.43	39.70
ACC_WIFI_ST, READ_PH_ST, READ_SMS	40.96	7.13	33.83
INT, READ_PH_ST, SEND_SMS	41.81	11.30	30.51
READ_PH_ST, READ_SMS, SEND_SMS	36.09	6.96	29.13
INT, SEND_SMS, RECEIVE_SMS	35.55	10.61	24.94
INT, READ_PH_ST, RECEIVE_SMS	37.09	13.74	23.35
INT, READ_CONT, READ_PH_ST	35.24	18.61	16.63
ACC_COARSE_LOC, INT, READ_PH_ST	35	21.22	13.78

Podobne ako v predošlých prípadoch, aj pre porovnanie miery využitia vybraných kritických trojíc povolení sme vytvorili tabuľku 4.21. Najväčšie rozdiel medzi malvérom a legitímnymi aplikáciami boli znova v možnosti úniku informácií o zariadení (INTERNET, READ_PHONE_STATE, READ_SMS), SMS správ (ACCESS_NETWORK_STATE, INTERNET, READ_PHONE_STATE) s rozdielom vyše 50% medzi týmito množinami aplikácií.

Z vyššie prezentovaných výsledkov vyplýva, že na budovanie pravidiel detekcie malvéru je možné použiť kombinácie vybraných povolení a určenia aplikácie. Určité kombinácie povolení by mali zvyšovať riziko (multiplikatívne), iné ho znižujú. Znalostný systém najprv zaradí aplikáciu do skupín, nastaví základné riziko a potom ho na základe pravidiel koriguje. Zaradenie do skupín a samotné pravidlá je však ťažké určiť mechanicky. Môžu nám však pomôcť samotní používatelia, čomu sa venujeme v nasledovnej časti.

4.5 Vyhodnotenie používateľského vstupu

V poslednej časti testovania sme sa zamerali na výsledky získané zapojením používateľov do hodnotiaceho procesu. Používateľmi bolo otestovaných 51 rôznych aplikácií (za rôzne aplikácie považujeme aj rôzne verzie jednej aplikácie) s celkovo 78 rôznymi hodnoteniami. Testovanie aplikácií vykonávali študenti našej fakulty na dobrovoľnej báze. Tí, ktorí hodnotenie vyplnili sa riadili našou farebnou stupnicou, číselne vyjadrenou od 1 do 5, kde číslo 1 reprezentuje zelenú farbu a teda najnižšiu podobnosť s malvérom a číslo 5 reprezentuje červenú farbu a teda najvyššiu podobnosť s malvérom. Výsledky testovania sú spracované v tabuľke 4.22.

V 35 prípadoch sa hodnotenie aplikácie používateľom a našim riešením zhodovalo, resp. líšilo sa len vo veľmi malej miere. Do tejto skupiny spadali aplikácie na oboch stranách hodnotenia, či už išlo o neškodné aplikácie ako Notes, katalóg LIDL, prehrávače hudby ÓČKO a Spotify a niekoľko aplikácií na internet banking. Medzi potenciálne nebez-

pečné aplikácie používatelia správne zaradili napr. rané verzie Facebook-ovej aplikácie, FB Messenger-a či aplikáciu na zdieľanie jázd Uber. V týchto prípadoch používatelia podľa nás správne rozoznali potenciálnu hrozbu, resp. neškodnosť testovanej aplikácie.

V 11 prípadoch bol rozdiel medzi naším a používateľským hodnotením dva stupne, čo už môže signalizovať neistotu používateľa, alebo naopak úpravu nášho hodnotenia. Pri aplikáciách WhatsApp a CCleaner používatelia upravili naše hodnotenie o dva stupne nižšie, keďže boli presvedčení o bezpečnosti týchto aplikácií. Pri aplikáciách vyžadujúcich viacero povolení naopak používatelia upravili ich hodnotenie v priemere až o dva stupne vyššie. Sem patria napr. aplikácia Facebook, tabuľkový procesor Sheets resp. jeho verzia Tabuľky, komunikátor Skype či aplikácia pre mikro sociálnu sieť Twitter. Pri týchto aplikáciách môžeme zvážiť upravenie váh nášho algoritmu tak, aby v neho mali používatelia vyššiu dôveru. Zaujímavosťou je zaradenie aplikácie na tímovú spoluprácu Slack do tejto kategórie, a tiež aj zaradenie našej aplikácie FEIDroid, čo považujeme za určitú formu recesie od študentov.

Poslednú kategóriu tvorí 5 prípadov, kde je odchýlka medzi hodnotením používateľov a naším vyššia ako tri stupne. Pri aplikáciách Skype a FB Messenger išlo o výrazné zníženie úrovne možného rizika, čo môže mať pri neopatrnom používateľovi závažný dopad na jeho súkromie a bezpečnosť. Opačným extrémom bolo výrazné zvýšenie úrovne možného rizika pre jednu bankovú aplikáciu, avšak hodnotenie jej sesterskej aplikácie spadalo do prvej kategórie, takže mohlo ísť o preklep, alebo prehnanú opatrnosť. Ďalej sem patrí antivírová aplikácia od ESET-u a testovacia aplikácia COHave od kolegov z našej fakulty. Tieto dva príklady nepokladáme za relevantne vyplnené odpovede, lebo s veľkou pravdepodovnosťou ide znova o recesiu zadávateľov hodnotenia.

Celkovo môžeme zhodnotiť túto časť testovania pozitívne, lebo väčšina používateľských hodnotení korelovala s našimi výsledkami, čo svedčí o pomerne dobrom povedomí našich študentov o bezpečnosti mobilných aplikácií. Na druhej strane môžu byť tieto výsledky skreslené, lebo študenti informatických smerov majú výrazne lepšie povedomie o IT bezpečnosti ako iní študenti, resp. bežní používatelia.

Pri akomkoľvek "sociálnom" učení je však nutné počítať s naschvál nesprávne zadaným hodnotením, viď vyššie. Táto metóda má však veľký potenciál pri systémoch s veľkým počtom používateľov (Google Play), kde škodlivé a nesprávne hodnotenia používateľov vieme štatisticky potlačiť.

Tabuľka 4.22: Používateľské hodnotenie

Názov	Ver.	# použ. hodn.	Pr. použ. hodn.	Naše hodn.
Adobe Acrobat	15.0.2	1	1	1
CCleaner	1.15.57	1	3	5
COHave	2.0.1	2	4,5	1

Disk	2.1.495.10.34	1	2	1
ESET MS&A	3.0.1318.0	1	5	1
Facebook	4.3.2	3	3,67	2
Facebook	77.0.0.20.66	1	5	3
Facebook	79.0.0.18.71	1	5	4
Facebook	81.0.0.22.70	1	1	2
FB Messenger	30.0.0.33.174	3	2,33	2
FB Messenger	71.0.0.10.65	2	2,5	3
FB Messenger	72.0.0.16.67	2	4,5	5
FB Messenger	73.0.0.15.70	2	2	5
FEIDroid	3.0	5	3	1
Fotky	1.21.0.123444480	1	3	2
Gmail	6.5.123664905	1	1	1
HERE Maps	1.1.10331	1	1	2
Instagram	8.1.0	1	2	2
Kalendar	5.5.2-121651413	1	2	1
LIDL	1.0.29(253)	1	1	1
Meniny a mena	4.07	1	1	1
MHDcka	1.0	1	1	1
Notes	1.5.6	1	1	1
ÓČKO	5.1.3	1	1	1
OK Timer	1.1	1	5	3
Platby Spor.	3.4.0	1	5	1
Sheets	1.4.392.08.35	1	4	2
Skype	6.3.0.2	1	4	2
Skype	7.01.0.669	1	1	4
Slack	2.11.0	4	2,5	1
Snake	1.2	1	2	1
Spotify	2.8.0.1081	1	1	1
Spotify	5.4.0.858	4	2	1
Spotify	5.7.0.781	1	1	1
SV Pismo	1.6.2	1	1	1
Tabulky	1.4.392.08.35	1	2	2
Tabulky	1.6.192.08.35	2	3	1
TatraBanka	2.6.0	1	1	1
Team Viewer	11.0.4905	1	2	2
TIME Mobile	1.6	1	2	1
Total Cmd	2.72	1	1	1
Twitter	5.55.0	2	4	2
Uber	4.154.8	1	5	4
Ucty Spor.	3.4.0	2	1	2

VLC	1.3.2	3	2	1
VUB Mobil Bank	2.3.1	2	1	1
Waze	4.2.0.1	2	2,5	2
Waze	4.3.0.2	2	2,5	2
WhatsApp	2.16.95	1	1	3
World Clock	2.0	1	1	2
Zuno SK	1.13	1	1	1

4.6 Záverečné zhodnotenie

Na záver testovania nášho riešenia by sme radi zhodnotili dosiahnuté výsledky. Rýchlosť analýzy aplikácií (časť 4.1) je z pohľadu používateľa limitovaná len rýchlosťou jeho mobilného pripojenia. V priemere sme dosahovali čas v rozmedzí 2.5 až 6 sekúnd na stiahnutie 1MB dát aplikácií. Rýchlosť samotnej analýzy aplikácií je podmienená veľkosťou aplikácií a ich zložitosťou, resp. množstvom zdrojového kódu, ktorý treba prejsť. V tejto oblasti sme dosiahli časy v rozmedzí 2 až 11 sekúnd. Celkovo tak naše riešenie dosahuje časy od zhruba 5 do 17 sekúnd od momentu zadania požiadavky používateľa na analýzu aplikácie až po jej zobrazenie sa na displeji mobilného zariadenia. Výhodu tu majú rôzne veľké úložiská aplikácií (Google Play), ktoré väčšinu tohto času, resp. aplikácií vedia rozložiť medzi veľké množstvo používateľov.

Presnosť výsledkov analýzy (časť 4.3) výrazne závisí od aktuálneho nastavenia váh jednotlivých čiastkových hodnotiacich algoritmov. Sme si vedomí skutočnosti, že naše riešenie je postavené na veľmi jednoduchom princípe, no napriek tomu dosahuje veľmi kvalitné výsledky na danej testovacej vzorke. Viac ako 60% vzoriek malvéru sa nášmu systému podarilo korektne označiť ako malvér, ak zoberieme do úvahy aj všetky vzorky so skóre z intervalu $< 0.4; 0.6)$, tak toto číslo narastie na 75.66%. Tieto výsledky by sme vedeli zlepšiť sprísnením hodnotenia, čo by však malo za následok zhoršenie úspešnosti správneho posúdenia legitímnych aplikácií. Ďalšou možnosťou by bolo pridanie modulu na dynamickú analýzu aplikácií, kde by mohli byť sledované ďalšie charakteristiky aplikácií ako napr. sieťová komunikácia.

Pri legitímnych aplikáciách naše riešenie dosahuje takmer 90%-nú úspešnosť správneho posúdenia legitímnych aplikácií. Pri niekoľkých aplikáciách sa vyskytli odchýlky, tieto však boli spôsobené veľkým počtom vyžadovaných povolení, čo však pri aktuálnom nastavení algoritmu nevieme lepšie odfiltrovať.

Výskyt povolení v našich vzorkách (časť 4.4) sme zmapovali na všetkých vzorkách aplikácií, ktoré sme mali k dispozícií. Celkovo sa jednalo o takmer 2300 vzoriek, z ktorých približne 1900 vyžadovalo aspoň jedno povolenie na svoje korektné fungovanie. Dosiahnuté výsledky do veľkej miery korešpondujú s výsledkami publikovanými v minulosti, napr. [102], [103] či [104]. Navyše podporujú správnosť nášho prístupu

a uvažovania nad jednoduchým spôsobom detekcie (aj potenciálne) nebezpečných aplikácií. Jednoduchým porovnaním výskytu jednotlivých povolení a ich skupín vieme vo veľkej miere určiť, či sa jedná o podozrivú aplikáciu alebo nie. Navyše je tento spôsob jednoduchý na pochopenie aj pre len minimálne technicky zdatného používateľa.

Používateľský vstup (časť 4.5) sme sa rozhodli zapracovať do nášho riešenia na základe výsledkov z kapitoly 4.3. Môže sa totiž stať, že aj napriek dôkladnému testovaniu nastavení hodnotiaceho algoritmu, bude naše riešenie vyhodnocovať niektoré legitímne aplikácie ako podozrivé a naopak, škodlivé ako neškodné. Preto sme sa rozhodli dať používateľom možnosť vyjadriť svoj názor na toto hodnotenie a takýmto spôsobom ho korigovať. Vo viacerých prípadoch bolo naše hodnotenie korigované používateľmi, v závislosti od typu a charakteru aplikácie.

Záver

Výskum v oblasti informačnej bezpečnosti sa čoraz viac posúva do oblasti mobilných zariadení. Je to hlavne z dôvodu ich neustáleho rozširovania sa a objavovania nových spôsobov ako tieto zariadenia zneužiť na rôznu podvratnú činnosť.

Našu prácu sme rozdelili na dve hlavné časti, teoretickú a implementačnú. V teoretickej časti, v podkapitolách 1.2 a 1.3 sme zmapovali a popísali najdôležitejšie komponenty a súčasti OS Android. Na tento popis sme nadviazali podrobným prehľadom bezpečnostných mechanizmov a opatrení, ktoré boli postupne do systému pridávané v neskorších verziách OS Android - podkapitoly 1.4 až 1.9. Tieto zmeny a opatrenia logicky nemohli byť implementované hneď, ale až po prejavení sa chýb v systéme alebo zverejnení konkrétnej zraniteľnosti. V podkapitole 1.10 sme na porovnanie pripravili popis dvoch najrozšírenejších upravených verzií OS Android s dôrazom na bezpečnosť a ochranu dát a súkromia používateľa.

Na popis bezpečnostných mechanizmov OS Android sme nadviazali v kapitole 2, pojednávajúcej o aktuálnom výskume na tejto mobilnej platforme. V tejto kapitole sme popísali široké spektrum hrozieb, útokov a zraniteľností vyskytujúcich sa v minulosti - na starších verziách OS Android - alebo aj doteraz pretrvávajúcich. K týmto nedostatkom sme popísali aj možné spôsoby, ako ich eliminovať úplne, resp. ich riziko znížiť na minimum.

Na základe poznatkov z teoretickej časti sme si zadefinovali základné rámcové tézy nášho výskumu, ktoré sme priebežne upravovali, vzhľadom na rýchly vývoj v oblasti:

- **Model povolení závislý na rolách.** Základným problémom platformy Android bola v minulosti nemožnosť výberu povolení, ktoré sú pridelené aplikácii pri inštalácii. Toto často spôsobuje tzv. privilege escalation útoky, pri ktorých dochádza k úniku citlivých dát. Jedným z riešení bol systém Apex na dynamické prideľovanie povolení. Ako sme však zistili, v čase začiatku prác na dizertačnom práci, systém Apex bol už nefunkčný. Systém povolení však môže byť pre používateľa nezrozumiteľný a príliš náročný na konfiguráciu. Naším cieľom bolo špecifikovať model dynamických povolení, v ktorých používateľ a(alebo) aplikácia vystupuje v nejakej role. Úlohou bolo navrhnúť a implementovať vhodný systém rolí a mechanizmy, ktorými by bolo možné docieliť želané správanie. Vo verzii 6 OS Android došlo k výraznej zmene modelu povolení zo statického na dynamický.

Takýto vývoj sme predpokladali nielen my, ale aj veľká časť komunity okolo OS Android. Týmto krokom Google ako majiteľ Androidu vyriešil jeden z hlavných a dlhotrvajúcich problémov, a s ním aj prvú z našich navrhovaných téz.

- **Detekcia malvéru.** Špecifikovaním a implementovaním modelu rolí by bolo možné lepšie detegovať niektoré typy malvéru a zabrániť niektorým typom útokov. Preto sme sa rozhodli skúmať aj prispôsobenie modelu rolí na detekciu čo najširšieho spektra malvéru.

- **Súvisiace bezpečnostné mechanizmy.** Na zabezpečenie nového systému povolení by bolo potrebné upraviť, resp. zaviesť nové bezpečnostné mechanizmy. Konkrétne sme uvažovali o zásahoch do spôsobu prihlasovania sa do zariadenia, kryptografickej ochrany až po prípadné zásahy do jadra OS. Tejto téze sme sa venovali v rámci vedenia resp. konzultovania diplomových prác popísaných v časti 2.1.1. Sumárne výsledky sme publikovali v článku [63], preto považujeme túto tézu za úspešne splnenú.

Ako sa postupne vyvíjala situácia bezpečnosti OS Android, rozhodli sme sa rozvinúť tézu detekcie malvéru resp. potenciálne nebezpečných aplikácií na základe nie modelu rolí, ale statickej analýzy požadovaných povolení a analýzy zdrojového kódu aplikácie, ktorá tvorí implementačnú časť práce.

Aktuálny výskum ukazuje, že žiadna metóda na detekciu škodlivých aplikácií nie je stopercentne úspešná. Pri navrhovaní čo najefektívnejšieho spôsobu detekcie sme sa inšpirovali predošlým výskumom vykonanom medzi študentami [112]. Rozhodli sme sa zapojiť do procesu detekcie malvéru aj používateľov, keďže detekčné systémy často vyhodnotia škodlivé aplikácie ako bezpečné a naopak. Za cieľ sme si stanovili prepojiť detekčný systém s klientskou aplikáciou, kde si budú môcť používatelia svoje nainštalované aplikácie skontrolovať a pridať aj vlastné hodnotenie, ktoré zasa poslúži nám na zlepšenie fungovania a presnosti detekcie celého systému.

Tretia kapitola našej práce preto zahŕňa podrobný popis implementačnej fázy práce. Naše riešenie je v nej popísané od úplného začiatku (podkapitola 3.1), sú tu vytýčené pojmy a definície, s ktorými ďalej pracujeme v návrhu a implementačnej fáze. Ďalej sme tu zadefinovali požiadavky na celkovú funkcionalitu riešenia a kritériá na posúdenie výslednej realizácie. Podkapitola 3.2 pojednáva o návrhu všetkých častí nášho riešenia. To sme rozdelili na tri moduly:

- **Klientská aplikácia:** slúži primárne na zobrazovanie výsledkov analýzy aplikácií používateľom. Má jednoduchý dizajn, je prehľadná a spĺňa všetky požiadavky na používateľský komfort. Jej funkcionalitou je zber informácií o aplikácii, ktorú sa používateľ rozhodol otestovať, odoslanie týchto informácií na server prostredníctvom webovej služby a následné zobrazenie výsledku analýzy používateľovi. Okrem toho môže pomocou nej používateľ odoslať aj vlastné hodnotenie konkrétnej aplikácie.

- **Webová služba:** slúži ako zabezpečený komunikačný kanál medzi klientskou aplikáciou a serverom. Dáta priamo nespracováva, len ich prenáša vo formáte JSON. Komunikácia je zabezpečená je pomocou protokolu HTTPS.
- **Serverová časť:** tu sa nachádza kompletná výpočtová logika nášho riešenia. Od podmodulov na sťahovanie a dekompiláciu inštalačných *.apk* súborov, cez ich ukladanie do databázy, až po samotné vykonanie analýzy. Navyše je spojená aj s administrátorským rozhraním vo forme webovej stránky.

Podkapitola 3.3 podrobne rozvádza technické a implementačné detaily vyššie uvedených modulov. Separátne sme popísali moduly serverovej časti a klientskej aplikácie. Popis webovej služby sme zahrnuli vo zvyšných dvoch častiach, lebo v každej časti je jej implementácia špecifická a lepšie tak zapadá do celkového popisu riešenia. Vo všeobecnosti popis modulov zahŕňa:

- **Popis hlavných tried:** podrobný popis hlavných tried a komponentov, ktoré tvoria jadro funkcionality jednotlivých modulov. Obsahuje aj konkrétne príklady volaní metód, či popis tabuliek v databáze.
- **Použité algoritmy:** podrobný popis najdôležitejších použitých algoritmov, napr. algoritmu na analýzu aplikácií, proces aktualizácie databázy, či proces zadávania požívateľského hodnotenia.
- **UML diagramy:** slúžia na vizuálne objasnenie prepojenia a náväznosti tried, algoritmickej logiky, či vzťahov jednotlivých entít v databáze.
- **Popis použitých technológií:** krátke predstavenie použitých technológií a objasnenie dôvodov na ich použitie.
- **Grafický návrh:** grafické návrhy (mockupy) častí riešenia, ktoré obsahujú aj grafické používateľské rozhranie. Okrem návrhov sme použili aj ukážky finálneho vizuálu niektorých častí riešenia.

V štvrtej kapitole práce sa venujeme testovaniu nášho riešenia. Zamerali sme sa ako na presnosť riešenia, tak aj na používateľský komfort. Okrem toho sme nadviazali na existujúce publikácie, napr. [102], [103] či [104] a preskúmali distribúciu povolení v dostupných vzorkách aplikácií. V poslednej časti sme preskúmali aj používateľské povedomie o bezpečnosti mobilných zariadení formou možnosti zadania vlastného hodnotenia k zvolenej aplikácii:

- **Rýchlosť analýzy:** čo sa týka rýchlosti vyhodnotenia rizika zvolenej aplikácie, používateľ je limitovaný rýchlosťou svojho pripojenia k Internetu, ale len v prípade, že musí posielať inštalačný *.apk* súbor na server. V našich podmienkach sme dosahovali priemerné rýchlosti analýzy od 5 do 17 sekúnd - od momentu zadania požiadavky používateľa na analýzu aplikácie až po jej zobrazenie sa na displeji mobilného zariadenia - v závislosti od veľkosti a zložitosti aplikácie. Výhodou veľkých úložísk aplikácií ako napr. Google Play je, že analýzu veľkého počtu aplikácií vedia rozložiť medzi veľké množstvo svojich používateľov.

- **Presnosť analýzy:** podrobným testovaním sme našli nastavenie váh čiastkových hodnotiacich algoritmov, ktorého výstupy majú najnižšie chyby prvého a druhého druhu. Napriek postaveniu nášho riešenia na jednoduchom princípe sme dosiahli veľmi kvalitné výsledky na dostupnej testovacej vzorke. Viac ako 60% vzoriek malvéru sa nášmu systému podarilo korektne označiť ako malvér, ak zoberieme do úvahy aj všetky vzorky so skóre z intervalu $< 0.4; 0.6)$, tak toto číslo narastie na 75.66%. Tieto výsledky by sme vedeli zlepšiť zmenou váh jednotlivých analytických modulov, čo by však malo za následok zhoršenie úspešnosti správneho posúdenia legitímnych aplikácií. Ďalšou možnosťou by bolo napr. pridanie modulu na dynamickú analýzu aplikácií, kde by mohli byť sledované ďalšie charakteristiky aplikácií ako napr. sieťová komunikácia. Pri legitímnych aplikáciách naše riešenie dosahuje takmer 90%-nú úspešnosť správneho posúdenia legitímnych aplikácií. Pri niekoľkých aplikáciách sa vyskytli odchýlky, tieto však boli spôsobené veľkým počtom vyžadovaných povolení, čo však pri aktuálnom nastavení algoritmu nevieme lepšie odfiltrovať

- **Distribúcia povolení:** rozhodli sme sa nadviazať na vyššie spomenuté články a vykonať podobne zamerané testy. Výskyt povolení sme zmapovali na všetkých dostupných vzorkách a do veľkej miery korešpondujú s výsledkami publikovanými v týchto prácach. Takisto podporujú správnosť našej metodiky testovania nainštalovaných aplikácií a následnej detekcie (aj potenciálneho) nebezpečenstva. Jednoduchým porovnaním výskytu jednotlivých povolení a ich skupín vieme vo veľkej miere určiť, či sa jedná o podozrivú aplikáciu alebo nie. Navyše je tento spôsob jednoduchý na pochopenie aj pre len minimálne technicky zdatného používateľa.

- **Používateľské hodnotenia:** analýzou výsledkov z testovania presnosti analýzy aplikácií sme zistili, že aj pri najlepšom možnom nastavení hodnotiaceho algoritmu sa objavia nebezpečné aplikácie označené ako bezpečné a naopak. Na základe týchto zistení sme sa rozhodli dať používateľom možnosť zadania vlastného hodnotenia na testované aplikácie. Používatelia tak prispievajú svojím hodnotení ku korekciám v hodnotiacom algoritme, a tým k zvýšenej presnosti analýzy. Vo viacerých prípadoch používatelia odhalili, že overená aplikácia bola považovaná za nebezpečnú a jej hodnotenie bolo korigované. Problémom však je, že nie všetci používatelia so systémom spolupracujú, naopak, niektorí zadávajú falošné hodnotenia. Pri dostatočne veľkej vzorke používateľov by však takéto správanie malo byť štatisticky odhaliteľné.

Tento výskum otvára možnosti na ďalšie vylepšovanie inteligentnej detekcie malvéru. Možnosti vidíme v troch hlavných smeroch:

- Prvým je vytvorenie pravidlového znalostného systému prepájajúceho jednotlivé analytické moduly.

- Druhým je zapracovanie strojového učenia (prípadne neurónových sietí alebo genetických algoritmov) do hodnotiaceho procesu. Takto by bolo možné naučiť

napr. neurónovú sieť náš rozhodovací algoritmus, navyše by sme ju mohli naučiť aj reagovať na odchýlky a rôzne anomálie, čo by znížilo počet chýb prvého a druhého druhu.

- Tretím je spresňovanie analytických modulov a pridávanie ďalších, napr. dynamickej analýzy aplikácií vo virtuálnom prostredí.

Na záver môžeme konštatovať, že sa nám podarilo zlepšiť v minulosti publikované výsledky z predošlých [114], resp. čiastkových [113] verzií tohoto projektu. Výsledné riešenie je modulárne, takže je možné ho ľubovoľne rozširovať o ďalšie moduly na analýzu aplikácií. Klientská aplikácia má jednoduchý dizajn, je používateľsky priateľská, takže práca s ňou je jednoduchá, rýchla a intuitívna. Analytické moduly sa dajú dopĺňať a postupne tak umožniť vybudovať väčší znalostný systém na detekciu škodlivých aplikácií. Plný potenciál podobného riešenia by však bolo možné uplatniť len v kontexte veľkého množstva aplikácií a používateľov, napr. integráciou s platformou Google Play.

Literatúra

[1] ANDRESON, R.: Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition, Wiley, April 2008, 1080 s., ISBN: 978-0-470-06852-6

[2] CATLIN, B. et. al.: Windows Internals, Book 1: User Mode, Microsoft Press; 7 edition (25 Oct. 2014), 900 s. ISBN: 978-0735684188

[3] WRIGHT, C. et. al.: Linux Security Modules: General Security Support for the Linux Kernel, *Proceedings of the 11th USENIX Security Symposium*, 2002, 16 s., Dostupné na internete `https://www.usenix.org/legacy/event/sec02/full_papers/wright/wright.pdf`, 20.4.2016

[4] BISHOP, M. A.: The Art and Science of Computer Security, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002, ISBN: 0201440997

[5] BIBA, K. J.: Integrity Considerations for Secure Computer Systems, MTR-3153, The Mitre Corporation, June 1975.

[6] MCLEAN, J.: Security Models, Encyclopedia of Software Engineering 2. New York: John Wiley & Sons, Inc. s. 1136–1145., 1994, DOI: 10.1002/0471028959.sof297

[7] NETMARKETSHARE: Mobile/Tablet Operating System Market Share, Dostupné na internete: `https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1`, 18.10.2017

[8] Open Handset Alliance: Industry Leaders Announce Open Platform for Mobile Devices, Dostupné na internete: `www.openhandsetalliance.com/press_110507.html`. 10.2.2013

[9] Android: Touch Devices, Dostupné na internete: `http://source.android.com/tech/input/touch-devices.html`. 10.2.2013

[10] Android: User Interface, Dostupné na internete: `http://developer.android.com/design/get-started/ui-overview.html`. 10.2.2013

[11] BRAY, T.: What Android Is, Dostupné na internete: `http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is`. 10.2.2013

[12] ELGIN, B.: Google Buys Android for Its Mobile Arsenal, Dostupné na internete: http://www.webcitation.org/5wk7sIvVb. 10.2.2013

[13] GORDON, W.: Real World Test Show That Android Task Killers Are Still Useless, Dostupné na internete: http://lifehacker.com/5862994/real-world-test-show-that-android-%20%20task-killers-are-still-useless. 10.2.2013

[14] LEYDEN, J.: Security takes a backseat on Android in update shambles, Dostupné na internete: http://www.theregister.co.uk/2011/11/22/android_patching_mess/. 28.3.2013

[15] ISAAC, M.: Android OS Hack Gives Virtual Early Upgrade, Dostupné na internete: http://www.wired.com/gadgetlab/2011/04/cyanogenmod-android/. 28.3.2013

[16] HTC: Unlock Bootloader, Dostupné na internete: http://www.htcdev.com/bootloader/. 28.3.2013

[17] eLinux.org: Android System Architecture, Dostupné na internete: http://elinux.org/images/c/c2/Android-system-architecture.jpg. 29.3.2013

[18] CHEN, J.: Android HAL Introduction: libhardware and its legacy, Dostupné na internete: http://www.slideshare.net/jollen/android-hal-introduction-libhardware-and-its-legacy. 29.2.2016

[19] GOODIN, D.: 950 million Android phones can be hijacked by malicious messages, Dostupné na internete: https://arstechnica.com/information-technology/2015/07/950-million-android-phones-can-be-hijacked-by-malicious-text-messages/. 20.10.2017

[20] ENCK, W. et. al.: A Study of Android Application Security, *Proceedings of the 20th USENIX Security Symposium (2011)*, 2011, 16 s.

[21] Android: Licenses, Dostupné na internete: http://source.android.com/source/licenses.html. 29.3.2013

[22] Android Architecture – The Key Concepts of Android OS, Dostupné na internete: http://www.android-app-market.com/android-architecture.html. 29.3.2013

[23] BRAHLER, S.: Analysis of the Android Architecture., *Studienarbeit von Stefan Brähler an der Fakultät für Informatik*, 2010, 52 s. Dostupné na internete: http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf. 29.3.2013

[24] Android Security Overview, Dostupné na internete: http://source.android.com/tech/security/index.html. 29.3.2013

[25] LEVI, J.: What is App Ops, and why did Google remove it from Android?, Dostupné na internete: http://pocketnow.com/2013/12/17/app-ops. 2.3.2016

[26] Xposed Module Repository.: AppOpsXposed, Dostupné na internete: http://repo.xposed.info/module/at.jclehner.appopsxposed. 2.3.2016

[27] CHESTER, B. - HO, J.: Encryption and Storage Performance in Android 5.0 Lollipop, Dostupné na internete: http://www.anandtech.com/show/8725/encryption-and-storage-performance-in-android-50-lollipop 9.3.2016

[28] SHRIVASTAVA, A. - MAHAJAN, P.: Android Tamer: Android Security Enhancements, Dostupné na internete: https://androidtamer.com/android-security-enhancements/. 2.3.2016

[29] Android Developers: Requesting Permissions at Run Time, Dostupné na internete: http://developer.android.com/training/permissions/requesting.html. 2.3.2016

[30] The Cheese Factory blog: Everything every Android Developer must know about new Android's Runtime Permission, Dostupné na internete: http://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en. 2.3.2016

[31] McDANIEL, P. - ENCK, W.: Not So Great Expectations: Why Application Markets Haven't Failed Security, *Security & Privacy, IEEE, Volume: 8 Issue: 5*, 2010, s. 76 - 78.

[32] ARMENDARIZ, T.: Is Google Play Safe?, Dostupné na internete: http://antivirus.about.com/od/wirelessthreats/a/Is-Google-Play-Safe.htm. 30.3.2013

[33] Android: Google Play Services, Dostupné na internete: http://developer.android.com/google/play-services/index.html. 30.3.2013

[34] KASSNER, M.: Google Play: Android's Bouncer can be pwned, Dostupné na internete: http://www.techrepublic.com/blog/security/google-play-androids-bouncer-can-be-pwned/8053. 30.3.2013

[35] HOU, O.: A Look at Google Bouncer, Dostupné na internete: http://blog.trendmicro.com/trendlabs-security-intelligence/a-look-at-google-bouncer/. 30.3.2013

[36] GORDON, S.A.: Google's security suite 'Play Protect' rolling out to Android phones, Dostupné na internete: https://www.androidauthority.com/google-play-protect-rolling-out-788614/. 17.10.2017

[37] AMADEO, R.: Android 8.0 Oreo, thoroughly reviewed, Dostupné na internete: https://arstechnica.com/gadgets/2017/09/android-8-0-oreo-thoroughly-reviewed/link. 20.10.2017

[38] CyanogenMod: About CyanogenMod, Dostupné na internete: https://wiki.cyanogenmod.org/w/About. 17.8.2016

[39] LINEAGE: LineageOS Android Distribution, Dostupné na internete: https://www.lineageos.org. 20.10.2017

[40] CyanogenMod: User Testimonials, Dostupné na internete: https://wiki.cyanogenmod.org/w/Testimonials. 17.8.2016

[41] DIAZ, J.: Your Android Phone Is Secretly Recording Everything You Do (Updated), Dostupné na internete: http://gizmodo.com/5863849/your-android-phone-is-secretly-recording-everything-you-do. 17.8.2016

[42] PATHAK, K.: How to Get the Most Out of CyanogenMods Privacy Guard Feature, Dostupné na internete: http://www.guidingtech.com/42045/cyanogenmod-privacy-guard/. 17.8.2016

[43] XIAOMI: Miui 8, Dostupné na internete: http://en.miui.com/. 17.8.2016

[44] XIAOMI: Security of Miui, Dostupné na internete: http://en.miui.com/security.php. 17.8.2016

[45] SCHMIDT, C.: All about custom ROMs for Android, Dostupné na internete https://www.androidpit.com/best-custom-roms-for-android. 23.8.2016

[46] SHABTAI, A. et. al.: Google Android: A Comprehensive Security Assessment. *Security & Privacy, IEEE, Volume: 8 Issue: 2*, 2010, s. 35 - 44.

[47] SHABTAI, A. et. al.: Google Android: A State-of-the-Art Review of Security Mechanisms. *CoRR abs/0912.5101*, 2009, 42 s. Dostupné na internete: http://tlabs.bgu.ac.il/index.php/innovation-development/android-security. 30.3.2013

[48] Android: Security Enhancements. Dostupné na internete: https://source.android.com/security/enhancements/index.html. 2.3.2016

[49] HILDENBRAND, J.: Android 7.0: Security benefits that truly matter, Dostupné na internete: http://www.androidcentral.com/how-android-n-addresses-security. 20.11.2016

[50] Android: Android Security 2016 Year in Review, Dostupné na internete: https://source.android.com/security/reports/Google_Android_Security_2016_Report_Final.pdf. 17.10.2017

[51] ROGAWAY, P.: Authenticated-encryption with associated-data, *Ninth ACM Conference on Computer and Communications Security (CCS-9)*, ACM Press, 2002, s. 98 - 107

[52] OWASP: Certificate and Public Key Pinning. Dostupné na internete: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning. 2.3.2016

[53] SHABTAI, A. et. al.: Google Android: An Updated Security Review. *Proceedings of the 2nd International Conference on Mobile Computing, Applications and Services (MobiCASE 2010)*, 2010, 14 s. Dostupné na internete: http://tlabs.bgu.ac.il/index.php/innovation-development/android-security. 30.3.2013

[54] Android: Permission List, Dostupné na internete: http://developer.android.com/reference/android/Manifest.permission.html. 30.3.2013

[55] AVIV, A.J. et. al.: Smudge Attacks on Smartphone Touch Screens, *WOOT'10 Proceedings of the 4th USENIX conference on Offensive technologies*, 2010, Article No. 1 - 7

[56] AVIV, A.J. et. al.: Practicality of Accelerometer Side Channels on Smartphones, *ACSAC '12 Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, s. 41 - 50.

[57] CAI, L. - CHEN, H.: TouchLogger: Inferring Keystrokes On Touch Screen From Smartphone Motion, *HotSec'11 Proceedings of the 6th USENIX conference on Hot topics in security*, 2011, s. 9 - 9

[58] CAI, L. - CHEN, H.: On the Practicality of Motion Based Keystroke Inference Attack, *TRUST'12 Proceedings of the 5th international conference on Trust and Trustworthy Computing*, 2012, s. 273 - 290

[59] UELLENBECK, S. et. al.: Tactile One-Time Pad: Leakage-Resilient Authentication for Smartphones, *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, 2015, s. 237 - 253

[60] ŠVANDA, D.: Autentizácia gestom do Android zariadenia pomocou akcelerometra, Diplomová práca, FEI STU, 2014, 49 s.

[61] VARCHOLA, M.: Vylepšenie autentizačného systému do Android zariadenia, Diplomová práca, FEI STU, 2015, 47 s.

[62] SHRESTHA, B. et. al.: Drone to the Rescue: Relay-Resilient Authentication using Ambient Multi-Sensing, *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, 2014, s. 349 - 364

[63] VARGA, J. - SVANDA, D. - VARCHOLA, M. - ZAJAC.P.: Authentication based on gestures with smartphone in hand, *Journal of Electrical Engineering Volume 68*, 2017, s. 256-266

[64] BOND, M. et. al.: Chip and Skim: cloning EMV cards with the pre-play attack, *Computing Research Repository (CoRR)*, 2012, 21 s.

[65] ELENKOV, N.: Emulating a PKI smart card with CyanogenMod 9.1. Dostupné na internete: http:// nelenkov.blogspot.sk/2012/10/emulating-pki-smart-card-with-cm91.html, 2012, 29.4.2016

[66] ROLAND, M. - LANGER, J.: Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless, *Presented as part of the 7th USENIX Workshop on Offensive Technologies*, 2013, 12 s.

[67] KYSEĽ, M.: Android NFC, Bakalárska práca, FEI STU, 2014, 44 s.

[68] KENWORTHY, G. - ROHATGI, P.: Mobile Device Security: The case for side channel resistance, *Cryptography Research Inc.*, 2012, 4 s.

[69] GENKIN, D. et. al.: ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels, *IACR Cryptology ePrint Archive*, 2016, 23 s.

[70] LIU, J. et. al.: uWave: Accelerometer-based personalized gesture recognition and its applications, *Pervasive and Mobile Computing archive Volume: 5 Issue: 6*, 2009, s. 657 - 675

[71] MÄNTIJÄRVI, J. et. al.: Enabling fast and effortless customisation in accelerometer based gesture interaction, *MUM '04 Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, 2004, s. 25 - 31

[72] ZHANG, X. - ACLICMEZ, O. - SEIFERT, J.-P.: A Trusted Mobile Phone Reference Architecture via Secure Kernel, *STC '07 Proceedings of the 2007 ACM workshop on Scalable trusted computing*, 2007, s. 7 - 14

[73] SHABTAI, A. - FLEDEL, Y. - ELOVICI, Y.: Securing Android-Powered Mobile Devices Using SELinux, *Security & Privacy, IEEE, Volume: 8, Issue: 3*, 2010, s. 36 - 44

[74] GILBERT, P. et. al.: Vision: Automated Security Validation of Mobile Apps at App Markets, *MCS '11 Proceedings of the second international workshop on Mobile cloud computing and services*, 2011, s. 21 - 26

[75] ZHOU, W. et. al.: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces, *CODASPY '12 Proceedings of the second ACM Conference on Data and Application Security and Privacy*, 2012, s. 317 - 326

[76] ZHOU, Y. et. al.: Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets, *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS 2012)*, 2012, 13s.

[77] MALISA, L. - KOSTIAINEN, K. - CAPKUN, S.: Detecting Mobile Application Spoofing Attacks by Leveraging User Visual Similarity Perception, *IACR Cryptology ePrint Archive*, 2015, 17 s.

[78] FERNANDES, E. et. al.: Android UI Deception Revisited: Attacks and Defenses, *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Barbados, February 2016*, 2016, 18 s.

[79] FAHL, S. et. al.: Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security, *CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, s. 50 - 61

[80] SCHRITTWIESER, S. et. al.: Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications, *Proceedings of the Network and Distributed System Security Symposium, NDSS 2012", herausgegeben von: The Internet Society; The Internet Society*, 2012, 9 s.

[81] CHIN, E. et. al.: Analyzing inter-application communication in Android, *MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011, s. 239 - 252

[82] ZHOU, Y. - JIANG, X.: Dissecting Android Malware: Characterization and Evolution, *Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012)*, 2012, 15 s.

[83] SUAREZ-TANGIL, G. et. al.: Evolution, Detection and Analysis of Malware for Smart Devices, *IEEE Communications Surveys & Tutorials (Volume:16 , Issue: 2)*, 2013, s. 961 - 987

[84] RETENAGA, A. M.: Android malware situation, *Report by Spanish National Cybersecurity Institute (Instituto Nacional de Ciberseguridad), INCIBE and HISPASEC*, 2015, 62 s.

[85] ENCK, W.: Defending Users Against Smartphone Apps: Techniques and Future Directions, *ICISS'11 Proceedings of the 7th international conference on Information Systems Security*, 2011, s. 49 - 70

[86] LIU, Y. et. al.: xShare: Supporting Impromptu Sharing of Mobile Phones, *MobiSys '09 Proceedings of the 7th international conference on Mobile systems, applications, and services*, 2009, s. 15 - 25

[87] FUCHS, A.P. - CHAUDHURI, A. - FOSTER, J.S.: SCanDroid: Automated Security Certification of Android Applications, *Technical Reports of the Computer Science Department*, 2009, 15 s.

[88] SHABTAI, A. - FLEDEL, Y. - ELOVICI, Y.: Automated Static Code Analysis for Classifying Android Applications Using Machine Learning, *Computational Intelligence and Security (CIS), 2010 International Conference on*, 2010, s. 329 - 333

[89] BURGUERA, I. - ZURUTUZA, U. - NADJM-TEHRANI, S.: Crowdroid: Behavior-Based Malware Detection System for Android, *SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, s. 15 - 26

[90] LINDORFER, M. et. al.: ANDRUBIS - 1,000,000 Apps Later: A View on Current Android Malware Behaviors, *2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2014, s. 3 - 17

[91] TAM, K. et. al.: CopperDroid: Automatic Reconstruction of Android Malware Behaviors, *22nd Annual Network and Distributed System Security Symposium, NDSS 2015 San Diego, California, USA, February 8-11, 2015*, 2015, 15 s.

[92] HEUSER, S. et. al.: DroidAuditor: Forensic Analysis of Application-Layer Privilege Escalation Attacks on Android (Short Paper), *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Barbados, February 2016*, 2016, 9 s.

[93] CARTER, P. et. al.: CuriousDroid: Automated User Interface Interaction for Android Application Analysis Sandboxes, *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Barbados, February 2016*, 2016, 17 s.

[94] ARP, D. et. al.: DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket, *21st Annual Network and Distributed System Security Symposium, NDSS 2014 San Diego, California, USA, February 23-26, 2014*, 2014, 15 s.

[95] COLETTA, A. - VAN DER VEEN, V. - MAGGI, F.: DroydSeuss: A Mobile Banking Trojan Tracker - Short Paper, *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Barbados, February 2016*, 2016, 9 s.

[96] ENCK, W. - ONGTANG, M. - MCDANIEL, P.: On Lightweight Mobile Phone Application Certification, *CCS '09 Proceedings of the 16th ACM conference on Computer and communications security*, 2009, s. 235 - 245

[97] ENCK, W. et. al.: TaintDroid: An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphones, *OSDI'10*, 2010, 15 s.

[98] MARFORIO, C. - FRANCILLON, A. - CAPKUN, S.: Application Collusion Attack on the Permission-Based Security Model and its Implications for Modern Smartphone Systems, Department of Computer Science, ETH Zurich, 2011, 16 s.

[99] NAUMAN, M. - KHAN, S. - ZHANG, X.: Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints, *ASIACCS'10*, 2010, 12 s.

[100] BERESFORD, A. R. et. al.: MockDroid: trading privacy for application functionality on smartphones, *HotMobile '11*, 2011, 6 s.

[101] GRACE, M.C. et. al.: Unsafe exposure analysis of mobile in-app advertisements, *WISEC '12 Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, 2012, s. 101 - 112

[102] FELT, A.P. et. al.: Android Permissions Demystified, *CCS'11*, 2011, 11 s.

[103] FELT, A.P. et. al.: Android Permissions: User Attention, Comprehension, and Behavior, *Symposium on Usable Privacy and Security (SOUPS) 2012*, 2012, 14 s.

[104] WEI, X. et. al.: Permission Evolution in the Android Ecosystem, *ACSAC '12*, Dec. 3-7, 2012, Orlando, Florida USA, 2012, 10 s.

[105] OZCAN, T. A. et. al.: BabelCrypt: The Universal Encryption Layer for Mobile Messaging Applications, *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, 2015, s. 355 - 369

[106] SZONGOTT, C. - BRENNER, M. - SMITH, M.: METDS - A Self-contained, Context-Based Detection System for Evil Twin Access Points, *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, 2015, s. 370-386

[107] VANRYKEL, E.: Leaky Birds: Exploiting Mobile Application Traffic for Surveillance, *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC'16), Barbados, February 2016*, 2016, 18 s.

[108] DMITRIENKO, A. et. al.: On the (In)Security of Mobile Two-Factor Authentication, *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, 2014, s. 365 - 383

[109] DAVI, L. et. al.: Privilege Escalation Attacks on Android, *ISC'10 Proceedings of the 13th international conference on Information security*, 2010, s. 346 - 360

[110] BUGIEL, S. et. al.: XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks, *Technical Report TR-2011-04*, 2011, 18 s.

[111] BUGIEL, S. et. al.: Towards Taming Privilege-Escalation Attacks on Android, *19th Annual Network & Distributed System Security Symposium (NDSS)*, 2012, 18 s.

[112] VARGA, J. - ZAJAC, P.: Mobile Security Experience of IT Students, *ELOSYS. Elektrotechnika, informatika a telekomunikácie 2012*, 2012, s. 161 - 164

[113] VARGA, J. - MUSKA, P.: Presenting risks introduced by Android application permissions in a user-friendly way, *Tatra Mt. Math. Publ. 60 (2014)*, 2014 s. 85 - 100

[114] VARGA, J. et. al.: Mitigating Possible Threats from Overprivileged Android Applications, *IN-TECH 2015: Proceedings of the International conference on innovative technologies, Dubrovnik, Croatia, 09.-11.09.2015*, 2015, s. 42 - 45

[115] W3C: Web Services Glossary, In *https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice*, 24.9.2016

[116] Oracle Corporation: What is Java?, Dostupné na internete `https://www.java.com/en/download/whatis_java.jsp`, 23.8.2016

[117] RESCORLA, E.: HTTP Over TLS, Dostupné na internete `https://tools.ietf.org/html/rfc2818`, 23.8.2016

[118] JSON: Introducing JSON, Dostupné na internete `http://www.json.org/`, 23.8.2016

[119] MySQL: About MySQL, Dostupné na internete `https://www.mysql.com/about`, 23.8.2016

[120] Let's Encrypt: About Let's Encrypt, Dostupné na internete: `https://letsencrypt.org/about/`, 10.8.2016

[121] Apktool: A tool for reverse engineering Android apk files, Dostupné na internete `https://ibotpeaches.github.io/Apktool/`, 19.8.2016

[122] Skylot.: jadx - Dex to Java decompiler, Dostupné na internete `https://github.com/skylot/jadx`, 29.10.2016

[123] Jersey - RESTful Web Services in Java: About, Dostupné na internete `https://jersey.java.net/`, 23.8.2016

[124] The Apache Software Foundation : Apache Tomcat, Dostupné na internete `http://tomcat.apache.org/`, 23.8.2016

[125] Oracle Corporation: Java Persistence API, Dostupné na internete http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html, 23.8.2016

[126] Eclipse.: EclipseLink, Dostupné na internete http://www.eclipse.org/eclipselink/, 23.8.2016

[127] PARASCHIV, E.: Jackson – Custom Serializer, Dostupné na internete http://www.baeldung.com/jackson-custom-serialization/, 23.8.2016

[128] Apache Maven Project: Welcome to Apache Maven, Dostupné na internete https://maven.apache.org/, 29.10.2016

[129] Python: About Python, Dostupné na internete https://www.python.org/about/, 23.8.2016

[130] RONACHER, A.: Flask, Dostupné na internete http://flask.pocoo.org/, 23.8.2016