Zraniteľnosti a forenzná analýza smartfónov na platforme Android

Analýza a návrh riešenia

Bc. Ján Paraska

1Im, 2016 - 2017

Abstrakt. Mobilné telefóny zaznamenali v tomto a minulom desaťročí enormný rozmach. V súčasnosti sa už aj cenovo dostupnejšie smartfóny dokážu výkonom priblížiť osobným počítačom. Ich konektivita (Wi-Fi, 3G, 4G, a.i.) zabezpečuje používateľom dostupnosť najrozličnejších služieb pomocou veľkého množstva aplikácií, ktoré majú okrem prístupu na internet, prístup k dátam na zariadení. Čoraz rýchlejší nárast počtu aplikácií, však bráni obchodom ako je Google Play v tom, aby dostatočne preverili každú aplikáciu či je škodlivá, alebo nie. Rozhodnutie je tak ponechané na užívateľovi. Ešte horšou správou je, že viac ako 70% aplikácií žiada prístup k dátam, ktoré nesúvisia s ich činnosťou.
Tieto skutočnosti prispeli k tomu, že útočníci sa snažia využiť všetky zraniteľnosti a nedostatky nie len samotných zariadení, či komunikačných kanálov, ktoré využívajú, ale aj spôsobov, akým sú používané bežnými ľuďmi. Výsledkom je množstvo hrozieb pre súkromné dáta užívateľov od SMS správ, e-mailovej komunikácie a bezpečnostných hesiel, čo môže viesť nie len k duševnej ujme ale aj finančným škodám.
Cieľom tejto práce je analýza známych útokov na platforme Android. Objektom skúmania bude niekoľko aktuálnych malvérových hrozieb, z ktorých je možné pomocou nástrojov reverzného inžinierstva, ako napr. ApkTool alebo Dex2Jar extrahovať ich pôvodný zdrojový kód. Ďalším krokom je zaradenie využitých zraniteľností a nedostatkov týmito útokmi do aktuálnej klasifikácie. Ako voliteľný cieľ si táto práca stanovuje využitie postupov forenznej analýzy na zistenie stavu mobilného zariadenia po úspešnom či neúspešnom útoku vybraného malvéru.

Kľúčové slová: Android, smartfón, bezpečnosť, zraniteľnosť, malware, statická analýza, dynamická analýza, monitorovanie za behu, cuckoo, klasifikácia

1 Úvod

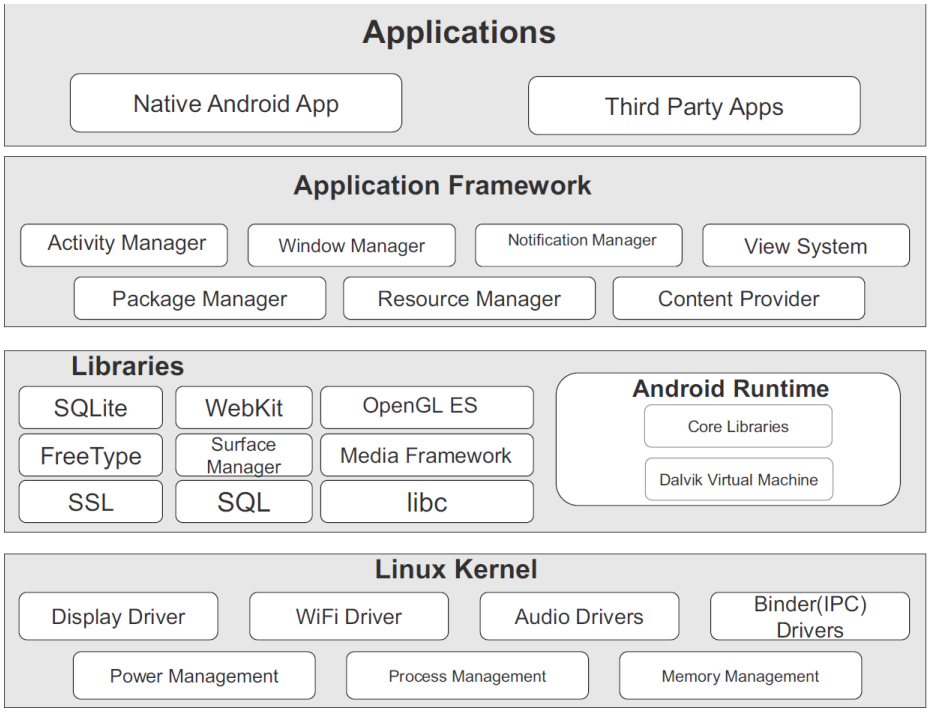
Od svojho uvedenia na trh v roku 2008 sa tešia „inteligentné telefóny“ na platforme Android veľkej obľube. V súčasnosti (02.11.2016) je podiel zaradení s týmto operačným systémom približne 88%. Dostupnosť, vysoký výkon a konektivita spôsobili, že uchovávajú oveľa väčšie množstvo citlivých informácií svojich užívateľov než bežné stolové počítače alebo notebooky. Veľkú zásluhu na tom majú mobilné aplikácie, ktoré poskytujú svojim užívateľom pestrú škálu služieb od zábavy až po manažment a bankovníctvo.

Vďaka otvorenosti OS Android počet aplikácií rastie každým dňom. Na začiatku roku 2017 bol ich počet cca 2.8 milióna. Toto číslo bráni spoločnosti Google, aby kontrolovala každú aplikáciu dostatočne dôkladne bez toho, aby sa tým citeľne nepredĺžil čas jej vydania. Zákazníci sú tak pri hodnotení či je daná aplikácia škodlivá, alebo nie ponechaní na seba. Výsledkom však je, že odhadom 80% aplikácií požaduje povolenia na prístup k dátam, ktoré vôbec, alebo aspoň čiastočne nesúvisia s ich činnosťou. Naviac len 3% užívateľov berie na tieto žiadosti ohľad. K bezpečnosti užívateľov a ich dát neprispieva ani veľký počet neoficiálnych obchodov s aplikáciami, ktorých kontrola je diskutabilná.

1.1 OS Android

Operačný systém Android ako otvorený štandard pre mobilné zariadenia bol vytvorený konzorciom Open Handset Alliance, ktoré zahŕňalo spoločnosti zaoberajúce sa výrobou mobilných telefónov, čipov alebo mobilných aplikácií, napr. Google, Intel, NVidia, Qualcomm, Samsung, a 29 ďalších. Je postavený na jadre systému Linux 2.6 a pozostáva zo štyroch hlavných vrstiev a piatich sekcií.

Obrázok 1. Architektúra operačného systému Android



1.1.1 Jadro

Základom je špeciálne upravené Linuxové jadro pre potreby systému s obmedzenými zdrojmi. Vystupuje ako abstraktná vrstva medzi hardvérom a ostatnými softvérovými vrstvami. Okrem ovládačov zariadení poskytuje aj základnú systémovú funkcionalitu ako správa procesov, správa pamäte a správa zariadení.

1.1.2 Knižnice

Vrstva obsahuje knižnice napísané v jazyku C alebo C++ potrebné pre spracovanie rôznych typov dát. Medzi najdôležitejšie patria WebKit pre zobrazovanie HTML obsahu, SQLite databáza, OpenGL pre zobrazenie 2D a 3D grafiky či MediaFramework obsahujúci rôzne kodeky.

1.1.3 Android Runtime

V rovnakej vrstve ako knižnice sa nachádza základný komponent pre beh aplikácií, tzv. Dalvik Virtual Machine (DVM). Jedná sa o špeciálne upravenú a optimalizovanú Java VM pre potreby Androidu. DVM umožňuje aplikáciám bežať ako procesy priamo na Linuxovom jadre vo vnútri svojho vlastného virtuálneho stroja (uzavreté do sandbox-u). Ako nadstavba Java VM umožňuje vývoj aplikácií v jazyku Java.

1.1.4 Aplikačný framework

Aplikačný framework je množina služieb, ktoré sú poskytované bežiacim aplikáciám vo forme Java tried. Medzi najdôležitejšie patria správca aktivít (ktorý kontroluje všetky aspekty životného cyklu aplikácií a zásobník aktivít), správca zdrojov (slúžiaci na prístup k dátam ako sú znakové reťazce či obrázky) alebo správca upozornení (ktorý umožňuje aplikáciám zobrazovať upozornenia o udalostiach, ktoré sa udiali na pozadí).

1.1.5 Aplikačná vrstva

Nachádza sa na vrchole tejto „hierarchie“ a obsahuje aplikácie predinštalované distribútorom aj aplikácie tretích strán.

1.2 Aplikácie

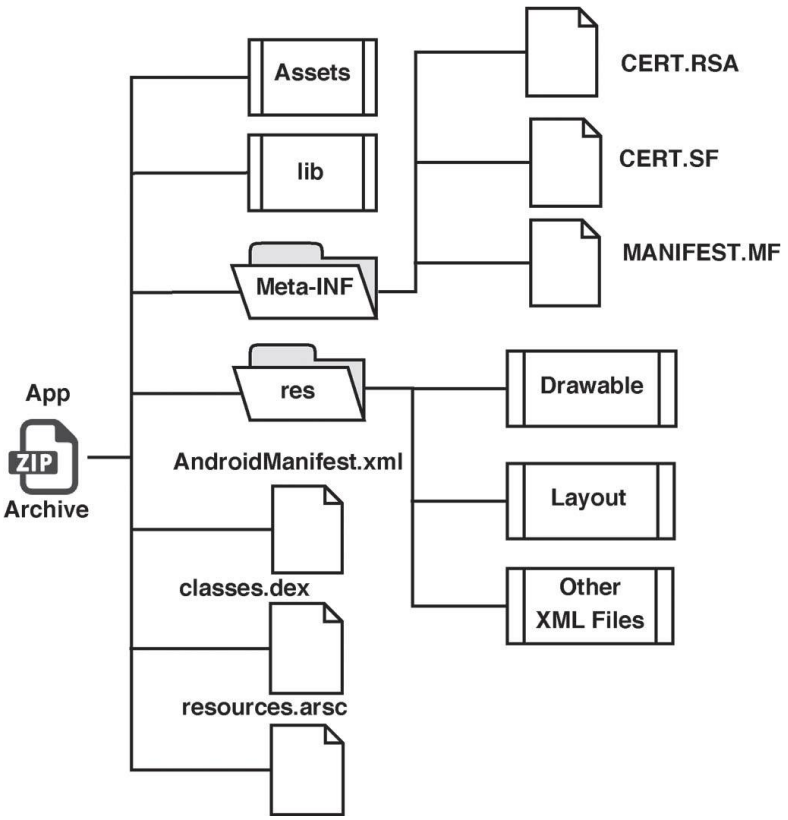
Aplikácie pre systém Android sú primárne písané v jazyku Java, kompilované sú do spustiteľného DVM byte-kódu a uložené v súbore s príponou *.dex*. Tento byte-kód sa od klasického Java byte-kódu odlišuje len neparne.

Dôležitou súčasťou aplikácie je tzv. manifest, ktorý je uložený v súbore *AndroidManifest.xml*. Tento súbor obsahuje všetky meta-dáta ako sú požadované oprávnenia, služby, aktivity či „broadcast receivers“.
Aplikácie tiež pozostáva z jedného alebo viacerých komponentov:

- Aktivity – používateľské rozhranie užívateľa s aplikáciou. Sú spúšťané pomocou tzv. intentov a ich životný cyklus je kontrolovaný správcom aktivít.
- Služby – vykonávajú úlohy na pozadí bez UI (prehrávanie zvuku, sťahovanie dát). Sú spúšťané pomocou intentov a ich životný cyklus môže (ale nemusí) byť zviazaný s komponentom, ktorý ich spustil.
- Broadcast receivers – počúvajú na systémom vytvorené udalosti ako *BOOT_COMPLETED* alebo *SMS_RECEIVED*.

Tento manifest, dex-súbor, knižnice, zdroje a certifikáty sú zabalené do zip súboru s príponou *.apk*, ktorý je dostupný na stiahnutie v obchodoch s aplikáciami.

Obrázok 2. Štruktúra súboru .apk



1.3 Bezpečnostný model

Aplikácie opísané v predchádzajúcej časti využívajú hardvér a softvér zariadenia rovnako ako využívajú prístup k užívateľským dátam na zariadení a prístup na internet. Aby boli tieto dáta, dáta iných aplikácií, zariadenie a komunikačné kanály ochránené, poskytuje Android niekoľko bezpečnostných mechanizmov.

1.3.1 Systém oprávnení

Aby bolo možné zabrániť aplikáciám v neoprávnenom využívaní zdrojov a/alebo funkcií ako sú GPS lokácia, SMS, telefónne hovory či prístup k dátam na SD karte, poskytuje Android bezpečnostný mechanizmus založený na oprávneniach. Vývojár musí v súbore *AndroidManifest.xml* deklarovať oprávnenia, ktoré je aplikácia potrebuje pre svoju činnosť. K dispozícii je viac ako sto oprávnení, rozdelených do štyroch kategórií:

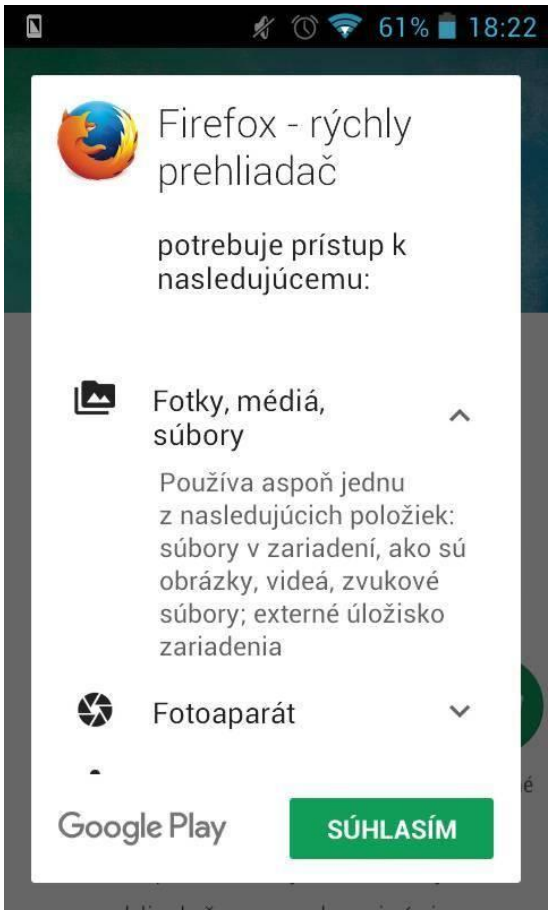
- Normálne (*Normal*) – prístup k dátam s minimálnym rizikom pre užívateľa a systém. Sú prideľované aplikácii automaticky pri inštalácii (čítanie nastavenej časovej zóny).
- Nebezpečné (*Dangerous*) – prístup k citlivým dátam užívateľa alebo kritickým zdrojom zariadenia. Užívateľ ich musí manuálne povoliť pred inštaláciou aplikácie (čítanie kontaktov).
- Podpis (*Signature*) – prístup k dátam je povolený len v prípade, že aplikácia má rovnaký podpis vývojára, ako aplikácia, ktorá deklarovala povolenia na prístup k týmto dátam.
- Podpis alebo Systém (*SignatureOrSystem*) – prístup k dátam je povolený len v prípade že aplikácia má rovnaký podpis ako obraz systému alebo aplikácia, ktorá dané povolenia deklarovala.

Systém oprávnení je veľmi neopracovaný. *READ_PHONE_STATE* umožňuje zistiť či telefón vyzváňa a zároveň dovoľuje aplikácií čítať jednoznačný identifikátor zariadenia. Prístup ku *WRITE_SMS* alebo *WRITE_CONTACTS* nezabezpečujú prístup ku *READ_SMS* a *READ_CONTACTS*. Povolenia nie sú teda ani hierarchicky usporiadané a musia byť deklarované samostatne.

Zhodnotenie toho, či aplikácia nežiada podozrivé oprávnenia sú ponechané na zákazníka, ktorý danú aplikáciu inštaluje. Vzhľadom na to, že väčšina užívateľov na ne neberie ohľad, odhadom 80% aplikácií žiada oprávnenia presahujúce rámec jej činnosti. Kameňom úrazu je tiež fakt, že popis povolení, ktoré zobrazujú oficiálne obchody je často stručný, neúplný či nezrozumiteľný a bežný užívateľ nevie zhodnotiť ich závažnosť.

Naviac k dispozícii sú len dve možnosti. Buď sú povolené všetky žiadané povolenia alebo sa aplikácia nenainštaluje.

Obrázok 3. Náhľad zoznamu oprávnení pred inštaláciou aplikácie



1.3.2 Sandboxing

Jadro Androidu implementuje *Linux Discretionary Access Control* (DAC). Každej aplikácii je pri inštalácii pridelené unikátne ID (UID). Všetky procesy, ktoré táto aplikácia spustí majú pridelené rovnaké UID a dostanú izolovaný adresný priestor, do ktorého sa dostane len proces s rovnakým UID. Tento systém vytvára pre každú aplikáciu izolované prostredie – sandbox.
Okrem UID môže mať aplikácia pridelené jedno alebo viac *Group ID* (GID), ktoré jej garantuje prístup k rôznym zdrojom a/alebo službám. Ak chce aplikácia napr. pristupovať k internetu, využívať Wi-Fi alebo Bluetooth, musí mať pridelené (ID skupiny) *newtork access ID*.

1.3.3 Medzi-komponentová komunikácia

Aj keď aplikácie bežia vo vnútri svojej vlastnej „škatuľky“, často potrebujú medzi sebou komunikovať. Android im preto ponúka mechanizmus Medzi-komponentovej komunikácie (*Inter-Component Communication*). Tú sprostredkúva middleware Androidu. Aplikácia tak môže volať komponenty inej aplikácie ako služby.

1.3.4 Bouncer

Oficiálne obchody s aplikáciami pre Android, akým je Google Play, nekontrolujú každú aplikáciu manuálne (na rozdiel od AppStore). Namiesto toho sa spoliehajú na tvz. *Bouncer* – dynamicky emulované prostredie, ktoré chráni obchod pred hrozbami zo strany nahraných aplikácií. Tento mechanizmus ale nekontroluje ich slabiny, ktoré sú často využívané útočníkmi, resp. škodlivými aplikáciami.

1.4 Bezpečnostné hrozby a nedostatky

Základným stavebným kameňom bezpečnostného modelu OS Android je systém oprávnení. Tento systém je ale často kritizovaný kvôli svojej hrubozrnnej kontrole a neefektívnemu manažmentu zo strany vývojárov, obchodov s aplikáciami.

1.4.1 Eskalácia / získanie oprávnení

Tento typ útoku má za následok pridelenie oprávnení, ktoré neboli resp. nemuseli byť aplikáciou navonok žiadané.

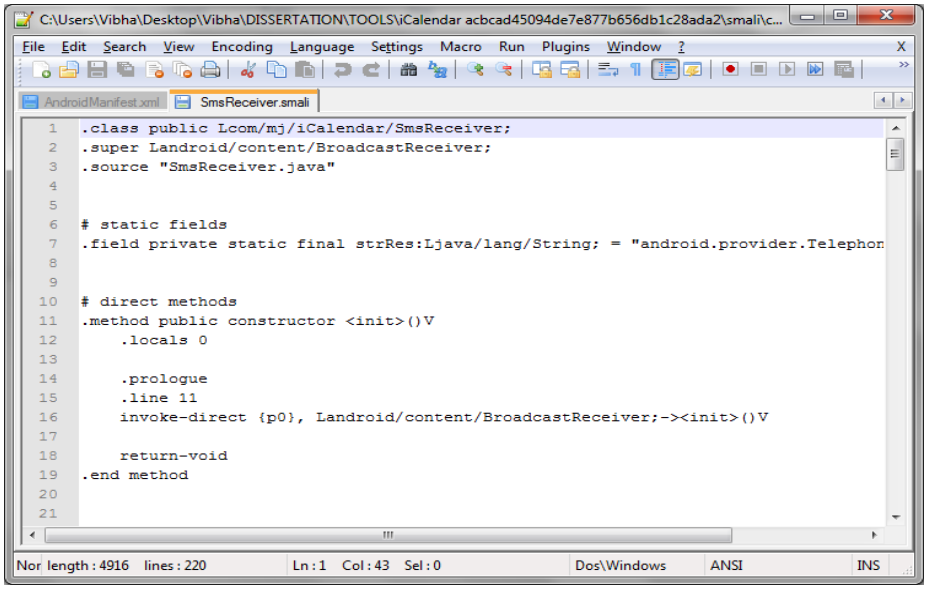
- Manipulácia používateľom – tento spôsob získavania oprávnení často neobsahuje žiadne technické aspekty, no je často až prekvapivo účinný. Užívatelia do značnej miery neberú ohľad na žiadané oprávnenia, prípadne nie sú s týmto bezpečnostným modelom dostatočne oboznámení. Do tejto kategórie patria aj „prebalené“ aplikácie stiahnuté z neoficiálnych obchodov.
- Využitie technickej zraniteľnosti – v ostatných prípadoch malware využije technický nedostatok, zraniteľnosť či zlú konfiguráciu systému/zariadenia. Medzi takéto nedostatky patria zraniteľnosti API, pretečenie buffera, injektovanie kódu, zraniteľnosti ICC či nedostatky v implementácii OS Android, ktoré sú vďaka jeho otvorenosti často verejne známe.

Výsledkom je tzv. rootovanie – pridelenie *root* oprávnení - zariadenia, ktoré je využívané aj skúsenými užívateľmi. Užívatelia si tak môžu prispôsobiť svoj smartfón a útočníci obísť niektoré bezpečnostné mechanizmy.

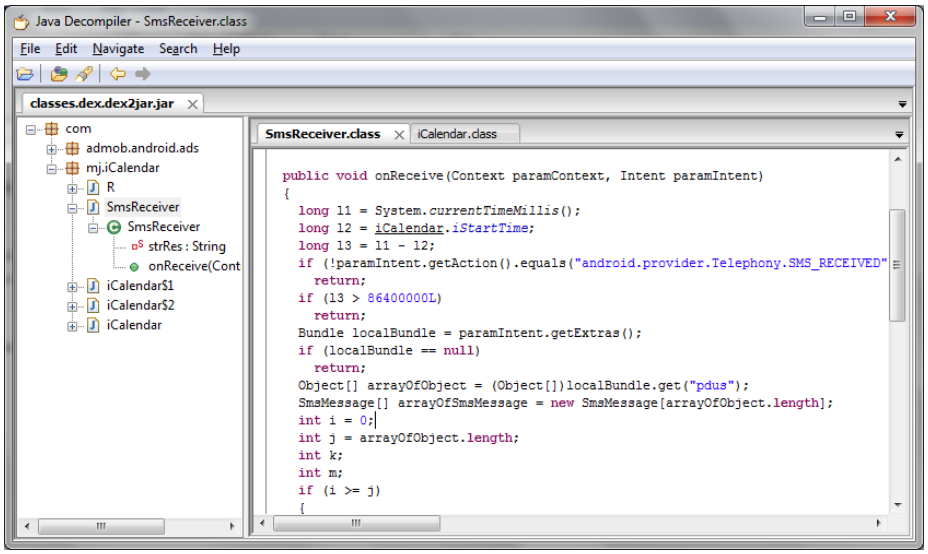
1.4.2 „Prebaľovanie“ aplikácií

Tzv. „prebaľovanie“ aplikácií je jeden z najbežnejších bezpečnostných problémov. Je to proces pozostávajúci z dekompilácie súboru *.apk* pomocou nástrojov reverzného inžinierstva (Dex2Jar, ApkTool, a.i. – tieto nástroje je možné využiť v rámci statickej analázy – viď 2.1. – pri manuálnej inšpekcii kódu), vloženia škodlivého kódu a opätovného skompilovania aplikácie. Prostriedkom sú prevažne populárne aplikácie (vrátane aplikácií bánk), ktoré sú následne ponúkané v alternatívnych obchodoch. Tento prístup nie je výsada poslednej generácie smartfónov, avšak rozmach tejto platformy a veľký počet dostupných aplikácií urobili z „prebaľovania“ aplikácií často využívanú stratégiu. Tento prístup často používa rôzne spôsoby obfuskácie, aby sa vyhol detekcii a sťažil prácu nástrojom statickej analýzy.

Obrázok 4. ApkTool dokáže z .apk súboru extrahovať celú štruktúru aplikácie vrátane zdrojového kódu, ktorý sa dá aj so slabšími znalosťami jazyku Java ľahko prečítať.



Obrázok 5. Nástroj Dex2Jar, ktorý umožňuje rozbaliť súbory .dex (Dalvik Executable) do čitateľnej podoby.



1.4.3 Stiahnutie škodlivého kódu (Update attack)

„Prebalená“ aplikácia nemusí nevyhnutne obsahovať celý škodlivý kód. V niektorých prípadoch tvorcovia vložia kód, ktorý na pozadí stiahne a nainštaluje inú škodlivú aplikáciu. Alternatívou je nahradenie už nainštalovanej aplikácie jej prebalenou verziou. Komunikácia takto vložených komponentov je často šifrovaná, aby sa predišlo detekcii.

1.4.4 Colluding – spolčovanie aplikácií

Pri tomto spôsobe si užívateľ nainštaluje niekoľko rôznych aplikácií od toho istého vývojára (s rovnakým podpisom), ktoré majú pridelené rôzne oprávnenia. Po inštalácii majú aplikácie možnosť zdieľať UID, čím môžu využívať prístup k oprávneniam prideleným aspoň jednej z nich. Spoločne tak môžu aplikácie vykonávať škodlivú činnosť, pričom ich vlastná funkcionalita zostáva navonok v poriadku. Napríklad aplikácia s povolením čítať SMS môže požiadať svojho spolčeného partnera s povolením prístupu na internet, aby správy odoslal na vzdialený server.

1.4.5 Mobilné botnety

Botnet je skupina kompromitovaných zariadení, ktoré sú na diaľku ovládané útočníkom (botnet master) pomocou príkazov. Príkazy môžu byť jednoduché (posielanie súkromných informácií na server) až po vytvorenie DoS útoku. Bot môže taktiež sťahovať ďalší škodlivý software do zariadenia. Medzi najznámejšie Androidové botnety patria Geimini (prvý malware s vlastnosťami botnetu využívajúci zariadenia Androidu), Answerbot alebo Beanbot.

2 Návrh riešenia

S narastajúcim počtom aplikácií je potrebné hľadať rýchle a spoľahlivé riešenia, pomocou ktorých môžu obchody s aplikáciami alebo aj poskytovatelia bezpečnostného softvéru identifikovať malvér. Jeho analýza je preto nevyhnutná k získaniu znalostí o zraniteľnostiach OS Android a teda aj dosiahnutiu tohto cieľu.

2.1 Statická analýza

Jedným z dvoch najčastejšie využívaných prístupov je statická analýza. Spočíva v dekompilovaní aplikácie a prehľadávaní jej zdrojového kódu. Výhodou tohto spôsobu analýzy je, že skúmaná aplikácia nie je nikdy spustená a celý postup je relatívne rýchly a jednoduchý. Problémom ale je, že pre správnu detekciu škodlivého kódu je potrebné poznať hľadané vzory a nový malvér je bez zapojenia strojového učenia alebo ľudského faktora takmer nemožné detegovať. Ďalším nedostatkom je minimálna schopnosť detekcie šifrovaného alebo polymorfického malvéru.
Hlavné nástroje statickej analýzy sú:

- hľadanie vzorov (signature-based) – z každej aplikácie je možné extrahovať zaujímavé vzory a znaky, ktoré je možné porovnať oproti databáze známych malvérov
- analýza povolení (permission-based) – keďže oprávnenia sú rozdelené do niekoľkých kategórií, je možné vyhodnotiť možné riziko, ktoré predstavuje aplikácia požadujúca oprávnenia ku kritickým zdrojom
- interakcia medzi komponentmi (component-based) – k detekcii škodlivých resp. potenciálne rizikových aplikácií je možné analyzovať interakciu medzi komponentmi aplikácie (aktivity, služby, broadcast receivers)
- analýza dalvik byte-kódu – detailnou analýzou je možné nájsť nebezpečné správanie aplikácie a obísť jednoduché techniky obfuskácie kódu. Keďže byte-kód obsahuje volania funkcií je možné z neho skonštruovať tzv. *control flow graph* a vytvoriť tak postupnosti volaní funkcií a práce s dátami

2.2 Dynamická analýza

V tomto prípade nie je analyzovaný zdrojový kód, ale podozrivá aplikácia je spustená v kontrolovanom prostredí – tzv. *sandboxe*. Beh aplikácie je detailne monitorovaný a zaznamenávaný (logovaný), čo zahŕňa aj zmeny v zariadení, tok dát a sieťovú aktivitu.
Škodlivá činnosť môže byť ale spustená interakciou s užívateľom (používanie UI – dotyky, posúvania, gestá). Android SDK je preto vybavený *monkey* nástrojom, ktorý je schopný simulovať spomenuté udalosti. Nezaznamenané tak môžu ostať činnosti, ktoré sú spustené špecifickými udalosťami, napr. konkrétny čas a dátum. Detekcii sa tiež môžu vyhnúť malvéry schopné detegovať emulované prostredie či spúšťajú škodlivú činnosť s oneskorením.

Medzi základné spôsoby dynamickej analýzy patria:

- kontrola toku dát a volaní API – dáta z citlivých zdrojov (GPS, mikrofón, fotoaparát, a.i.) je možné označiť a následne sledovať ako sa v aplikácií spracovávajú a či sa ich aplikácia nesnaží odoslať z telefónu rovnako je možné sledovať volania API a kontrolovať tak správanie sa aplikácie
- emulácia – spustením aplikácie v izolovanom prostredí je možné sledovať akúkoľvek jej aktivitu bez rizika

2.3 Existujúce riešenia

V súčasnosti existuje niekoľko nástrojov a mechanizmov na detekciu malvéru v systéme Android. Podľa [1] ich možno rozdeliť do troch základných kategórií:

- prevencia – keďže *dalvik byte-kód* je zraniteľný voči reverznému inžinierstvu, útočníci sa často zameriavajú na modifikovanie, zmenu správania či injekciu škodlivého kódu do populárnych aplikácií
    - AppInk – vytváranie verifikačného odtlačku, ako prevencia pred prebaľovaním
    - Kirin – analýza žiadaných oprávnení pri inštalácii a porovnanie ich zoznamu oproti databáze známych malvérov
- analýza – rovnako ako pri PC, aj mobilný malvér sa snaží vyhnúť sa detekcii. Nástroje v tejto kategórii sa tak snažia odhaliť škodlivé správanie, podobné aplikácie (podozrivé z prebalenia), odhalenie zneužitia pridelených oprávnení alebo detegovanie zraniteľností aplikácií
    - RiskMon – na základe strojového učenia sa snaží identifikovať rizikové správanie sa aplikácie
    - AppGuard – aplikuje používateľom špecifikované bezpečnostné pravidlá na podozrivé aplikácie
- monitorovanie za behu – spočíva predovšetkým s kontrolovaním privilégií pri pokuse o získanie zdroja (kvôli eskalácii oprávnení) a celkového monitorovania aktivít aplikácie
    - RecDroid – radí užívateľom pri povoľovaní, resp. zamietaní žiadaných povolení na základe dát získaných tzv. crowdsourcing-om
    - MockDroid – vytvára „mock“ dáta pre spustenú aplikáciu čím dáva užívateľovi náhľad do skutočnej funkcionality podozrivej aplikácie
    - TaintDroid – umožňuje sledovať tok dát pri používaní aplikácie

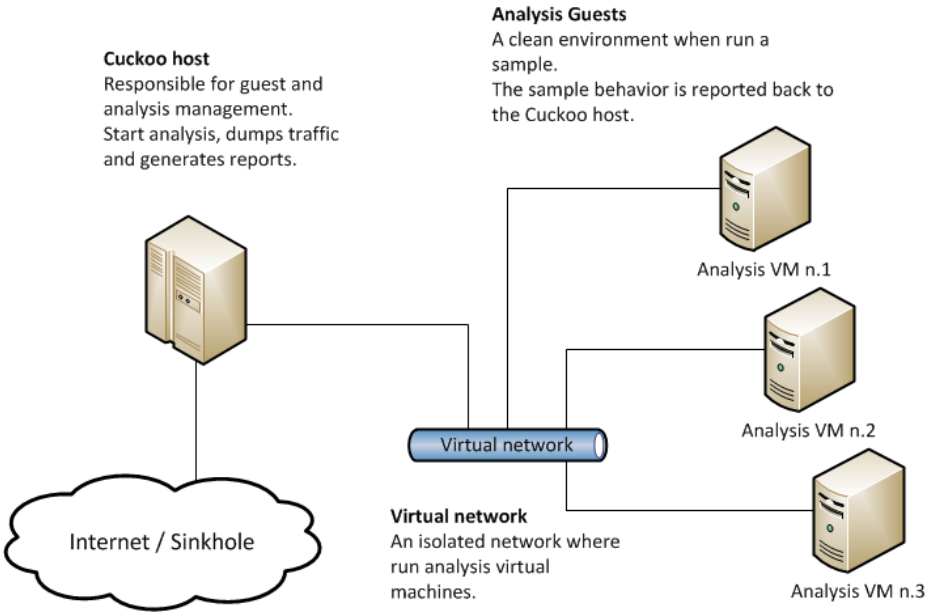
2.3.1 Cuckoo Sandbox

Cuckoo Sandbox je open-source automatizovaný systém na analýzu malvéru, ktorý je schopný vykonávať rôzne typy analýz na každom programe vo virtuálnom stroji. Vznikol v roku 2010 ako súčasť projektu *The Honeynet Project*. Cuckoo podporuje programy pre Windows, Linux, OS X a od verzie 2.0 aj aplikácie pre Android, čo zahŕňa:

- analýzu súborov (spustiteľných, stiahnutých, vytvorených, zmazaných) rôznych typov (DLL súborov, PDF dokumentov, HTML súborov, PHP, VisualBasic a Python skriptov, ZIP súborov a takmer akýchkoľvek iných súborov)
- stopovanie volaní API a správania sa procesov
- analýzu (aj zašifrovanej) sieťovej komunikácie
- analýzu pamäte virtuálneho zariadenia
- analýzu pamäte procesov

Cuckoo pozostáva z centrálneho manažovacieho softvéru, ktorý je zodpovedný za spúšťanie a analýzu vzorky (programu, aplikácie).
Každá takáto analýza je spúšťaná na čistom a izolovanom virtuálnom, prípade fyzickom zariadení. Hlavným komponentom infraštruktúry je hlavný – *Host* - počítač (manažovací softvér) a niekoľko podriadených - *Guest* - počítačov (virtuálnych alebo fyzických, ktoré vykonávajú analýzu).
Hlavný počítač riadi celý proces, zatiaľ čo ostatné počítače sú izolované prostredia, v ktorých sa bezpečne spúšťa a analyzuje vzorka.

Obrázok 6. základná architektúra systému Cuckoo

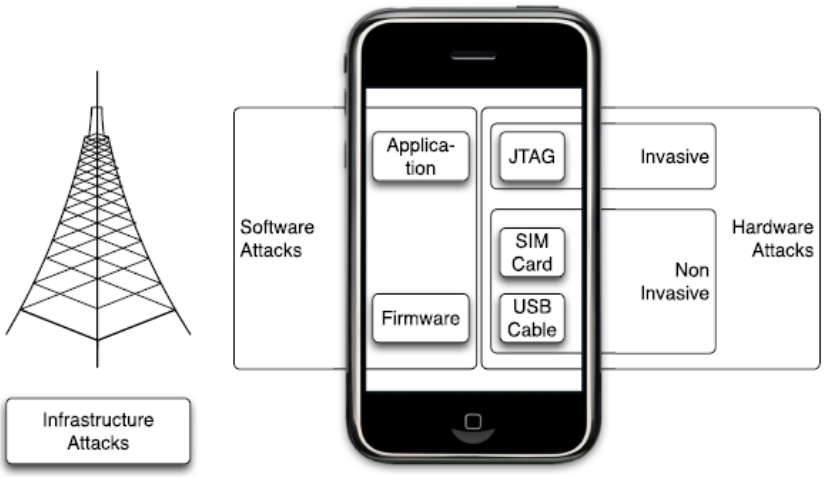


2.4 Klasifikácia zraniteľností

Výstupom analýzy vybraných vzoriek malvéru pomocou systému Cuckoo bude klasifikácia vektorov útokov týchto malvérov. Tie môžeme podľa [6] zaradiť do nasledujúcich kategórií:

- Hardvérové útoky – všetky útoky namierené proti samotnému zariadeniu a vyžadujú si jeho fyzickú prítomnosť. Táto trieda útokov je schopná využiť širokú škálu zraniteľností.
    - Invazívne útoky – pozmeňujú správanie zariadenia. Tento typ útoku si v minulosti vyžadoval veľa času. V súčasnosti však existujú aj rýchle invazívne útoky [12]
    - Neinvazívne útoky – oproti invazívnym útokom sú zneužité odpojiteľné súčasti zariadenia ako SIM karty alebo USB káble. Tento typ útoku umožňuje rýchly prístup k informáciám uloženým na zariadení
- Útoky na infraštruktúru – zameriavajú sa na komponenty využívané zariadením (sieť, online služby, platobné portály a.i.). Umožňujú odpočúvanie bezdrôtovej komunikácie či zaútočiť na využívané služby.
- Softvérový útok – sú zamerané na bežiaci softvér (operačný systém, aplikácie, drivery). Často je využívané aj sociálne inžinierstvo ako spôsob, ktorým si obeť nainštaluje škodlivý softvér.
    - Malvér – cieľom nainštalovanej aplikácieje ukradnúť uložené dáta alebo poškodiť zariadenie. Obeť ho inštaluje v domnení, že sa jedná o legitímnu aplikáciu alebo je využitá zraniteľnosť vo webovom prehliadači
    - Spyware – cieľom je zbieranie informácií o obeti. Takáto aplikácia je nainštalovaná útočníkom bez vedomia užívateľa
    - Grayware – cieľom je popri poskytovaní deklarovanej funkcionality zbierať údaje o používateľovi

Obrázok 7. Klasifikácia útokov



3 Záver

Po naštudovaní potrebnej literatúry sme dospeli k záveru, že na analýzu vzoriek malvéru budú použité existujúce mechanizmy. Kombináciou ich schopností by vznikla charakteristika zraniteľností OS Androidu a kategorizácia analyzovaných útokov. Po diskusii s odborníkmi v oblasti bezpečnosti sme ale tento postup mierne prehodnotili a rozhodli sme sa využiť systém Cuckoo. Ako dúfame, jeho komplexnosť a všestrannosť nám dovolí detailnejšiu analýzu širšieho spektra vzoriek. Výsledná klasifikácia môže potom v budúcnosti slúžiť ako podklad pre vytvorenie aplikácie pre detekciu škodlivých aplikácií v OS Android.

Literatúra

[1] Rashidi, B., Fung, C., A Survey of Android Security Threats and Defenses, Virginia Commonwealth University, Richmond, Virginia, USA
[2] Kireet, .M, Rao, M., S., A Survey on Malware Attacks on Smartphones, International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 3002-3004
[3] Raveendranath, R., Venkiteswaran R, Babu, A., J., Android Malware Attacks and Countermeasures: Current and Future Directions, College of Engineering, Trivandrum, India
[4] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M., S., Conti, M., Rajarajan, M., Android Security: A Survey of Issues, Malware Penetration, and Defenses
[5] Casey, E., Digital Evidence and Computer Crime, Forensic Science, Computers and the Internet, 2011, Elsevier Inc., ISBN 9780123742674
[6] Spreitzenbarth, M., Dissecting the Droid: Forensic Analysis of Android and its malicious Applications, Erlangen, 2013
[7] Suarez-Tangil, G., Tapiador, J., E., Peris-Lopez, P., Ribagorda, A., Evolution, Detection and Analysis of Malware for Smart Devices

[8] Manjunath, V., Reverse Engineering Of Malware On Android, SANS Institute, 2011
[9] Vidas, T., Zhang, C., Christin, N., Towards a General Collection Methodology
for Android Devices, The Digital Forensic Research Conference, New Orleans, LA, 2011
[10] Manjunath, V., Reverse Engineering Of Malware On Android, 2011
[11] Tam, K., Feizzolah, A., Anuar, N., B., Salleh, R., The Evolution of Android Malware and Android Analysis Techniques, 2017

Odkazy

[12]https://arstechnica.com/security/2016/10/using-rowhammer-bitflips-to-root-android-phones-is-now-a-thing/