KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Nikodemas Martinėnas

PERIMTOS ADMINISTRACINIO VARTOTOJO KONTROLĖS IDENTIFIKAVIMO METODAI ANDROID MOBILIUOSIUOSE ĮRENGINIUOSE

Baigiamasis magistro darbas

Vadovas
Doc. dr. Agnius Liutkevičius

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
(parašas) Prof. dr. Algimantas Venčkauskas
(data)

PERIMTOS ADMINISTRACINIO VARTOTOJO KONTROLĖS IDENTIFIKAVIMAS ANDROID MOBILIUOSIUOSE ĮRENGINIUOSE

Baigiamasis magistro darbas
Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Doc. dr. Agnius Liutkevičius
(data)

Recenzentas
(parašas) Prof. dr. Rimantas Plėštys
(data)

Projektą atliko
(parašas) Nikodemas Martinėnas
(data)

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Nikodemas Martinėnas

(Studento vardas, pavardė)

"Informacijos ir informacinių technologijų sauga" (621E10003)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto

„Perimtos administracinio vartotojo kontrolės identifikavimas Android mobiliuosiuose įrenginiuose“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. gegužės 16 d.

Kaunas

Patvirtinu, kad mano **Nikodemo Martinėno** baigiamasis projektas tema „Perimtos administracinio vartotojo kontrolės identifikavimo metodai Android mobiliuosiuose įrenginiuose“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Martinėnas, N. „Perimtos administracinio vartotojo kontrolės identifikavimo metodai Android mobiliuosiuose įrenginiuose“. Magistro baigiamasis projektas / vadovas doc. dr. Agnius Liutkevičius; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2016. 53 psl.

SANTRAUKA

Per pastarąjį dešimtmetį ypač išpopuliarėjo išmanieji mobilieji įrenginiai, kurių dauguma paremti kompanijos „Google“ operacine sistema „Android“. Tarp saugumo iššūkių, aktualių šiems įrenginiams, viena iš ryškėjančių tendencijų yra perimta administracinio operacinės sistemos vartotojo kontrolės prieiga. Tai smarkiai sumažina bendrą mobilaus įrenginio atsparumą informacijos saugos pažeidimams. Mobiliųjų programų, dirbančių su konfidencialia informacija (pavyzdžiui, banko pavedimais), kūrėjai yra suinteresuoti iš anksto identifikuoti tokią nesaugią aplinką, tačiau šiuo metu nėra universalių, viešai prieinamų, lengvai integruojamų pažeistos saugumo būklės požymių patikros būdų.

Šiame magistriniame projekte suformuluojamas ir argumentuojamas metodas, leidžiantis aptikti nesaugią, perimtą administracinio vartotojo kontrolę operacinėje sistemoje. Metodas apibendrina jau egzistuojančius bei siūlo naujus tokios būsenos požymių aptikimo algoritmus. Metodas pagrįstas daugialype įvairių požymių grupių analize, kuri užtikrina, kad perimta administracinio vartotojo kontrolė būtų aptikta net tuose mobiliuosiuose įrenginiuose, kur įdiegta specializuota, aktyviai požymius slepianti programinė įranga.

Eksperimentinėje fazėje metodo pagrindu sukurtas, Java programavimo kalba paremtas prototipas. Atliktų bandymų metu nustatyta, kad ši programa yra efektyvi priemonė ieškant perimtos administracinio vartotojo kontrolės požymių įvairiose „Android“ sistemos konfigūracijose bei naudojimo scenarijuose. Prototipas gali būti lengvai integruojamas su bet kokia „Android“ platformai skirta mobiliąja programa, kuriai reikalingas perimtos administracinio vartotojo kontrolės aptikimo funkcionalumas.

Martinenas, Nikodemas. *Rooted environment detection methods in Android-based mobile devices.*

Master‘s thesis in “Information and Information Technology Security” / supervisor assoc. prof. Agnius Liutkevičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Information technology security in smart mobile devices

Key words: mobile devices, Android, security, root, rooted environment, detection

Kaunas, 2016. 53 p.

SUMMARY

Over the last decade smart mobile devices have become increasingly popular technology. The biggest share of mobile market is held by Android-based devices that are built on operating system created by Google. Among various security challenges that are applicable to such devices one of the more prominent issues is the so-called „rooted“ operating system environment. Such state significantly reduces overall security posture of the device. Mobile applications that are handling and processing sensitive, confidential data (e.g. financial payments) are especially vulnerable to malicious attacks when operating in a rooted operating system. Creators of such applications are looking for ways to identify the unsafe „rooted“ envirnoment in order to stop their program from starting up. However, at this point in time there are no clearly identified, publicly available tools or methods to easily check for rooted environment symptoms.

This paper formulates and describes a method that allows for complete detection of unsafe system where root access is available on devices. Method aggregates existing and offers new detection algorithms. Method is based on multi-layer analysis of various groups of symptoms. This approach ensures that root access is detected even in those devices where there is a specialized cloaking software deployed which actively attempts to hide rooted state.

A prototype based on the method was created using Android Studio in the experimental phase of this paper. Multiple tests were run against the prototype to account for variety of Android configurations and use cases. Based on results of those tests it was confirmed that prototype is effective tool for detection of rooted environment. Prototype or its derivations can be easily integrated with any Android mobile application that requires root detection functionality.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Android – kompanijos „Google“ išpopuliarinta operacinė sistema, skirta išmaniesiems įrenginiams
iOS – kompanijos „Apple“ mobiliesiems ir nešiojamiems įrenginiams skirta operacinė sistema
DVM – „Android“ programų paleidimo aplinka (angl. *Dalvik Virtual Machine*)
APK – „Android“ operacinės sistemos programinių paketų formatas (angl. *Android PacKage*)
SSL – tinklo saugumo protokolas (angl. *Secure Sockets Layer*)
DNS – interneto adresų vardų išrišimo protokolas (angl. *Domain Name System*)
SQL – duomenų bazių užklausų kalba (angl. *Standard Query Language*)
XSS –žiniatinklio atakos tipas (angl. *Cross-Site Scripting*)
XSRF – žiniatinklio atakos tipas (angl. *Cross-Site Request Forgery*)
OWASP – elektroninėje viešojoje erdvėje veikiantis bendruomeninis projektas, skirtas saugumo pažeidimams identifikuoti ir klasifikuoti (angl. *Open Web Application Security Project*)
AVK – administracinio vartotojo kontrolė, apibūdinanti operacinės sistemos būklę, kai originalūs įrenginio gamintojo saugumo apribojimai technologiškai apeinami ir savininkas gauna pilną administracinę prieigą prie įrenginio;
VPN – virtualus privatus tinklo tunelis (angl. *Virtual Private Network*)
Fastboot – kompiuteryje, kuris per USB jungtį sujungtas su mobiliu įrenginiu, įdiegtas įrankis ir tuo pačiu protokolas, skirtas rašyti duomenis tiesiai į telefono darbinę atmintį (angl. *flash memory*)
Atvaizdis – operacinės sistemos diegimo failas, dažniausiai platinamas kaip vienas didelis archyvas
ADB – kompiuteryje, kuris per USB jungtį sujungtas su mobiliu įrenginiu, įdiegtas įrankis ir tuo pačiu protokolas, skirtas tiesiogiai komunikuoti su įrenginiu per komandinę eilutę (angl. *Android Debug Bridge*)
Root – aukščiausias (administracines) prieigos ir valdymo teises turintis sisteminis vartotojas „Unix“ šeimos operacinėse sistemose, įskaitant ir „Android“
GPS – globali, satelitinė koordinačių nustatymo sistema (angl. *Global Positioning System*)

ĮVADAS

Šis magistro baigiamasis darbas priklauso Kauno technologijos universiteto magistrantūros studijų programai "Informacijos ir informacinių technologijų sauga" (621E10003).

Šiame darbe nagrinėjama išmaniųjų mobiliųjų telefonų saugumo problema, atsirandanti tais atvejais, kai vartotojai savavališkai perima mobiliosios operacinės sistemos kontrolę (angl. *root privilege*). Mobiliųjų telefonų gamintojai kurdami įrenginius jų sistemos saugą modeliuoja darydami prieladą, kad vartotojai telefonu naudosis neprivilegijuota sistemine paskyra (angl. *user account*). Vartotojams apėjus šį saugumo modelį atsiranda papildomos saugumo grėsmės, kurių gamintojai nėra nenumatę ir kurios gali sąlygoti įvairius mobilaus įrenginio saugumo pažeidimus.

Darbo pradžioje apžvelgiami egzistuojantys „Android“ operacinės sistemos nulaužimo metodai bei priemonės. Taip pat nagrinėjami tokių aplinkų identifikavimo metodai, išskiriant ir palyginant jų trūkumus ir privalumus. Darbo eigoje suformuluojamas ir pasiūlomas naujas išbaigtas metodas, skirtas aptikti kuo didesnį spektrą operacinės sistemos su perimta administracinio vartotojo kontrole simptomų. Prototipo realizavimo bei eksperimentinėje dalyje aprašomi konkretūs bandymai, skirti įvertinti pasirinkto metodo efektyvumą.

Darbo pabaigoje pateikiamos išvados su bandymų įvertinimais bei rekomendacijomis, kaip toliau tobulinti metodą ir prototipą.

Darbo problematika ir aktualumas

Darbe nagrinėjamos esminės saugumo problemos, susijusios su mobiliosiomis programomis „Android“ operacinėje sistemoje, kai telefono vartotojas turi sisteminio administracinio vartotojo (angl. *rooted phone*) privilegijas. Mobiliosios programos, veikdamos potencialiai nesaugioje aplinkoje, susiduria su daugybe papildomų rizikų, tokių kaip neteisėtas konfidencialių duomenų perėmimas arba pačios programos atvirkštinė inžinerija. Tai ypač aktualu su bankinėmis sistemomis dirbančioms programoms, kur saugumo pažeidimai gali sąlygoti rimtus nuostolius tiek vartotojams, tiek mobiliųjų programų kūrėjams.

Nors privilegijuotos kontrolės perėmimo problema aktuali visoms mobiliosioms operacinėms sistemoms, įskaitant „iOS“, „Blackberry“ ar „Windows Mobile“, šiame darbe išimtinai koncentruojamasi į „Android“ operacinę sistemą, kadangi tai yra populiariausia ir labiausiai įsilaužėlių atakuojama platforma.

Darbe nagrinėjama problematika bei siūlomi saugumo sprendimai ypač aktualūs tiek mobiliųjų programų kūrėjams, programuotojams, užsakovams, tiek ir saugumo ekspertams, užsiimantiems programų saugumo patikra.

Darbo tikslas ir uždaviniai

Šio darbo *tikslas* yra sukurti metodą bei realiai įgyvendinti juo paremtą prototipą, skirtą aptikti nesaugią, t. y. su technologiškai perimta administracinio vartotojo kontrole (toliau - AVK), operacinės sistemos aplinką. Darbo tikslas bus laikomas pasiektu, jei prototipo eksperimentinio įvertinimo metu bus įrodyta, kad prototipas efektyviau už kitus egzistuojančius metodus bei produktus identifikuoja operacinės sistemos su perimta privilegijuota kontrole simptomus.

Darbo uždaviniai:

1. išanalizuoti egzistuojančius administracinio vartotojo kontrolės perėmimo būdus ir populiariausių tam skirtų įrankių veikimo principus;
2. išanalizuoti egzistuojančius administracinio vartotojo kontrolės aptikimo metodus ir tam skirtus komercinius produktus;
3. nustatyti ir palyginti aptikimo metodų / įrankių privalumus bei trūkumus;
4. sukurti ir pasiūlyti visavertį perimtos administracinio vartotojo kontrolės aptikimo metodą;
5. metodo pagrindu sukurti prototipą;
6. eksperimentiškai įvertinti metodo bei AVK aptikimo prototipo efektyvumą.

Darbo rezultatai ir jų svarba

Mobiliųjų programų, kurios dirba su konfidencialia informacija (pvz., banko pavedimais), kūrėjai yra suinteresuoti iš anksto identifikuoti tokią nesaugią aplinką, nes programišių atakos prieš tokias programas gali reikšti finansinėms įmonėms didelius nuostolius. Šiame magistriniame projekte suformuluojamas ir argumentuojamas metodas, leidžiantis aptikti nesaugią, perimtą administracinio vartotojo kontrolę sistemoje. Metodo pagrindu sukuriamas veikiantis prototipas - mobilioji „Android“ programa. Eksperimentiniu būdu įrodomas tokio prototipo efektyvumas lyginant su egzistuojančiais komerciniais sprendimais. Metodo pagrindu sukurta mobilioji programa ar Java klasių rinkinys gali būti lengvai integruojamas į bet kuria „Android“ mobiliąją programa, kuriai reikalingas perimtos administracinio vartotojo kontrolės aptikimo funkcionalumas.

Darbo struktūra

Šiame darbe yra trys pagrindiniai skyriai.

Pirmajame pateikta „Android“ kaip operacinės sistemos ir kaip mobiliųjų įrenginių platformos analizė, koncentruojantis į sistemos saugumo principus bei architektūrą. Skyriaus tikslas yra ištirti ir nustatyti, dėl kokių priežasčių ir kokiais būdais operacinė sistema gali būti nulaužta pačių vartotojų, kokie egzistuojantys metodai taikomi administracinio vartotojo kontrolės aplinkai aptikti, kokie akademiniai tyrimai buvo atlikti šia tema ir kokie šį funkcionalumą teikiantys komerciniai produktai šiuo metu egzistuoja rinkoje.

Antrajame skyriuje aprašomas naujai siūlomas metodas, skirtas apjungti visus įmanomus simptomų aptikimo testus. Metodas detalizuojamas išsamiai aprašant kiekvieną simptomą bei jiems aptikti skirtų testų kategoriją.

Trečiajame skyriuje aprašomas sprendimo prototipo projektas. Išnagrinėti realizacijos metu naudoti įrankiai bei sukurti komponentai, atsižvelgiant į antrajame skyriuje iškeltus funkcinius ir nefunkcinius reikalavimus. Tuomet pateikiamas detalus tyrimo eigos aprašymas, pateikiami bandymų metu surinkti duomenys ir analizuojami rezultatai. Tokiu būdu įvertinamas pasiūlyto saugos metodo efektyvumas.

Darbo pabaigoje teikiama bendra darbo apžvalga, apibendrinamos išvados, pateikiamas literatūros sąrašas.

1. ADMINISTRACINIO VARTOTOJO KONTROLĖS „ANDROID“ OPERACINĖJE SISTEMOJE PERĖMIMO, IDENTIFIKAVIMO BEI PREVENCIJOS ANALIZĖ

1.1. Analizės tikslas

Norint įsigilinti į specifines „Android“ saugumo problemas, pirmiausia reikia apžvelgti pačią „Android“ operacinės sistemos struktūrą ir architektūrą, ypač atkreipiant dėmesį į įdiegtus saugumo kontrolės mechanizmus. Tai leis geriau suprasti saugumo grėsmes ir įvertinti jų įvykio tikimybę bei potencialią žalą.

Šio skyriaus tikslas yra išanalizuoti „Android“ apsaugos mechanizmus bei kaip tuos mechanizmus gali įveikti sistemai nulaužti skirtos atakos priemonės. Skyriaus eigoje detalizuojama nagrinėjama problema – administracinio vartotojo kontrolės (angl. *root*) perėmimo metodai, būdai bei įrankiai. Identifikuojamos esminės dėl tokios veiklos kylančios rizikos pačiam vartotojui. Nustatomi ir palyginami egzistuojantys metodai bei priemonės tokiai aplinkai aptikti. Skyrius užbaigiamas išvadomis, kokie yra pagrindiniai egzistuojantys aptikimo metodų trūkumai, ir ką galima tobulinti.

1.2. „Android“ platformos ir saugumo iššūkių apžvalga

1.2.1. Išmaniųjų mobiliųjų telefonų rinkos ir saugumo apžvalga

Modernių išmaniųjų telefonų eros pradžia galima laikyti 2007 metus, kai kompanija Apple išleido savo pirmąjį išmanųjį telefoną „iPhone“, o po metų rinkoje pasirodė ir pirmasis „Android“ operacine sistema paremtas išmanusis telefonas „HTC Dream“. Nuo to laiko pasaulyje išmaniųjų telefonų naudojimas kasmet augo didžiuliais tempais, ir šiuo metu išmaniuoju mobiliuoju telefonu naudojasi kas trečias pasaulio žmogus [1]. Išmaniųjų telefonų pagrindinė populiarumo priežastis yra jų neišsemiamas pritaikymo spektras – nuo bankininkystės ir socialinės erdvės, iki kompaso ir kalendoriaus. Šiuose įrenginiuose įtaisyti įvairūs davikliai ir mikroįrenginiai: belaidis ryšys (angl. *WiFi*), balso apdorojimo įrenginiai, duomenų apdorojimo įrenginiai, GPS posistemė, ir kt.

Žvelgiant į išmaniųjų telefonų bei planšetinių kompiuterių rinkos pasidalinimą pagal operacines sistemas, neabejotinai dominuojanti yra atvirojo kodo Google kompanijos platforma „Android“. Tarptautinės rinkos tyrimų kompanijos „Gartner“, besispecializuojančios informacinėse technologijose, 2015 lapkričio mėnesio duomenimis „Android“ operacinę sistemą naudojo beveik 85 proc. išmaniųjų telefonų [2].

1.1 lentelė. Mobiliųjų įrenginių rinkos pasiskirstymas pagal operacines sistemas (2015 m.)

Operacinė sistema	**2015 III ketvirtis**	**Rinkos dalis (proc.)**	**2015 III ketvirtis**	**Rinkos dalis (proc.)**
„Android“	298 797 000	84,7	254 354 000	83,3
„iOS“	46 062 000	13,1	38 187 000	12,5
„Windows“	5 874 000	1,7	9 033 000	3,0
„BlackBerry“	977 000	0,3	2 420 000	0,8
Kitos	1 133 000	0,3	1 310 000	0,4
Iš viso	352 844 000	100,0	305 384 000	100,0

„Android“ operacinę sistemą savo gaminiuose naudoja absoliuti dauguma išmaniųjų telefonų gamintojų. Kadangi viena esminių išmaniųjų telefonų galimybė yra įdiegti bet kokią, su originaliu gamintoju nesusijusią mobiliąją programą, lygiagrečiai susikūrė plati mobiliųjų programų kūrėjų bei vartotojų ekosistema. Dėl nuolat kintančių vartotojų poreikių ir išmaniųjų įrenginių specifikacijų, mobiliosios programos į rinką paprastai yra išleidžiamos greitais ciklais, prioritetą teikiant funkcionalumui, kuris tiesiogiai susijęs su programos nešamomis pajamomis. Tokios programos dažnai nėra pakankamai testuojamos, be to, jų kūrimu užsiima ne tik patyrę profesionalai, bet ir mėgėjai. Akivaizdu, kad dėl nepakankamos programavimo disciplinos bei programų testavimo kenčia

tokių programų sauga. Saugumo problemos mobiliuosiuose įrenginiuose ne tik atkartojo panašias problemas, jau nuo seno egzistuojančias asmeniniuose kompiuteriuose, bet ir dar tapo paaštrintos saugumo rizikų, būdingų specifiškai mobiliosioms platformoms. Todėl išmanieji telefonai tapo labai patraukliais ir lengvai pažeidžiamais taikiniais, kuriuos netruko pastebėti kenkėjiškų programų autoriai.

IT saugos industrijoje universaliai pripažinta „OWASP“ pažeidžiamumų reitingavimo sistema mobiliesiems įrenginiams nustatė dešimties opiausių saugumo problemų, susijusių su mobiliaisiais įrenginiais, sąrašą [3]:

- nepakankami serverio saugos mechanizmai;
- nesaugi įrenginio duomenų saugykla;
- nepakankama transporto sluoksnio apsauga;
- atsitiktinis duomenų nutekinimas;
- autorizacijos ir autentifikacijos trūkumai;
- netinkami kriptografiniai sprendimai;
- kliento pusės injekcijos atakos;
- nepatikimos programos įvestys;
- netinkamas prisijungimo sesijos valdymas;
- nepakankama sisteminio kodo apsauga.

Šis sąrašas atspindi pažeidžiamumus, aktualius bendro pobūdžio mobiliosioms programoms: žaidimams, kalendoriams, pomėgiams, žemėlapiams ir t.t. Tačiau tam tikrose veiklos ir verslo nišose išryškėja kiti, specifiniai saugos trūkumai. Vienas iš jų, administracinio vartotojo teisių perėmimas mobiliajame įrenginyje, bus detalizuojamas toliau.

1.2.2. Administracinio vartotojo kontrolės perėmimo problemos apžvalga

Standartiškai tiek „Android“, tiek „iOS“ operacinėse sistemose vartotojui nėra suteikiamos operacinės sistemos administratoriaus teisės. Taip gamintojai apsisaugo nuo įvairių saugumo problemų, kurios gali kilti vartotojui nekorektiškai naudojantis sistema ar netyčia įdiegus kenkėjiškas programas. Tačiau vis dažniau vartotojai savanoriškai nulaužia šias apsaugas ir perima išmaniojo telefono privilegijas (angl. *root / jailbreak*) tiek „Android“, tiek „iOS“ operacinėse sistemose apeidami gamyklinius apribojimus. Priežastys, skatinančios apeiti šiuos apribojimus, yra įvairios, bet dažniausiai tai yra įrenginių savininkų noras turėti didesnę įrenginio kontrolę, įdiegti nestandartinius ar iš neoficialių svetainių parsisiųstus programinius paketus, turėti galimybę daryti pilnas (t.y. failų sistemos lygmens) rezervines mobilaus įrenginio kopijas.

Kad ir kokie būtų geri vartotojų ketinimai, tokie įrenginiai smarkiai padidina egzistuojančius informacijos saugos rizikos faktorius ir neretai įneša naujus: padidėjusi kenkėjiško užkrato tikimybė, neveikiantys atnaujinimai, neadekvačiai plačios programų privilegijos. Įrenginyje su perimta administracinio vartotojo kontrole smarkiai paveikiama veikimo garantija, tiek dėl minėtų saugumo problemų, tiek dėl techninių priežasčių (pvz. padidėjęs resursų, baterijos sunaudojimas). Nors pats AVK perėmimo veiksmas daugumoje valstybių nėra nelegalus, ypač kai šis veiksmas atliekamas su nuosavu įrenginiu, tačiau esant tam tikriems panaudojimo atvejams, tokia nesaugi aplinka yra nepageidautina žvelgiant iš mobiliųjų programų kūrėjų perspektyvos:

- finansines operacijas (pvz. bankinius pavedimus ar prekybą valiutomis) teikiančios mobilios programos pagrįstai vengia veikti nesaugioje aplinkoje dėl smarkiai padidėjusios sukčiavimo, vagysčių ar identiteto perėmimo rizikos;
- mobilaus įrenginio garantiją teikiančios įmones (gamintojai, platintojai, techninės pagalbos teikėjai) suinteresuoti identifikuoti vartotojus su nulaužta operacine sistema, kadangi tokiu atveju garantija yra anuliuojama;
- nuosavo įrenginio įmonės tinkle (angl. BYOD, *Bring Your Own Device*) strateginį valdymą teikiančios programos veikdamos padidintos rizikos aplinkoje gali viršyti įmones rizikos toleranciją ir prieštarauti saugos politikai;

• autorių teisių apsaugos organizacijos, nusikalstamos veiklos tyrėjai tam tikrose situacijose gali būti suinteresuoti žinoti, ar mobilusis įrenginys nusikaltimo metu buvo nulaužtas, nes tai gali paveikti tyrimo eigą.

Tokio tipo mobiliosioms programoms aktualu identifikuoti, ar operacinė sistema yra su perimta administracinio vartotojo kontrole (AVK). Sėkmingai nustačius tokį faktą, šios programos tada gali daryti atitinkamą, autorių numatytą loginį sprendimą: pavyzdžiui, sustoti ir išsijungti. Tačiau nėra akivaizdaus ar paprasto būdo greitai nustatyti, ar operuojama padidintos rizikos aplinkoje, juolab kad dauguma privilegijuotai kontrolei perimti skirtų paketų aktyviai „slepiasi“ nuo standartinių aptikimo metodų.

1.3. Perimtos administracinio vartotojo kontrolės „Android“ platformoje apžvalga

1.3.1. „Android“ operacinė sistema ir jos saugumo modelis

„Android“ operacinės sistemos ištakos siekia 2003 metus, bet reali jos sėkmė prasidėjo kompanijai „Google“ įsigijus teises 2005 metais. Jau 2008 metais buvo išleistas pirmas įrenginys su šia operacine sistema. Pati Google įrenginių negamina, tiesiog operacine sistema leidžia naudotis didiesiems telefonų ar planšetinių kompiuterių gamintojams. „Android“ operacinė sistemoje naudojamas „Linux“ operacinės sistemos branduolys, bet likusi sistemos dalis smarkiai modifikuota ir pritaikyta mobiliajai aplinkai. Esminė to pasekmė yra tai, kad nemažai „Linux“ sistemos saugumo spragų ar bendro pobūdžio rizikų bus aktualios ir „Android“ sistemai (pavyzdžiui, failų ir direktorijų struktūra, prieigos teisių nustatymo sistema, produktų gamintojų programinės bibliotekos ir kt.). Bendrai paėmus, „Android“ galima laikyti viena iš „Linux“ distribucijų.

„Android“ architektūra yra programinės įrangos dėklas (angl. *stack*) su trimis esminiais komponentais: pati operacinė sistema, tarpinės programinės įrangos sluoksnis (angl. *middleware*) bei pagrindinės programos. Žemiau pateikiama bazinė „Android“ architektūros iliustracija:

1.1 pav. „Android“ operacinės sistemos programinė architektūra [4]

Programos - „Android“ telefonuose dažniausiai būna gamintojo įdiegtos numatytosios programėlės, tokios kaip el. pašto klientas, naršyklė, kalendorius, SMS programa, žemėlapiai. Dažniausiai programuojama su Java kalba.

Programų karkasas (angl. *framework*) - programų autoriams suteikiama galimybė naudotis įvairiomis įdiegtomis funkcijomis, pavyzdžiui, buvimo vietos nustatymas, aliarmo konfigūracija, prieiga prie aparatūrinės įrangos, foninių servisų nustatymai, ir kt.

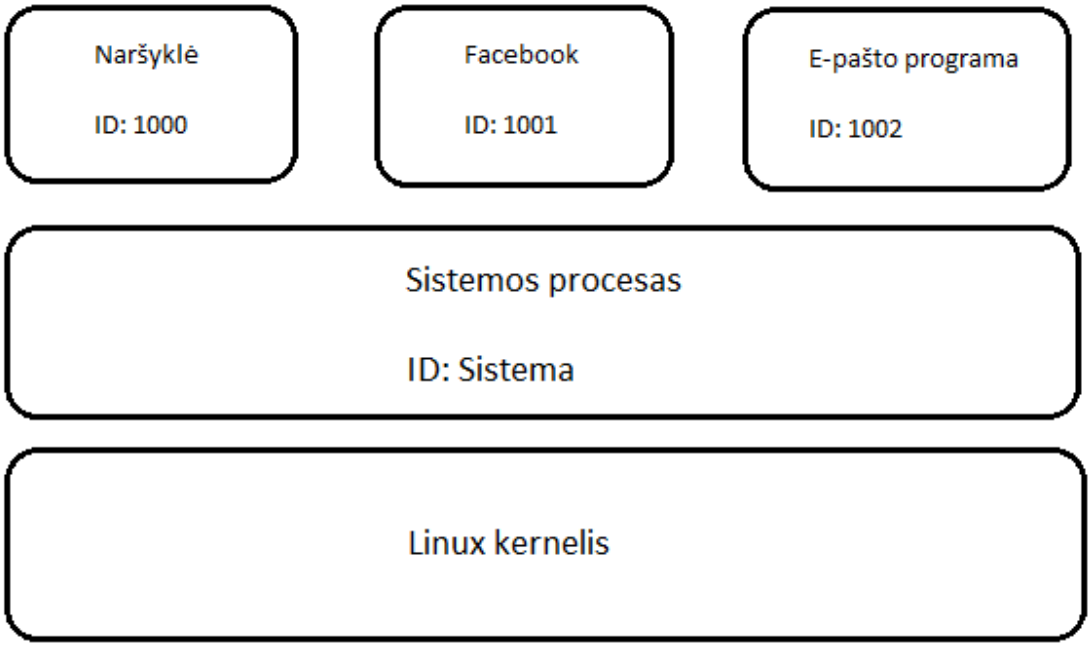
Bibliotekos (angl. *libraries*) - „Android“ sistema platinama su C/C++ bibliotekų komplektu, kurį naudoja keletas sistemos komponentų.

Programos vykdymo aplinka (angl. *runtime*) – skirta Java kalbos programų vykdymui. Visos mobiliosios programos turi joms priskirtus atskirus procesus ir yra virtualiai izoliuotos vadinamojoje „Dalviko“ virtualiojoje mašinoje („DVM“).

Linux branduolys - esminis operacinės sistemos variklis, užtikrinantis procesų, tvarkyklių, atminties, saugos ir tinklo valdymą.

„Android“ sistemos saugumo modelis sukurtas taip, kad nei viena programa veikimo metu neturėtų galimybės piktavališkai ar atsitiktinai paveikti kitą įrenginyje veikiančią programą, vartotoją ar pačią operacinę sistemą. Kiekvienai programai priskiriamas atskiras procesas, todėl „Android“ laikoma daugiaprocesine sistema. Programoms priskiriami Linux sistemos vartotojai ir vartotojų grupės. Taip sistemos procesų lygyje užtikrinamas saugumas tarp sistemos ir programos.

„Smėlio dėžės“ principas (angl. *sandbox*) bei privilegijų mechanizmas yra pagrindiniai „Android“ saugumo modelio komponentai [5]. Kiekvienos programos kodas leidžiamas tam reikalui skirtoje DVM mašinoje su atskiro vartotojo privilegija. Kiekviena programa vykdoma izoliuotai nuo kitų ir neturi teisės prieiti prie kitų programų failų.



1.2 pav. „Android“ saugumo modelis: programų paskyrų atskyrimo pavyzdys

Kiekviena „Android“ programa turi būti pasirašyta su saugumo sertifikatu, kurio atitinkamas privatus raktas žinomas tik programos autoriui. Taip programos autorius gali save, reikalui esant, vienareikšmiškai identifikuoti. Kai programa diegiama įrenginyje, jai priskiriamas unikalus „Linux“

vartotojo identifikacinis numeris taip išvengiant konfliktų su kitomis programomis. Programai privalu diegimo metu nurodyti visus resursus, kurių jai prireiks veikimo metu. Šių teisių užklausa perduodama įrenginio vartotojui, kuris atsižvelgdamas į programos sertifikatus ir kitus autentiškumą ir saugumą užtikrinančius faktorius gali suteikti reikalaujamas privilegijas arba šį prašymą atmesti.

Esminis „Android“ skirtumas tiek nuo gimininges „Linux“, tiek nuo kitų „Unix“ ar „Windows“ šeimos operacinių sistemų yra tas, kad „Android“ vienas šalia kito egzistuoja skirtingi programiniai paketai, tarp kurių nėra jokių pasitikėjimo ryšių. Netgi vartotojas, diegdamas naują programinę įrangą, gali ne iki galo ja pasitikėti ir atitinkamai suteikti jai tik labai apribotas veiklos operacinės sistemos terpėje teises. „Android“ programinės įrangos paketai teoriškai gali perimti sistemos valdymą, kadangi vartotojas retai supranta, koks yra pilnas įdiegto paketo funkcionalumas. Dėl šių saugumo sumetimų „Android“ taikomi tokie apsaugos mechanizmai:

- apribojamas programinio paketo veiklos kontekstas bei sisteminės teisės (angl. *sandboxing*);
- programinių paketų tarpusavio komunikacijai taikoma prieigos teisių kontrolė, t.y. kam leidžiama prieiti prie paketo duomenų ar metodų, ir atitinkamai prie kokių resursų „Android“ sistema leidžia prieiti pačiam paketui;
- Mobiliosios programos paketas prieigos kontrolės taisykles ir reikalavimus apsirašo faile AndroidManifest.xml.

Kitas esminis „Android“ saugumo aspektas – kiekvienas programinis paketas įdiegiamas kaip atskiras, unikalus sisteminis vartotojas. Kiekviena programa turi savo nuosavą namų direktoriją, kurioje talpinamas programos kodas, duomenys bei darbiniai failai. Prie šios direktorijos ir jos turinio gali prieiti tik tos programos sisteminis vartotojas, todėl bet kokia kita programa ar vartotojas neturi jokios galimybės pasiekti ne savo direktorijoje esančius duomenis. Bet koks tokių resursų dalinimasis su kitomis programomis gali būti tik aktyvuotas tik su savininko leidimu.

Svarbu prisiminti, kad išmaniojo telefono savininkas sistema naudojasi kaip neprivilegijuotas vartotojas. Jis neturi tiesioginio priėjimo prie daugelio sisteminių resursų, procesų ar konfigūracinių failų ir telefonu gali naudotis tik taip, kaip numatyta telefono gamintojo. Akivaizdu, kad ne visus vartotojus tenkina tokia apribota galimybė naudotis įrenginiu, todėl vartotojai ieško būdų, kaip apeiti gamintojo saugumo apribojimus.

1.3.2. Administracinio vartotojo kontrolės perėmimo priežastys

Trumpai apžvelgsime pagrindines priežastis, kodėl vartotojai siekia apeiti gamyklinius privilegijuotos kontrolės apribojimus (angl. *root access*) [6]. Šio konteksto suvokimas padės geriau suprasti kitus šio darbo skyrius, kuriuose nagrinėjami perėmimo metodai ir įrankiai.

Kadangi standartiškai į „Android“ įrenginius naujus programinius paketus galima diegti tik per „Google Play“ ar kitas oficialias mobiliųjų programų elektronines prekyvietes, bet kokia piratinė programinė įranga į „Android“ įrenginį oficialiai negali būti įdiegiama. Tam tikros vartotojų grupės gali būti suinteresuotos privilegijuotos kontrolės perėmimu siekdami apeiti šį gamintojų apribojimą ir naudotis piratine įranga.

Turėdami administratoriaus lygio sistemines privilegijas vartotojai gali įdiegti mobiliąsias programas su gamintojų nenumatytu funkcionalumu. Pavyzdžiui, programa gali keisti žadintuvo garso stiprumą priklausomai nuo įvairių aplinkybių: foninio garso triukšmo, kameros duomenų analizės ar akcelerometro parodymų.

Rezervinių kopijų darymo programinė įranga efektyviausiai veikia tuomet, kai turi administratoriaus lygio sisteminį priėjimą prie failų ir kitų duomenų. Tokia programinė įranga gali lengvai perkelti visus duomenis iš seno telefono į naują arba atstatyti buvusią mobiliosios programos versiją.

Mobiliųjų paslaugų tiekėjai dažnai „pririša“ parduodamą telefoną prie savo teikiamų telekomunikacijų paslaugų. Vartotojai, norintys pasinaudoti kito tiekėjo paslaugomis, paprastai tokios galimybės su pririštu telefonu neturi. Privilegijuotos kontrolės perėmimas gali padėti apeiti šiuos apribojimus.

Išmaniųjų telefonų gamintojai dažnai labai atsargiai žiūri į „Android“ versijų atnaujinimus pagrįstai baimindamiesi, jog atnaujinta operacinė sistema gali neigiamai paveikti sistemos ar

konkrečių programinių paketų stabilumą. Tačiau senos versijos taip pat yra labiau pažeidžiamos dėl susikaupusių, viešai žinomų informacinės saugos pažeidimų. Vartotojai gali nenorėti taikytis prie gamintojo atnaujinimų ciklo ir perimti telefono atnaujinimo kontrolę į savo rankas.

Tarp kitų priežasčių taip pat galima paminėti poreikį įdiegti nestandartinį „Android“ branduolį (testavimo ar kitais sumetimais), reklamų blokavimą, baterijos išnaudojimo optimizavimą, tinklo ugniasienės diegimą ir kt.

1.3.3. Administracinio vartotojo kontrolės perėmimo metodai ir įrankiai

„Android“ administracinio vartotojo kontrolės perėmimo procedūra skiriasi priklausomai nuo kiekvieno įrenginio. Daugumoje atvejų perėmimo procedūra, veikianti vienam įrenginiui, neveiks kitame. Taip pat verta atkreipti dėmesį, kad kiekvienam įrenginiui gali egzistuoti daugiau nei vienas būdas perimti administracinio vartotojo kontrolę. [7]

Egzistuoja dvi esminės administracinio vartotojo kontrolės perėmimo būdų grupės: modifikuoto „Android“ operacinės sistemos atvaizdžio (angl. *image*) paleidimas per įrenginio paleidyklę (angl. *bootloader*), bei kontrolės perėmimas pasinaudojant įdiegtos operacinės sistemos ar jos komponentų saugumo spragomis.

Įjungus „Android“ įrenginį, paleidyklė yra pirmoji aktyvuojama sisteminė programa. Paleidyklės procesas paprastai susideda iš dviejų dalių – pirminės ir antrinės paleidyklės. Daugelyje „Android“ įrenginių pirminės paleidyklės kodo keisti nėra galimybės, kadangi jos kodas yra įdiegtas ASIC (angl. *Application-Specific Integrated Circuit*) integruotos mikroschemos lygmenyje. Šiame lygmenyje esančios kompiuterinės instrukcijos į atmintį užkrauna antrinę paleidyklę bei nurodo jai, kaip galima pasiekti sistemos atmintį ir procesorių, bei kaip užkrauti pačią operacinę sistemą [8]. Turėdami galimybę keisti antrinės paleidyklės konfigūraciją, vartotojai gali į mobilų įrenginį įdiegti bet kokį pasirinktą operacinės sistemos atvaizdį, kuriame dažniausiai jau būna vartotojui prieinamas administracinis vartotojas „*root*“.

Daugelis gamintojų bei paslaugų tiekėjų (pavyzdžiui, JAV telekomunikacijų milžinė „AT&T) vartotojams nesuteikia prieigos teisių prie paleidyklių siekdami užtikrinti, jog vartotojai naudosis tik originaliai įdiegta operacine sistema ir tik gamintojo patvirtintomis mobiliosiomis programomis. Kita dalis gamintojų (pvz. „Google Nexus“ serija, gamintojo „HTC“ įrenginiai) siekia patraukti mobiliųjų entuziastų bei kūrėjų dėmesį atvirai suteikdami galimybę keisti antrinės paleidykės nustatymus.

Kitas būdas perimti administracinio vartotojo kontrolę paremtas pačios operacinės sistemos ar jos komponentų saugumo spragų išnaudojimu. *Zhang ir kt.* [9] apibrėžia keturis esminius „Android“ architektūros sluoksnius, per kuriuose esančius pažeidžiamumus gali būti vykdomos administracinio vartotojo kontrolės perėmimas. Kiekvieno sluoksnio saugumo spragų išnaudojimo atakai prieš įrenginio gamintojo įdiegtus technologinius apribojimus specifika smarkiai skiriasi.

1) **Linux branduolys**. Tai yra centrinė sistemos dalis su privilegijuota prieiga prie failų, procesų ir kitų resursų, todėl sėkmingos kibernetinės atakos prieš branduolį beveik garantuotai suteiks pilną „Android“ sistemos kontrolę. Branduolio pažeidžiamumai dažnai suteikia galimybę perimti kontrolę visuose tą branduolio versiją naudojančiuose įrenginiuose nepriklausomai nuo gamintojo ar įrenginio modelio.
2) **Gamintojo branduolys arba tvarkyklė**. Gamintojai gali modifikuoti standartinį „Android“ branduolį pagal savo poreikius, arba pateikti specifines sistemines tvarkykles (angl. *device drivers*). Toks kodas paprastai būna vykdomas branduolio erdvėje, kas reiškia, jog sėkmingos atakos prieš kodo pažeidžiamumus taip pat reikštų pilną sistemos kontrolę. Mobiliųjų įrenginių gamintojai dažnai tokius komponentus programuoja be išorinio audito, nepateikdami kodo viešam įvertinimui. Dėl to išauga saugos pažeidžiamumų tikimybė. Tačiau tokie pažeidžiamumai gali būti išnaudojami tik konkrečiuose to gamintojo modeliuose (pavyzdžiui, „Samsung“), kai tuo tarpu aukščiau paminėti bendri „Android“ branduolio pažeidžiamumai kelia grėsmę visiems tos „Android“ versijos įrenginiams.
3) **Sisteminės bibliotekos**. Bazinės „Android“ bibliotekos arba išorinių gamintojų papildomos programinės bibliotekos naudojamos mobiliųjų programų funkcionalumo palaikymui. Saugumo pažeidimai tokiose bibliotekose gali reikšti, kad bet kuri jas naudojanti programa

taip pat bus pažeidžiama. Jei bent viena tokia programa yra vykdoma privilegijuotomis administracinio vartotojo teisėmis, ataka prieš tokią programą vėlgi reikš pilną sistemos kontrolės perėmimą.

4) **Mobiliosios programos ir jų karkasas**. Šiame sluoksnyje vykdomos atakos pasinaudoja saugumo spragomis konkrečiose mobiliosiose programose, kurios turi teisę vykdyti kai kurias sistemines komandas administracinio vartotojo teisėmis (pvz. „*setuid*“ operacijos). Tokie pažeidžiamumai yra pakankamai reti ir įmanomi tik tuose įrenginiuose, kuriuose gamintojas įdiegė papildomą funkcionalumą, kuriam reikalinga privilegijuota prieiga ir kuriam nebuvo įdiegti pakankami saugos užtikrinimo mechanizmai.

Ball [10] identifikuoja šiuos pagrindinius atakų tipus, kuriais pasinaudojus dažniausiai perimama administracinio vartotojo kontrolė:

- „Exploid“;
- „RageAgainstTheCage“ (RATC);
- „Gingerbreak“;
- „Asroot“;
- „KillingInTheNameOf“;
- „zergRush“.

Šiomis spragomis naudojasi tiek kenkėjiška programinė įranga be vartotojo žinios, tiek patys vartotojai, norėdami be gamintojo žinios ar leidimo perimti sistemos valdymą. Apie 36% „Android“ sistemos virusų bandys pasinaudoti šiomis spragomis kaip pagrindiniu užkrato vektoriumi, o iš jų per 80 proc. bandys naudotis daugiau nei vienu aukščiau paminėtu metodu [10].

Mobilaus įrenginio savininkai, savo noru ketindami perimti sistemos kontrolę, gali rinktis iš plataus spektro programinės įrangos įrankių. *Zhang ir kt.* [9] pateikia sąrašą populiariausių tam skirtų paketų.

1.2 lentelė. Administracinio vartotojo perėmimo paketų apžvalga [9]

Pavadinimas	**Komponentai**	**Įrenginių modelių palaikymas**
Root Genius	Asm. kompiuterio programa, mobili programa	20 000+
360 Root	Asm. kompiuterio programa, mobili programa	20 000+
IRoot	Asm. kompiuterio programa, mobili programa	10 000+
KingRoot	Asm. kompiuterio programa, mobili programa	10 000+
SRSRoot	Asm. kompiuterio programa	7 000+
Baidu Root	Asm. kompiuterio programa, mobili programa	6 000+
Root Master	Asm. kompiuterio programa, mobili programa	5 000+
Towelroot	Mobili programa	-
Framaroot	Mobili programa	-

Šiame sąraše esančių paketų bei jų paliekamų pėdsakų „Android“ operacinėje sistemoje analizė bus svarbi formuluojant aptikimo metodą kituose skyriuose.

1.3.4. Administracinio vartotojo kontrolės perėmimo pasekmės sistemos saugumui

Šiame skyriuje apžvelgiamos saugumo grėsmės, kurias sukelia įrenginio savininko savanoriškas administracinio vartotojo „*root*“ perėmimas išmaniajame telefone [6].

Visų pirma, programa, kuriai leidžiama veikti privilegijuotomis teisėmis, išeina iš standartiškai numatytos izoliuotos aplinkos (angl. *sandbox*), t.y. visos saugumo modelio jai suteiktos prieigos teisės prie sisteminių resursų ar funkcionalumo nebetenka prasmės. Tokia mobilioji programa gali atlikti bet kokius norimus veiksmus, įskaitant prieigą prie kitų programų failų, įrenginio savininko veiksmų stebėjimą, mikrofono naudojimą ir kt.

Antra, programos, skirtos privilegijuotos kontrolės perėmimui, paprastai būna sukuriamos mėgėjų programuotojų, todėl labai tikėtina, kad joje bus saugumo spragų. Tuo gali pasinaudoti

piktavališka programinė įranga, perimdama administracinį sistemos valdymą be vartotojo žinios ar sutikimo.

Trečia, privilegijuotos kontrolės perėmimo metu telefono konfigūracija smarkiai keičiama, kuo gali pasinaudoti programišiai ar automatizuota kenkėjiška programinė įranga, pavyzdžiui, kompiuteriniai kirminai. Vienas pavyzdžių – tokiame telefone gali būti automatiškai įjungiama SSH paslauga, teikianti nuotolinės prieigos prie įrenginio galimybę. Prijungtas prie interneto ryšio telefonas tokiu būdu gali būti piktavališkai perimtas nuotoliniu būdu.

Kitas svarbus saugumo pažeidimo scenarijus – kai administracinio vartotojo perėmimas įvykdomas kenkėjiškos programos be vartotojo žinios. Trojos arklio tipo virusų užkrėsti mobilūs įrenginiai gali be vartotojo žinios siųsti SMS žinutes į apmokamus numerius, perimti balso ar tekstinių pokalbių turinį, arba tapti užkrėstų kompiuterių tinklo (angl. *botnet*) dalimi ir vykdyti įvairias atakas prieš kitus įrenginius tinkle.

Taigi akivaizdu, kad išmaniųjų telefonų su perimtomis administracinio vartotojo teisėmis saugumo rizika smarkiai išauga palyginti su standartiniu gamykliniu įrenginiu.

1.4. Esamų AVK problemos sprendimo metodų bei įrankių analizė

Privilegijuotos kontrolės perėmimo problema yra pakankamai naujas fenomenas mobiliųjų įrenginių saugos srityje. Mobiliosios saugos produktų gamintoja „Lookout“ atliko tyrimą, skirtą identifikuoti didžiausias grėsmes išmaniesiems telefonams. Jo metu buvo nustatyta, kad privilegijuotos kontrolės perėmimo programinė įranga yra vienas iš trijų rimčiausių saugumo iššūkių (kartu su Trojos arklio tipo virusais bei šnipinėjimo programine įranga) [11].

Šiame analizės skyriuje apžvelgiami standartiniai prevencijos būdai, kuriais mobiliųjų įrenginių gamintojai apsaugo vartotojus nuo potencialių saugumo spragų, susijusių su perimta administracinio vartotojo kontrole. Tuomet nagrinėjama akademinė literatūra ieškant egzistuojančių AVK aplinkos aptikimo metodų. Galiausiai analizuojami komerciniai ir kiti viešai prieinami programiniai produktai, teikiantys perimtos AVK požymių aptikimo funkcionalumą. Tiek teoriniams metodams, tiek egzistuojantiems produktams pateikiamos palyginimo lentelės pagal pasirinktus kriterijus.

1.4.1. AVK perėmimo prevencijos būdai

„Android“ operacine sistema paremtus mobiliuosius įrenginius gamintojai platina su ženkliais saugumo apribojimais. Įrenginio savininko procesai ir įdiegtos programos sistemoje leidžiamos neprivilegijuotomis paskyromis (angl. *unprivileged user accounts*), be specialaus įsikišimo ar ekspertinių žinių tipinis vartotojas neturi galimybės pasinaudoti administracinio vartotojo *root* privilegijomis.

Gamintojai taip pat imasi papildomų priemonių užkirsti kelią administracinio vartotojo kontrolės perėmimui iš operacinės sistemos išorės, pavyzdžiui, neleisdami keisti sisteminės paleidyklės (angl. *bootloader*) nustatymų bei uždrausdami priėjimą prie specialios kietojo disko vietos, kur saugoma informacija apie paleidykles bei „Android“ branduolį. Kai kuriose mobiliųjų telefonų serijose (pvz. „Google Nexus“ [12]) paleidyklę galima atrakinti su gamintojo leidimu be jokio papildomo saugumo spragų išnaudojimo, tačiau tam irgi reikia bazinių techninių žinių.

Vartotojams taip pat kur kas sunkiau būna perimti administracinio vartotojo kontrolę tuose įrenginiuose, kur tyčia išjungta USB derinimo (angl. *USB debugging*) funkcija, kuri paprastai reikalinga norint pasinaudoti „Android“ derintuvo „ADB“ paketu.

Visos šios paminėtos gamintojo apsaugos priemonės sumažina riziką, kad techniškai nesigaudantis įrenginio savininkas pažeis sistemos saugumo profilį atverdamas administracinio vartotojo prieigą naujai diegiamoms programoms. Tačiau bent kiek labiau patyrę ar apsiskaitę vartotojai tokias pirmines apsaugas apeis pasinaudodami 1.3.3 skyriuje aptartais būdais.

1.4.2. Perimtos AVK aplinkos aptikimo metodai akademinėje literatūroje

Šiame skyriuje pateikiama akademinės literatūros, susijusios su administracinio vartotojo kontrolės perėmimu, apžvalga. Svarbu pabrėžti, kad analizės metu nebuvo rastas nei vienas metodas, kurio pagrindinis tikslas būtų perimtos administracinio vartotojo kontrolės simptomų identifikavimas. Ši tema paliečiama daugelyje nagrinėtų darbų, bet dažniausiai kaip paviršutiniška, papildoma informacija. Kita vertus, šios literatūros analizė suteikė svarbių įžvalgų, kurios padėjo suformuluoti visavertį sprendimo metodą, aprašomą tolimesniuose šio magistrinio projekto skyriuose.

Ham ir kt. [13] siūlo sisteminių įvykių stebėjimu paremtą metodą, kuris aptiktų kenkėjiškos programos bandymus perimti administracinio vartotojo kontrolę. Metodas įgyvendinamas paleidus aktyvų stebėjimo procesą „Android“ branduolio fone. Šis procesas reguliariai tikrintų sisteminio bei įdiegtų mobiliųjų programų aktyvumo žurnalinius įrašus ir ieškotų įtartinų požymių. Tačiau toks metodas gali aptikti tik aktyvius bandymus perimti kontrolę realiu laiku. Tai būtų efektyvi priemonė bandant apsisaugoti nuo specifinių kenkėjiškų programų (pavyzdžiui, mobiliųjų virusų), bet nėra tinkamas diagnozuoti perimtai administracinio vartotojo aplinkai. Jei metodu paremto sprendimo aktyvavimo metu „Android“ operacinė sistema jau buvo nulaužta anksčiau, metodas to neparodys.

Jindal [14] nagrinėja asmeninių mobiliųjų įrenginių naudojimo darbinėje aplinkoje (angl. *Bring Your Own Device*) bendrą saugumo problematiką ir kaip vieną iš būtinų saugos užtikrinimo aspektų paliečia perimto administracinio vartotojo aptikimo temą. Nors autorius identifikuoja net penkis aptikimo metodus, išties trys iš jų tėra elementarūs testai, tikrinantys, ar sistemoje egzistuoja administracinio vartotojo prieigai reikalinga komanda „su“ bei „Android“ programinis paketas „Superuser.apk“. Kiti du paminėti metodai skirti aptikti sisteminės paleidyklės saugumo nustatymams, pvz. ar „S-OFF“ parametras yra įjungtas. Tai išties gali būti simptomas, jog įrenginyje yra perimta administracinio vartotojo kontrolė, tačiau šis metodas siūlomas tik „HTC“ gamintojo įrenginiams, be to nėra lengvai automatizuojamas iš pačios operacinės sistemos vidaus. Be to, tokiam metodui reikalingas aktyvus žmogaus įsikišimas ir jį gali būti sunku automatizuoti.

Zhang ir kt. [9] nagrinėja legalių administracinio vartotojo kontrolės perėmimo produktų veikimo būdus bei jų saugos pažeidžiamumus, kuriais potencialiai gali pasinaudoti kenkėjiškos įrangos kūrėjai. Savo darbe autoriai taip pat paliečia klausimą, ar populiariausios antivirusinės programos, skirtos mobiliesiems įrenginiams, aptinka ir identifikuoja AVK perėmimo paketus kaip įtartinus. Bandymų metu autoriai konstatuoja, kad antivirusinė įranga nėra efektyvi priemonė šiam tikslui pasiekti.

Ball [10] savo darbe koncentruojasi į identifikavimą kenkėjiškos programinės įrangos, kuri bando perimti sistemos kontrolę pasinaudodama žinomais, AVK perėmimui skirtais „Android“ platformos pažeidžiamumais. Metodas pagrįstas dinamine sistemos įvykių analize, t.y. kenkėjišką įrangą bandoma aptikti pasitelkiant aktyvius sisteminius sensorius, veikiančius sistemos bei tinklo lygmenyje. Nors autorius daugiausiai nagrinėja „Android“ virusų atakos principus, savo darbe taip pat pamini ir kai kuriuos failų sistemos lygio simptomus: pvz., „/system“ disko skirsniui suteiktos rašymo teisės.

1.3 lentelė. Perimto administracinio vartotojo aplinkos aptikimą taikantys metodai

Palyginimo kriterijus	**Ham ir kt.**	**Jindal**	**Zhang ir kt.**	**Ball**
Dinaminis ar statinis metodas	Dinaminis	Statinis	Statinis	Dinaminis
Pilnai automatizuojamas	Taip	Ne	Taip	Taip
Antivirusinės įrangos pasitelkimas	Ne	Ne	Taip	Taip
Failų sistemos simptomų paieška	Ne	Taip	Ne	Taip
Skirtas AVK požymių aptikimui	Ne	Ne	Ne	Ne

Iš lentelėje pateiktų kriterijų palyginimo matyti, kad nei vienas metodas negali pasiūlyti visaverčio sprendimo, kuris atitiktų pagrindinį šiame darbe iškeltą tikslą sukurti AVK požymių aptikimo algoritmą.

1.4.3. Komerciniai perimtos AVK aplinkos aptikimo įrankiai ir produktai

Kartu su akademinės literatūros apžvalga, išdėstyta prieš tai buvusiame skyriuje, buvo atlikta išsami egzistuojančių komercinių (tiek mokamų, tiek nemokamų), perimtos AVK aplinkai aptikti skirtų produktų apžvalga. Žemiau apžvelgiami ir tarpusavyje palyginami populiariausi produktai, tiesiogiai ar netiesiogiai teikiantys administracinio vartotojo būsenos identifikavimo funkcionalumą.

„Google Play“ skaitmeninėje prekyvietėje galima rasti daug mobiliųjų programų, kurios skelbiasi galinčios identifikuoti administracinio vartotojo aplinką telefone. Šios programos paremtos ypač supaprastintais testais, kurie dažniausiai apsiriboja „su“ sisteminės komandos bei „Superuser.apk“ paketo telefone paieška ir bandymu juos aktyvuoti. Jokių kitų požymių šios programos neieško, kadangi jų tikslas nėra saugumo būklės įvertinimas. Tokių programų nauda apsiriboja tuo, kad savininkas yra informuojamas, jei telefono kontrolė buvo perimta standartiniais perėmimo paketais. Įvedus paiešką su raktiniais žodžiais „root checker“ randama per dešimt tokių programų, kurios iš esmės tėra viena kitos kopijos.

„RootTools“ yra populiarus „Android“ programų kūrėjams skirtas bibliotekų paketas, suteikiantis standartizuotus programinius metodus administracinio vartotojo aplinkos funkcionalumui įgyvendinti [15]. Naudojantis šiuo paketu galima atlikti tokius veiksmus, kaip perimtos administracinio vartotojo aplinkos patikrinimas, sisteminių programų paieška, administracinių privilegijų paėmimas. Darbo eigoje šis kodas buvo išanalizuotas ir buvo nustatyta, jog testai, skirti perimtos AVK aplinkai identifikuoti, yra pakankamai skurdūs ir neišbaigti. Paketas tiesiog tikrina dvejetainių sisteminių komandų „su“ bei „busybox“ egzistavimą, bei paleidžia „id“ komandą siekdamas nustatyti aktyvaus vartotojo sisteminį numerį.

Kompanija „Samsung“ 2013 metais rinkai pasiūlė saugumo sprendimą „KNOX“. Šis produktas suteikia galimybę vartotojo nuosavame telefone įdiegti atskirą virtualų konteinerį, skirtą išimtinai darbui su verslo programomis ir aplinka [16]. Vartotojas gali lengvai „šokinėti“ tarp savo standartinės namų aplinkos ir verslui skirtos aplinkos, nors šios yra tarpusavyje visiškai izoliuotos. Tarp daugelio šio produkto teikiamų saugumo funkcijų yra ir galimybė aptikti perimto administracinio vartotojo kontrolės aplinką, tačiau kompanija neatskleidžia techninių detalių, kaip tai yra įgyvendinama. „KNOX“ programai aptikus, jog telefone šiuo metu yra įdiegtas arba kažkada praeityje buvo įdiegtas nestandartinis, gamintojo nepatvirtintas operacinės sistemos atvaizdis, automatiškai pakeičiama aparatūrinės įrangos lygyje esančio saugiklio „*e-fuse*“ reikšmė iš 0 į 1 [17]. Telefonai su tokiu saugiklio nustatymu nebetenka garantinio aptarnavimo paslaugos, o „KNOX“ virtuali aplinka tokiuose įrenginiuose apskritai atsisako veikti. Atlikus šio produkto analizę konstatuota, kad toks funkcionalumas, nors ir efektyvus specifiniais atvejais, yra pernelyg invazinis atsižvelgiant į šiame darbe kuriamam metodui iškeltus reikalavimus. Taip pat svarbu pažymėti, jog šis produktas veikia tik ant „Samsung“ telefonų ir yra pakankamai brangus sprendimas, todėl negali būti laikomas universaliu metodu.

Žymios virtualizacijos produktų kompanijos „VMWare“ padalinys „AirWatch“ teikia sprendimus verslo klientams centralizuoto mobiliųjų įrenginių valdymo (angl. MDM, *Mobile Device Management*) srityje. „AirWatch“ platforma suteikia galimybę koordinuoti ir valdyti tiek verslui priklausančius, tiek asmeninius darbuotojų mobiliuosius įrenginius, naudojamus įmonės tinkle; palengvina konfigūracijos sinchronizavimą, įdiegia saugos kontrolės mechanizmus. Vienas iš daugelio saugumo patikros testų, įdiegtų šiame produkte, taip pat bando nustatyti prie sistemos naujai pridedamo įrenginio administracinio vartotojo aplinkos būklę, t.y. ar „Android“ įrenginyje nėra perimta privilegijuota kontrolė [18]. Gamintojas dėl akivaizdžių komercinių sumetimų neatskleidžia konkrečių metodų ar algoritmų, kurių pagalba testuojama sistemos būklė, tačiau remiantis IT saugos ekspertų atlikta atvirkštinės inžinerijos analize [19] [20] galima spręsti, jog testai apsiriboja statiniais standartinių požymių patikrinimais (komandos „su“ bei „id“ paleidimo rezultatų tikrinimas, įtartinų „Android“ mobiliųjų programų paketų identifikavimas). Kadangi „AirWatch“ yra uždaro kodo programinė įranga ir yra komercinis, į atskirus komponentus nedalomas produktas, jo panaudojimas kaip universalaus AVK aptikimo metodo būtų labai ribotas.

1.4 lentelė. Komerciniai ir nekomerciniai produktai, teikiantys AVK aptikimo funkcionalumą

Palyginimo kriterijus	**RootChecker šeimos mobiliosios programos**	**RootTools**	**Samsung Knox**	**VMWare AirWatch**
Atviro kodo programinė įranga	Ne	Taip	Ne	Ne
Sisteminės programos „su“ identifikavimas	Taip	Taip	Tikėtina	Taip
Sisteminės komandos „id“ rezultato sulyginimas su reikšme „0“	Ne	Taip	Tikėtina	Taip
Įtartinų paketų, pvz. „Superuser.apk“, identifikavimas	Taip	Ne	Tikėtina	Tikėtina
Kitų požymių identifikavimas (nestandartinės bibliotekos, prieigos teisės, tinklo nustatymai, t.t.)	Ne	Ne	Tikėtina	Tikėtina
Požymių aptikimo būdas	Statinis	Statinis	Statinis ir dinaminis	Statinis
Invaziniai pakeitimai aparatūros lygmenyje	Ne	Ne	Taip	Ne
Tinka visų gamintojų įrenginiams	Taip	Taip	Ne	Taip
Gali būti panaudotas kaip kitos programos modulis	Ne	Taip	Ne	Ne

Šioje lentelėje pateikiama nagrinėtų produktų apžvalga pagal kriterijus, kurie būtų svarbūs kuriant universalų AVK aplinkos aptikimo produktą. Arčiausiai siekiamo tikslo yra viešai prieinama Java kalba parašyta programinė biblioteka „RootTools“, tačiau jos teikiamas AVK požymių aptikimo funkcionalumas yra labai ribotas (2-3 elementarūs testai).

1.5. Darbo tikslas, uždaviniai, planas ir siekiami privalumai

Šio darbo *tikslas* yra sukurti universalų metodą ir juo paremtą prototipą, skirtą aptikti perimtos administracinio vartotojo kontrolės aplinką „Android“ operacinėse sistemose.

Šio darbo *uždaviniai*:
- išanalizuoti egzistuojančius privilegijuotos kontrolės *perėmimo* būdus bei populiariausių tam skirtų įrankių veikimo principus;
- išanalizuoti egzistuojančius privilegijuotos kontrolės *aptikimo* metodus ir tam skirtus komercinius produktus;
- nustatyti ir palyginti aptikimo metodų / įrankių privalumus bei trūkumus;
- sukurti ir pasiūlyti visavertį perimtos administracinio vartotojo kontrolės aplinkos aptikimo metodą;
- metodo pagrindu sukurti prototipą;
- eksperimentiškai įvertinti metodo bei prototipo efektyvumą.

Sėkmingai įgyvendinus užsibrėžtus tikslus bus pasiekti šie *privalumai*:
- akademinei bendruomenei bus suteikta naujausia apibendrinta informacija apie šiuolaikinius AVK perėmimo ir aptikimo būdus;

- mobiliųjų programų kūrėjams bus pateiktas realiai veikiantis Java kalbos pagrindu sukurtas AVK aptikimo modulis.

1.6. Siekiamo sprendimo apibrėžimas

Sprendimui iškeliami šie papildomi tikslai:

- sprendimas skirtas atpažinti statinius perimtos AVK požymius, egzistuojančius šiuo metodu apsaugotos programos paleidimo metu;
- metodo pagrindu įgyvendintas prototipas gali būti naudojamas kaip atskiras komponentas bet kurioje mobiliojoje programoje, nepriklausomai nuo operacinės sistemos versijos, įrenginio gamintojo arba įrenginio modelio;
- sprendimas neskirtas stebėti dinaminių pasikeitimų operacinėje sistemoje realiu laiku, pavyzdžiui, diegiant sisteminio žurnalo stebėsenos sensorius;
- sprendimas skirtas atpažinti požymius, paliktus tiek kenkėjiškų programų (mobiliųjų virusų), tiek vartotojų savanoriškai atliktų AVK perėmimo veiksmų;
- sprendimas nėra skirtas aptikti kitus kenkėjiškų programų požymius, nesusijusius su AVK kontrolės perėmimu;
- sprendimas neįgyvendins papildomos apsaugos nuo atvirkštinės inžinerijos, kuri gali būti vykdoma siekiant apeiti šio sprendimo atliekamus AVK požymių aptikimo testus;
- sprendimas kuriamas remiantis esmine prielaida, kad vartotoju, kurio mobiliajame įrenginyje paleidžiamas metodu paremtas prototipas, nėra pasitikima – t.y. algoritmo vykdymo metu jokia papildoma informacija iš įrenginio savininko nėra užklausiama.

1.7. Analizės išvados

Analitinėje dalyje buvo apžvelgtas administracinio vartotojo „*root*“ kontrolės perėmimo procesas „Android“ platforma paremtuose mobiliuosiuose įrenginiuose, taip pat identifikuotos esminės su šiuo procesu susijusios neigiamos pasekmės įrenginio saugai. Įvairūs šaltiniai tiek akademinėje literatūroje, tiek tarp mobiliosios saugos ekspertų akcentuoja, jog tai išties yra aktuali problema. Tą faktą patvirtina ir komercinių produktų įvairovė. Kita vertus, kadangi mobiliųjų įrenginių sauga (ir tuo pačiu perimto administracinio vartotojo kontrolės problematika) yra palyginus nauja tema, šiuo metu nėra nei vieno išbaigto metodo, skirto visavertiškai įvertinti visus perimtos AVK „Android“ operacinėje sistemoje požymius.

Mokslinėje literatūroje daugiau koncentruojamasi į kenkėjiškos programinės įrangos aptikimą dinaminiais metodais (pvz., aktyvus procesas stebi sistemos žurnalo įrašus arba failų sistemos pokyčius) ir mažai dėmesio skiriama tiems atvejais, kai apsauginė programa yra įdiegiama po fakto – t.y. jau po to, kai įrenginyje su ar be savininko žinios yra perimama administracinio vartotojo kontrolė. Net ir tais atvejais, kai užsimenama apie perimtos AVK problemą, tai būna daugiau šalutinis ir tik paviršutiniškai nagrinėjamas uždavinys.

Mobiliųjų programų rinkoje (tiek „Google Play“ virtualioje prekyvietėje, tiek didžiųjų gamintojų siūlomų produktų asortimente) taip pat nebuvo rasta nei vieno pilnaverčio produkto, atliekančio išsamų „Android“ operacinės sistemos auditą konkrečiai ieškant perimtos AVK požymių. Yra nemokamų ar atviro kodo produktų, skirtų tik šiam tikslui, tačiau jų siūlomas funkcionalumas yra labai primityviame lygyje ir dažniausiai susideda iš dviejų ar trijų elementarių testų, kuriuos lengva nuspėti ir kurių lengva išvengti. Šie testai dažnai neatspindi tikrosios įrenginio būklės, be to gali būti lengvai apeinami AVK perėmimui skirtų programų. Tuo tarpu komercinė programinė įranga (pavyzdžiui, „Samsung KNOX“ arba „VMWare AirWatch“), nors ir pateikdama platesnį technologinių testų rinkinį, yra uždaro kodo, ir jos pritaikymas yra apribotas tik konkrečiam produktui arba tinkamas tik ribotam mobilių įrenginių asortimentui. Tai reiškia, kad mobilių programų kūrėjams pasinaudoti šiuo funkcionalumu kuriant savo programas dažnai nėra galimybių.

Analizės metu buvo patvirtinta darbo pradžioje iškelta prielaida, jog mobiliųjų įrenginių rinkoje egzistuoja perimtos AVK identifikavimo poreikis, tačiau šiuo metu nėra nei vieno pilnaverčio metodo ar produkto. Kadangi nei akademinėje literatūroje, nei tarp komercinių mobiliųjų produktų nebuvo rastas universaliai panaudojamas, viešai prieinamas ir lengvai su mobiliosiomis programomis integruojamas perimtos AVK aptikimo metodas, šiame magistriniame darbe toliau dirbama ties tuo, kaip užpildyti šią trūkstamo saugumo sprendimo nišą.

2. PERIMTOS ADMINISTRACINIO VARTOTOJO KONTROLĖS „ANDROID“ PLATFORMOJE APTIKIMO METODAS

Analizės etapo metu nebuvo rastas nei vienas visavertis teorinis metodas ar praktiškai realizuotas komercinis produktas, kuris užtikrintų perimtos administracinio vartotojo kontrolės „Android“ platformoje aptikimą pagal visas požymių kategorijas ir būtų panaudojamas nepriklausomai nuo konkretaus įrenginio, gamintojo ar produkto. Šiame darbe siūlomo metodo tikslas yra apjungti egzistuojančius patikros testus (pavyzdžiui, sisteminės komandos „su“ paieška) su visu spektru iki šiol nenaudotų ar literatūroje neaprašytų testų (pavyzdžiui, specifiniai tinklo konfigūracijos nustatymai).

2.1. Perimtos administracinio vartotojo kontrolės aptikimo metodo reikalavimų apibrėžimas

Šiame skyriuje pateikiamas esminių reikalavimų sąrašas, padėsiantis lengviau suformuluoti ir suvokti toliau pateikiamo metodo aprašymą.

2.1.1. Funkciniai reikalavimai kuriamam metodui

Funkciniai reikalavimai apibrėžia kuriamos sistemos galimybes, t.y. ką ji konkrečiai atlieka ir kokia jos teikiamų funkcijų nauda:

- metodas atliks seriją operacinės sistemos testų, skirtų aptikti perimtos administracinio vartotojo „*root*“ kontrolės požymius;
- testų rezultatams esant teigiamiems, bus sugeneruotas pranešimas (arba perduotas rezultato kodas kviečiančiajai programai), jog veikiama nesaugioje aplinkoje;
- testų rezultatams esant neigiamiems bus sugeneruotas pranešimas (arba perduotas rezultato kodas kviečiančiajai programai), jog veikiama saugioje aplinkoje;
- testai, skirti perimtai administracinio vartotojo kontrolei aptikti, turi būti paleidžiami kiekvieną kartą programai pradėjus veikti;
- testai, skirti perimtai administracinio vartotojo kontrolei aptikti, turi būti pakartotinai paleidžiami taikant atsitiktinės (t.y. neprognozuojamos potencialiam atakuotojui) trukmės intervalus tarp kiekvieno patikrinimo;
- esant nevienareikšmiškiems kriterijams (kurie gali egzistuoti tiek nulaužtoje, tiek nenulaužtoje sistemoje), ieškoma kombinacija dviejų ar daugiau kriterijų, kurių bendras egzistavimas įrodo nesaugią „Android“ operacinės sistemos aplinką;
- praktiškai realizuotame (suprogramuotame) metode galima įdiegti dinamiškai kuriamą ir konfigūruojamą saugumo profilį, aprašantį, kuriems testams pasitvirtinus programa turi nutraukti savo veikimą (preventatyvi apsauga), o kuriems pasitvirtinus programa turi informuoti centrinį serverį (aptikimo apsauga);
- saugumo konfigūravimo profilis turi būti užšifruojamas ir pasirašomas centriniame serveryje prieš perduodant jį vartotojo įrenginyje veikiančiai programai, kuri savo ruožtu turi sugebėti parašą patikrinti ir iššifruoti duomenis;
- aptikimo apsaugos mechanizmai turi pranešti centriniam serveriui apie nestandartinius testuojamos sistemos parametrus, kad būtų galima nuolatos tobulinti apsaugines taisykles;
- preventatyvios apsaugos mechanizmai turi neleisti programai veikti toliau, jei yra aptinkami operacinės sistemos nulaužimo įrodymai.

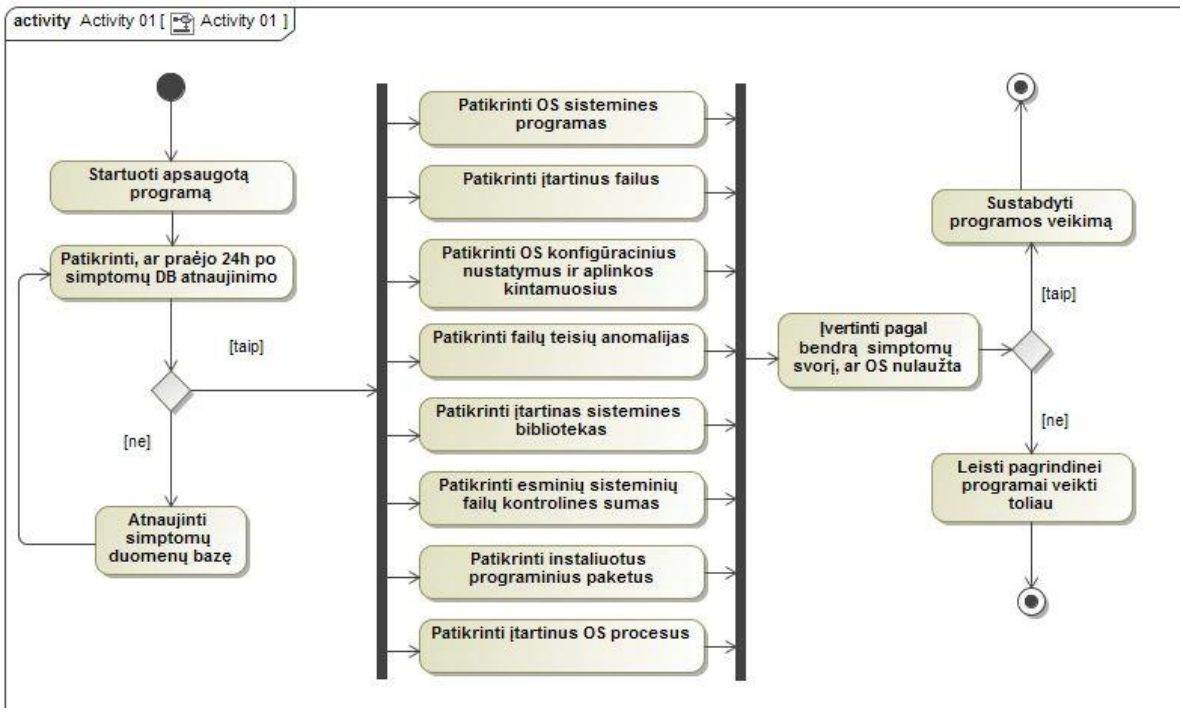
2.1.2. Nefunkciniai reikalavimai kuriamam metodui

Nefunkciniai reikalavimai aprašo apsauginės sistemos veikimo charakteristikas, o ne konkrečias jos atliekamas funkcijas. Šiame skyriuje aprašoma ideali kombinacija visų nefunkcinių reikalavimų, kurie užtikrintų maksimalų metodo pagrindu sukurtos realizacijos efektyvumą. Kadangi šio magistrinio darbo tikslas apsiriboja perimtos administracinio vartotojo kontrolės požymių identifikavimu, ne visi teoriniai nefunkciniai reikalavimai bus diegiami prototipo kūrimo etape.

- Testai turi būti sudaryti iš statinių testų, integruotų į pačią mobiliąją programą, ir tuo pačiu turi palaikyti dinamiškai nustatomus saugumo politikos tikrinimus programos paleidimo ir veikimo metu;
- praktiškai realizuotame (suprogramuotame) metode testavimo funkcijos neturi būti vadinamos akivaizdžiais vardais (pavyzdžiui, „tikrintiNulauzima()“), kad potencialus atakuotojas negalėtų lengvai identifikuoti funkcijos tikslo;
- taikoma taktika tyčia įdiegti netikrą programos kodą su nulaužtos sistemos paieška imituojančiais pavadinimais - taip papildomai supainiojama ir sulėtinama potenciali atvirkštinės inžinerijos analizė;
- programos siunčiami pranešimai privalo būti užšifruoti, ir, jei įmanoma, paslėpti nuo potencialių atakuotojų;
- apsaugos mechanizmai neturi atskleisti jokios specifinės informacijos apie nesaugios aplinkos aptikimą, t.y. jei programai duodamas signalas užbaigti veikimą, tai turi būti bendro pobūdžio klaidos pranešimas be jokių papildomų detalių;
- saugumo konfigūravimo profilį turi būti galima atnaujinti neperrašant ir neatnaujinant pačios programos kodo;
- metodas gali būti realizuotas kaip programinis modulis, lengvai integruojamas į bet kurią „Android“ mobiliąją programą, kuriai reikalingas perimtos AVK aplinkos aptikimo funcionalumas;
- tikrinamų sistemos failų pavadinimai, kurių pagalba vykdomi nulaužtos aplinkos aptikimo testai, turi būti užslaptinti kode ir atslaptinti tik prieš pat jų panaudojimą.

2.2. Metodo esmė

Metodas sudaromas remiantis prielaida, jog juo pagrįsta realizacija bus integruota į mobiliąsias „Android“ programas, kurioms reikalingas šis saugumo patikros funkcionalumas. Metodas bus paremtas seka testų, kurie turės visavertiškai ir vienareikšmiškai įvertinti operacinės sistemos būklę – ar buvo aptikta perimtos administracinio vartotojo kontrolės požymiai. Metodas grąžina vieną iš dviejų įmanomų reikšmių – taip (aptikta) arba ne (neaptikta). Žemiau pavaizduotoje diagramoje iliustruojama siūlomo metodo veiksmų eiga.



2.1 pav. Metodo algoritmo loginė diagrama (UML notacija)

Metodas numato, bet nebūtinai reikalauja, galimybę įdiegti dinamiškai atnaujinamą simptomų duomenų bazę, kaip vieną iš realizacijos komponentų. Tokiam atnaujinimui reikalinga interneto prieiga, kurios metodo pagalba apsaugota mobilioji programa gali ir neturėti. Todėl šis funkcionalumas siūlomas kaip papildomas, o ne esminis ar būtinas.

Metodas taip pat numato, net nebūtinai reikalauja, galimybę suteikti kiekvienam individualiam perimtos AVK požymiui arba požymių grupei atskirą svorį. Jei identifikuotų požymių svorių suma viršija iš anksto numatytą slenkstį, tuomet operacinė sistema identifikuojama kaip nulaužta. Jei svorių suma neviršija slenksčio, tačiau buvo aptiktas bent vienas požymis, operacinė sistema identifikuojama kaip įtartina. Šie rezultatai perduodami metodą kviečiančiai mobiliajai programai, ir pati programa jos autorių nuožiūra daro sprendimą, ar nutraukti, ar tęsti darbą. Naudojant svorių sistemą šiek tiek sumažinama rizika, kad sistema bus klaidingai identifikuota kaip nulaužta (I tipo testo klaida), tačiau svoriais paremto metodo realizacijos palaikymas tampa kur kas sudėtingesnis. Šio darbo tolimesniuose skyriuose aprašomi požymiai beveik visais atvejais vienareikšmiškai indikuoja perimtos AVK būseną, todėl svoriais ir slenksčiais paremtas funkcionalumas siūlomas kaip papildomas, o ne esminis ar būtinas.

Pagrindinė metodo dalis yra perimtos AVK aplinkos požymių paieška pagal įvairias požymių grupes. Šie testai gali būti vykdomi lygiagrečiai, nebent egzistuotų priklausomybė, jog konkretus testas gali būti vykdomas tik sulaukus kito testo pabaigos ir rezultato. Žemiau esančioje lentelėje pateikiama trumpa visų požymių grupių apžvalga. Detalesnė kiekvieno grupėje esančio individualaus požymio analizė pateikiama 2.3 skyriuje “Metodo detalizavimas”.

2.1 lentelė. Perimtos „Android“ administracinio vartotojo kontrolės požymių grupių apžvalga

Perimtos „Android“ AVK požymių grupė	**Požymio pavyzdys**
Įtartinos ar nestandartinės sisteminės komandos	/bin/su
Įtartina ar nestandartinė operacinės sistemos konfigūracija	android.os.Build.TAGS=test-keys
Su nulaužimo paketais komplektuojamos bibliotekos	libAndroidCydia.cy.so
Nulaužimui skirti mobiliųjų programų paketai	eu.chainfire.supersu
Nulaužimui skirtų mobiliųjų programų paketų direktorijos	/data/data/stericson.busybox
Įtartini mobiliųjų programų aktyvumo nustatymai	cyanogenmod.superuser
Failų ir direktorijų prieigos teisių anomalijos	/system/bin su 777 teisėmis
Operacinės sistemos procesai, kuriems reikalinga perimta AVK	sshd
Nestandartiniai ar trūkstami vidiniai saugumo sertifikatai	/etc/security/otacerts.zip
Įtartini atviri tinklo prievadai	TCP#80 (HTTP)

Lentelėje pateikiami tik pavieniai pavyzdžiai. Kitame skyriuje kiekviena požymių grupė aprašoma detaliau.

2.3. Metodo detalizavimas

Šiame skyriuje pateikiami konkretūs numatytų operacinės sistemos testų aprašymai, kartu pateikiant ir priežastį, kodėl vienas ar kitas simptomas leidžia numanyti, jog sistema nulaužta. Perimtos administracinio vartotojo kontrolės aplinkos požymiai buvo identifikuoti tiek naudojantis akademine literatūra, tiek šaltiniais internete, tiek eigoje eksperimentavimo su įvairiais AVK perėmimui skirtais programinės įrangos paketais.

Kuriant požymių tikrinimo algoritmą atskiri testai buvo sugrupuoti pagal požymio tipą. Pačios požymių testų grupės savo ruožtu pateiktos mažėjančios tikimybės tvarka. Kuo didesnė tikimybė testui grąžinti teigiamą rezultatą (t.y. patvirtinant kad aplinka nulaužta), tuo anksčiau stengiamasi tą testą įvykdyti.

Turint omenyje, kaip lengvai gali būti atgaminamas (angl. *decompile*) Java kalba parašytas kodas, ne kiekvieną tikrinimo testą reikėtų laikyti kode. Idealiu atveju tik failų egzistavimo patikrinimai ir bazinės sisteminės komandos turėtų būti paleidžiamos iš Java kodo. Visa likusi dauguma patikrinimų turėtų būti paversti dvejetainiu, sisteminiu kodu, kuris savo ruožtu būtų kviečiamas iš pagrindinės Java

programos per „JNI“ (angl. *Java Native Interface*) [21] sąsają. Užklausos į šį “išorinį” kodą turėtų būti slepiamos, kad nebūtų galima jų susieti su saugumo funkcijų veikla.

Dauguma žemiau aprašomų testų gali būti vykdomi vieną kartą, t.y. metodu apsaugotos mobiliosios programos paleidimo metu. Tačiau kai kurie testai, skirti dinaminiam sisteminių bibliotekų užkrovimui aptikti, turėtų būti pakartotinai paleidžiami neprognozuojamais intervalais programos veikimu metu. Testų, kurie turėtų būti vykdomi pakartotinai, aprašymuose bus atskirai paminėtas šitas reikalavimas.

2.3.1. Sisteminių komandų testai

Kiekviena „Android“ programa standartiškai paleidžiama atskiro, kiekvienai programai individualiai priskirto sisteminio vartotojo teisėmis. Akivaizdu, kad originalioje, nenulaužtoje sistemoje tokios programos niekada nebus paleistos administracinio vartotojo “*root*” teisėmis. Bandant paleisti sistemines komandas ar programas, kurios prieinamos tik “*root*” vartotojui, galima identifikuoti, ar programa vykdoma nulaužtoje operacinėje aplinkoje.

- Įvykdoma operacinės sistemos komanda „*id*“, nustatanti, kokio vartotojo teisėmis paleista tikrinimo programa, ir interpretuojamas jos atsakymas:

  Jei grąžinamas atsakymas yra „*0 (root)*“, tuomet procesas paleistas administracinio vartotojo teisėmis. Tai reiškia, kad administracinio vartotojo kontrolė perimta (t.y. sistema nulaužta).

- Įvykdoma operacinės sistemos komanda „*su*“, bandanti perimti administracinio vartotojo “*root*” teises, ir interpretuojamas jos rezultatas:

  Jei komanda įvykdoma sėkmingai arba gaunamas klaidos pranešimas „*permission denied*“, vadinasi įrenginys yra nulaužtas. Standartinėje sistemoje leidžiant „*su*“ turi būti grąžinamas pranešimas „*not found*“.

- Įvykdoma operacinės sistemos komanda „*busybox*“ bandant nustatyti šio paketo egzistavimą. Šis paketas paprastai yra įdiegiamas tam, kad po AVK perėmimo sistemoje būtų galimybė naudotis standartinėmis Linux administravimo priemonėmis [22].

  Jei komanda įvykdoma sėkmingai (nepaisant papildomų išvesties rezultatų), tuomet įrenginys yra nulaužtas.

- Gali būti atvejų, kai įtartina sisteminė komanda egzistuoja, tačiau nėra įtraukta į vartotojo „PATH“ aplinkos parametrą (angl. *environment variable*) ir dėl to nebus prieinama bandant ją tiesiogiai paleisti. Siekiant apeiti šį apribojimą, vykdoma šių komandų kaip paprastų failų paieška tiek standartinėse, tiek žinomose nestandartinėse direktorijose.

  Ieškomi failai:
  - su
  - mu
  - su-backup
  - suhappy
  - amphoras
  - daemonsu
  - we-need-root
  - busybox

  Ieškoma direktorijose:

- /system/bin/
- /system/xbin/
- /sbin/
- /system/su
- /system/bin/.ext/.su
- /system/usr/we-need-root/su-backup
- /system/xbin/mu
- /system/bin/

Jei randamas bent vienas iš paminėtų failų, tuomet įrenginys yra nulaužtas.

2.3.2. Sistemos konfigūracijos testai

Standartinėse, gamintojo paruoštose „Android“ konfigūracijose tam tikri parametrai visuomet būna pastovūs ir prognozuojami. Ši testų grupė paremta nestandartinių reikšmių paieška.

- Sistemos konfigūravimo faile „*./system/build.prop*“ ieškoma parametro „*ro.build.tags*“ reikšmė. Kitas būdas šiai reikšmei surasti yra kreipinys į „Android“ Java kintamąjį „*android.os.Build.TAGS*“.

  Jei parametro reikšmė nėra lygi „*release-keys*“, ir ypač tais atvejais, kai reikšmė tapati „*test-keys*“, daroma išvada, jog įrenginys nulaužtas.

- Papildomame sistemos konfigūravimo faile „*./data/local.prop*“ ieškoma parametrų „*ro.secure*“ ir „*ro.kernel.emu*“ reikšmių. Šie parametrai taip pat gali būti pasiekiami per sisteminę komandą „*getprop*“.

  Jei parametro „*ro.secure*“ reikšmė identifikuojama kaip *0*, didelė tikimybė, jog įrenginys nulaužtas. Jei parametro „*ro.kernel.emu*“ reikšmė lygi *1*, tuomet įrenginys yra emuliuojamas, kas reiškia, jog sistemos savininkas administraciniame lygmenyje gali kontroliuoti visus sistemos aspektus, net jei pati operacinė sistema nėra nulaužta.

2.3.3. Sisteminių bibliotekų testai

Kai kurie AVK perėmimui skirti programiniai paketai į operacinę sistemą įdiegia papildomas sistemines bibliotekas. Tokių bibliotekų egzistavimo faktas vienareikšmiškai liudija perimtos AVK būseną.

- Viena populiaresnių tokio tipo programų yra „Cydia Substrate“ [23], dažnai randama nulaužtose „Android“ sistemose. Jai aptikti vykdoma ši bibliotekų paieška:

  Ieškoma direktorijose:
  - /system/lib/
  - /vendor/lib/

  Ieškomos bibliotekos:
  - liblog!.so
  - libsubstrate.so
  - libsubstrate-dvm.so
  - libAndroidCydia.cy.so
  - libDalvikLoader.cy.so

  Radus bent vieną teigiamą paieškos rezultatą daroma išvada, jog įrenginys nulaužtas.

2.3.4. „Android“ programų paketų testai

Programiniai paketai, skirti „Android“ administracinio vartotojo kontrolei perimti bei jau egzistuojančios AVK būsenos funkcionalumui išplėsti, paprastai registruojami bendrame „Android“ programinių paketų sąraše. Peržiūrėti šį sąrašą galima įvairiais būdais:
- su vidine sistemine komanda „*pm list packages*“;
- peržiūrint direktorijos „*/system/app*“ turinį;
- peržiūrint užregistruotų paketų sąrašą iš Java aplinkos.

- Vykdoma paieška populiariausių AVK perėmimo bei jau perimtos aplinkos funkcionalumą išplečiančių paketų („SuperSU“, „OTA RootKeeper“, „Busybox“ ir kt.):
    - com.noshufou.android.su
    - eu.chainfire.supersu
    - com.jrummy.busybox.installer.pro
    - stericson.busybox.donate
    - com.jrummy.busybox.installer
    - stericson.busybox
    - com.stericson.rootshell
    - com.cih.game_cih
    - com.cih.game_cih:p
    - com.cih.game_cih:h
    - com.cih.game_cih:m
    - com.koushikdutta.superuser
    - org.projectvoodoo.otarootkeeper
    - com.noshufou.android.su
    - com.thirdparty.superuser
    - eu.chainfire.supersu
    - com.koushikdutta.superuser
    - kingouser.com

    Bent vieno paketo egzistavimas vienareikšmiškai liudija apie perimtą įrenginio kontrolę.

- Vykdoma paieška populiariausių perimtos AVK fakto paslėpimo funkcionalumą teikiančių paketų („HideMyRoot“, „Temp Root Remover“, „App Quarantine“ ir kt.):

    - com.amphoras.hidemyroot
    - com.zachspong.temprootremovejb
    - com.ramdroid.appquarantine

    Bent vieno paketo egzistavimas vienareikiškai liudija apie perimtą įrenginio kontrolę.

2.3.5. „Android“ programų paketų direktorijų testai

Kai kurie su AVK perėmimu susiję programiniai paketai gali taikyti aktyvias priemones, skirtas paslėpti AVK požymius nuo jų ieškančių programų. Alternatyvus būdas tokiems paketams identifikuoti yra paieška direktorijose, kurias operacinė sistema išskiria mobiliosioms programoms failams laikyti.

- Vykdoma paieška „*/data/data/*“ direktorijoje, kurioje laikomi darbiniai mobiliųjų programų failai. Pačios „*/data/data/*“ direktorijos turinys neturi būti rodomas, tačiau žinant konkrečių gilesnio lygio direktorijų pavadinimus galima aklai prie jų prieiti.

Ieškoma direktorijų:
- /data/data/com.apmhoras.hidemyroot
- /data/data/eu.chainfire.supersu
- /data/data/stericson.busybox
- /data/data/stericson.busybox.donate
- /data/data/com.jrummy.busybox.installer.pro
- /data/data/com.jrummy.busybox.installer
- /data/data/com.noshufou.android.su
- /data/data/org.projectvoodoo.otarootkeeper
- /data/data/com.devadvance.rootcloak2
- /data/data/com.devadvance.rootcloakplus

- Taip pat nustatyta, kad kai kurie kiti paketai savo failus laiko */dev* kataloge.

Ieškoma direktorijų:
- /dev/com.koushikdutta.superuser
- /dev/com.thirdparty.superuser.daemon/server

Sėkmingai identifikavus nors vieną direktoriją iš pateikto sąrašo, daromas loginis sprendimas, jog sistema nulaužta.

2.3.6. „Android“ programų aktyvumo testai

Viename bazinių „Android“ sisteminių paketų „*com.android.settings*“ laikoma informacija apie „Android“ aktyviąsias veiklas (angl. *Activities*). Vykdant paiešką šių veiklų registre galima rasti perimtos AVK požymių.

- Atviro kodo programinė įranga „CyanogenMod“ yra vienas populiariausių pakaitalų standartinei, gamintojo įdiegtai „Android“ operacinei sistemai. Ši įranga automatiškai įrenginio savininkui suteikia ir administracinio vartotojo teises. Siekiant identifikuoti šią sistemą aktyviųjų veiklų registre daroma paieška „*cyanogenmod.superuser* “ įrašui.

Sėkmingai suradus įrašą vienareikšmiškai daromas sprendimas, kad įrenginys nulaužtas.

2.3.7. Sisteminių failų ir direktorijų prieigos teisių testai

„Android“ sistemoje pagrindinės sisteminių direktorijų bei failų prieigos teisės yra iš anksto žinomas. Kadangi „Android“ yra „Linux“ šeimos operacinė sistema, prieigos teisės pagrįstos tipiniu „skaityti, rašyti, paleisti“ (angl. *read / write / execute*) modeliu. Tačiau AVK perimti skirti paketai dėl įvairių priežasčių pakeičia kai kurių failų sistemos objektų teises, pavyzdžiui, siekdami pakeisti sistemos konfigūraciją arba įrašyti papildomų sisteminių komandų. Vykdant tokių anomalijų paiešką galima identifikuoti, ar sistemoje administracinio vartotojo teisėmis buvo vykdomi prieigos teisių pakeitimai.

- Vykdoma paieška žemiau išvardintų direktorijų prieigos teisių nustatymuose siekiant įsitikinti, ar į jas gali rašyti ne tik savininkas bei savininko grupės (angl. *world-writable*).

Tikrinamos direktorijos:
- /
- /data
- /data/data
- /system
- /system/bin

- /system/sbin
- /system/xbin
- /vendor/bin
- /sys
- /sbin
- /etc
- /proc
- /dev

Nemodifikuotoje sistemoje neturėtų būti rašymo teisių į šias direktorijas. Rašymo testui pasitvirtinus daroma išvada, jog įrenginys nulaužtas.

- Vykdoma paieška žemiau išvardintų direktorijų prieigos teisės nustatymuose siekiant nustatyti, ar jas gali skaityti ne tik savininkas ar savininko grupės (angl. *world-readable*).

Tikrinamos direktorijos:
- /data
- /data/data

Nemodifikuotoje sistemoje neturėtų būti skaitymo teisių šioms direktorijoms. Skaitymo testui pasitvirtinus daroma išvada, jog įrenginys nulaužtas.

- „Android“ sistemos „*/proc*“ direktorijoje laikoma visa informacija apie aktyvius procesus, įskaitant ir nuorodą į jų naudojamas operatyvinės atminties sritis. Šios nuorodos yra laikomos failo pavidalu ir niekas, išskyrus administracinį vartotoją, neturėtų turėti teisės prie jų prieiti. Vykdoma paieška, šioje direktorijoje randami visi procesai, kurių identifikacinis numeris yra mažiau nei 1000. Tuomet jų šakninėse direktorijose ieškomi „*mem*“ failai ir tikrinamas jų savininkas bei prieigos teisės.

Visi „*mem*“ failai turėtų turėti savininką „*root*“ bei prieigos teises 0600. Jei randamas nors vienas failas, neatitinkantis šių reikalavimų, didelė tikimybė, jog sistema nulaužta.

2.3.8. Sistemos procesų testai

Dar viena požymių grupė, pagal kurią galima identifikuoti perimtos AVK aplinką, yra sistemos procesų sąrašas.

- Vykdoma paieška procesų registre siekiant nustatyti, ar nėra AVK perėmimui skirtų paketų procesų pavadinimų.

- com.noshufou.android.su
- com.amphoras.hidemyroot
- eu.chainfire.supersu
- com.jrummy.busybox.installer.pro
- stericson.busybox.donate
- com.jrummy.busybox.installer
- stericson.busybox
- com.cih.game_cih
- com.cih.game_cih:p
- com.cih.game_cih:h
- com.cih.game_cih:m
- com.koushikdutta.superuser
- org.projectvoodoo.otarootkeeper

- com.devadvance.rootcloak2
- com.devadvance.rootcloakplus

Identifikavus bent vieną pavadinimą, vienareikšmiškai daroma išvada, jog sistema nulaužta.

- Vykdoma paieška procesų registre siekiant nustatyti, ar nėra paleisti „Linux“ paslaugų (angl. services) procesai, kurių neturėtų būti standartinėje, gamintojo diegtoje „Android“ sistemoje. Vienas tokių procesų yra „*sshd*“, įrodantis, jog sistemoje paleista nuotolinės prieigos paslauga. Šią paslaugą įmanoma paleisti tik su administracinio vartotojo teisėmis (bent jau „Android“ aplinkoje), todėl šio proceso egzistavimas įrodo, jog sistema nulaužta.

2.3.9. Saugumo sertifikatų testai

„Android“ operacinė sistema standartiškai yra reguliariai atnaujinama naudojantis mobilaus įrenginio interneto ryšiu per vadinamąjį „OTA“ procesą (angl. *over-the-air*) [24]. Šio proceso metu yra naudojami „Google“ viešojo rakto sertifikatai, kurie yra reikalingi norint užtikrinti duomenų persiuntimo autentiškumą ir vientisumą. Bet kokie nuokrypiai nuo proceso, pavyzdžiui, sertifikatų nebuvimas ar sertifikatų failų pervadinimas, gali rodyti perimtos AVK būsenos egzistavimą.

- Vykdoma sertifikatų paketų failų paieška šiose direktorijose:

  - /etc/security/otacerts.zip
  - /system/etc/security/otacerts.zip

  Jei nei vienoje direktorijoje nerandamas sertifikatų failas, vadinasi „Android“ operacinė sistema yra paremta nestandartiniu sistemos atvaizdžiu (angl. *custom ROM*), kurio atnaujinimo procesas skiriasi nuo standartinio atvaizdžio. Tokiose sistemose beveik be išimties vartotojai automatiškai gauna administracinio vartotojo teises, todėl pasitvirtinus šiam požymiui daroma išvada, jog sistema nulaužta.

2.3.10. Tinklo nustatymų testai

Dar viena grupė požymių, liudijančių apie nestandartinę „Android“ konfigūraciją, yra susijusi su sistemos tinklo nustatymais. Standartiškai „Android“ sistemoje neturi būti atidarytų tinklo prievadų, ypač administraciniame intervale (tinklo prievado numeris mažesnis nei 1024). Tokių atidarytų tinklo prievadų egzistavimas gali padėti identifikuoti perimtos AVK aplinką.

- Vykdoma paieška atvirų tinklo prievadų, susijusių su populiariausiomis tinklo paslaugomis, kurios gali būti paleidžiamos ant nulaužtų „Android“ įrenginių:

  - Prievadas TCP#20,21 (FTP serverio paslauga)
  - Prievadas TCP#22 (SSH serverio paslauga)
  - Prievadas TCP#53 (DNS serverio paslauga)
  - Prievadas TCP#80,443 (HTTP serverio paslauga)

Radus bent vieną atvirą prievadą daroma išvada, jog įrenginys nulaužtas. Svarbu pastebėti, jog prievado neradus dar nereiškia, jog paslauga nėra paleista ant kito, nestandartinio prievado. Tokiu atveju ši testų grupė perimtos AVK aplinkos neaptiks, tačiau tikėtina, jog nestandartinį servisą (pavyzdžiui, SSH) aptiks kitos požymių paieškos grupės, aprašytos kituose poskyriuose.

2.3.11. Emuliatoriaus identifikavimas

Emuliatorius – įrenginys ar programa, kuri priima tuos pačius duomenis bei komandas, kaip ir emuliuojamoji (pamėgdžiojama) sistema, bei duoda analogiškus emuliajamai sistemai rezultatus [25]. Dėl įvairių saugumo sumetimų gali būti nepageidautina, kad mobilioji programa būtų paleidžiama ne fiziniame įrenginyje (t.y. telefone ar planšetiniame kompiuteryje), o virtualioje aplinkoje. Emuliatoriaus identifikacijai skirti šie testai:

- Jei „*/proc/version*“ faile atrandamas įrašas “Ubuntu”, vadinasi įrenginys yra emuliuojamas „BlueStacks“ programos pagalba.
- Jei akcelerometro parodymai visų trijų ašių atžvilgiu yra 0.0, vadinasi įrenginys yra emuliuojamas. Būtina atkreipti dėmesį, jog ne visuose „Android“ įrenginiuose apskritai yra instaliuojamas akcelerometras, todėl šį testą reikėtų naudoti atsargiai.
- Artumo (angl. *proximity*) sensorius, veikiantis normaliame įrenginyje, visais atvejais rodys bent kažkokį informacinį triukšmą. Jei sensoriaus parodymai yra 0, vadinasi įrenginys emuliuojamas.

2.4. Metodo apibendrinimas

Skyriuje pateiktas perimtos „Android“ administracinio vartotojo aplinkos aptikimo metodas yra skirtas apsaugoti mobiliąsias „Android“ programas, kurioms dėl vykdomos veiklos specifikos ar padidinto duomenų saugumo nerekomenduojama veikti tokioje padidintos rizikos operacinėje aplinkoje. Metodu pagrįsta praktinė realizacija informuotų ją integravusią mobiliąją programą apie „Android“ operacinės sistemos nulaužimo būklę ir leistų mobiliajai programai padaryti loginį sprendimą nutraukti ar tęsti veiklą.

Metodas paremtas požymių, liudijančių apie perimtą administracinio vartotojo kontrolę, paieška pagal įvairias kriterijų grupes. Daugialypio testavimo principas užtikrina didelę aptikimo sėkmės tikimybę net ir tais atvejais, kai nulaužtoje sistemoje taikomos aktyvios priemonės paslėpti AVK perėmimo požymius.

Metodas gali būti pritaikomas visoms „Android“ operacinės sistemos versijoms bei visiems mobiliųjų įrenginių modeliams nepriklausomai nuo gamintojo.

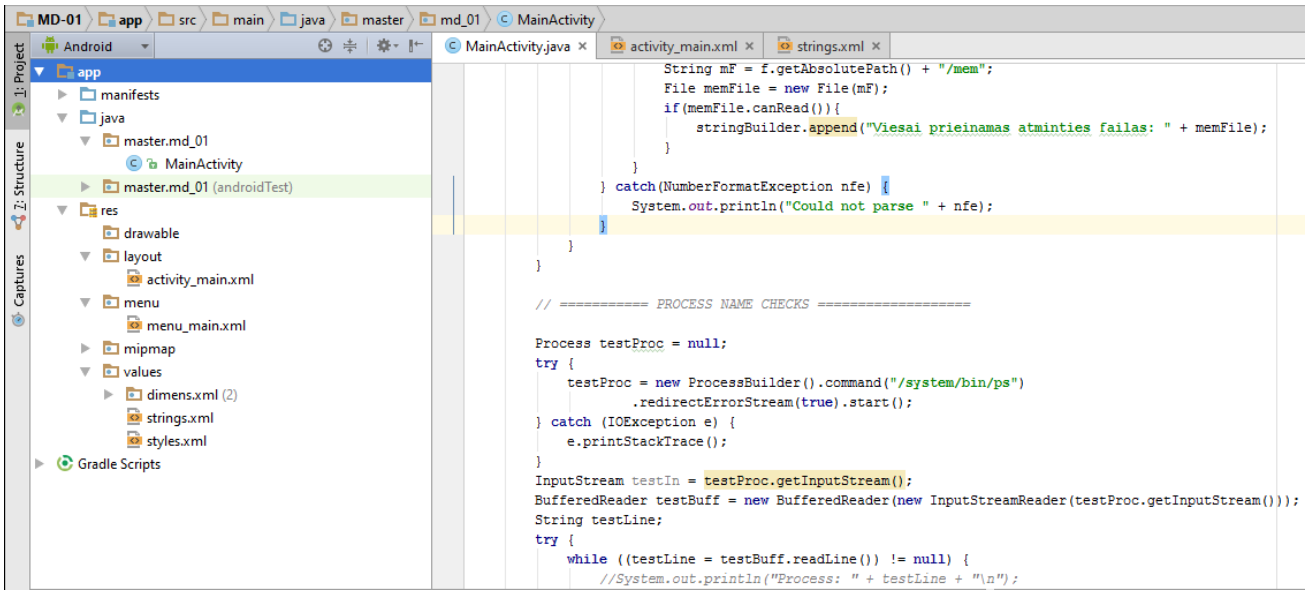
Svarbu pažymėti, kad metodo tikslas nėra ir negali būti 100 procentų aptikimo tikimybės garantija, kadangi nauji AVK perėmimo būdai ir su jais susiję simptomai atsiranda kiekvieną mėnesį. Metodo tikslas yra tiesiog smarkiai sumažinti riziką, susijusią su apsaugotos mobiliosios programos veikimu nesaugioje „Android“ aplinkoje. Tai ypač aktualu, pavyzdžiui, bankines paslaugas teikiančioms įmonėms, kur kiekvienas finansinių operacijų sukčiavimo (angl. *fraud*) procentas nuo bendro operacijų skaičiaus organizacijai gali reikšti didžiulius nuostolius [25].

3. PERIMTOS ADMINISTRACINIO VARTOTOJO KONTROLĖS APTIKIMO METODO REALIZACIJA IR TYRIMAI

Šiame skyriuje apžvelgiama metodo pagrindu paremto prototipo (t.y. realios mobiliosios programos) kūrimo eiga. Tuomet pereinama prie prototipo veikimo principų aprašymo. Galiausiai atliekami bandymai, skirti įvertinti prototipo (ir tuo pačiu – metodo) efektyvumą lyginant su egzistuojančiais kitų autorių metodais bei komerciniais produktais. Skyriaus pabaigoje pateikiamos išvados ir testų rezultatų lentelės.

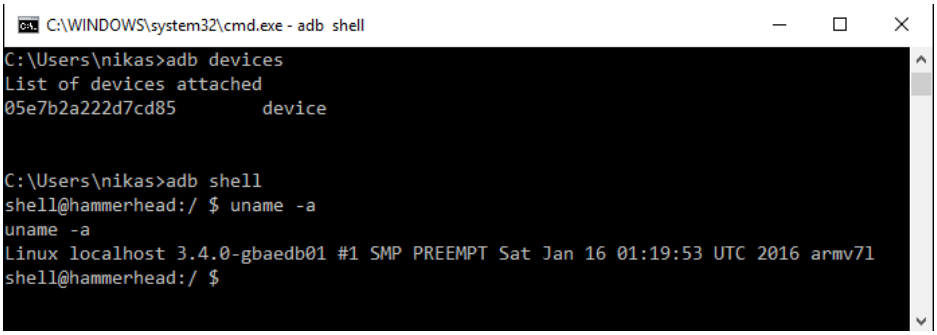
3.1. Perimtos AVK aptikimo prototipo kūrimo procesas

Metodu paremtas prototipas yra „Android“ mobilioji programa, sukurta su „Android Studio“ programavimo aplinka [26]. Šis programinis paketas paremtas kitu populiariu „IntelliJ IDEA“ programavimo paketu, kurį kompanija „Google“ adaptavo specifiškai „Android“ reikmėms. „Android Studio“ programa yra visavertė platforma, leidžianti programinius paketus adaptuoti pagal įrenginio operacinės sistemos versijas, naudotis emuliatoriaus funkcionalumu, betarpiškai perkelti ir paleisti rašomą programą įrenginyje ir kt. Prototipas pateikiamas ir į mobilų įrenginį diegiamas standartiniu „APK“ (angl. *Android package*) formatu. Android programos dažniausiai kuriamos Java programavimo kalba, tačiau dauguma standartinių Java bibliotekų bei klasių yra pakeistos Android skirtomis klasėmis, kadangi mobiliosios programos proceso ir įvesčių valdymui reikalingi specifiniai mechanizmai.



3.1 pav. Android Studio programavimo aplinka

Taip pat prototipo realizacijos ir eksperimentavimo metu intensyviai naudojamas ADB (angl. *Android debug bridge*) sistemos administravimo įrankis, skirtas prototipo programos paketo perkėlimui ir instaliavimui Android aplinkoje, operacinės sistemos parametrų tikrinimui, procesų bei failų peržiūrai. ADB programa leidžia valdyti įrenginį (t.y. mobilų telefoną) per USB jungtį, kopijuoti failus, instaliuoti ar trinti programas, naudotis Android komandine eilute.



3.2 pav. ADB įrankio naudojimo pavyzdys

Prototipo programa kuriama ir testuojama ant dviejų realių, Android operacine sistema paremtų mobiliųjų išmaniųjų telefonų: „Google Nexus 5“ bei „LG G4“. Įrenginiai buvo specialiai parinkti taip, kad skirtųsi jų gamintojai, modeliai ir technologinės kartos. Taip siekiama užtikrinti platesnę požymių ir jiems aptikti skirtų testų apimtį.

Prototipo kūrimo metu pasiremiama 2-ame skyriuje formalizuota veiklos diagrama (2.1 pav.), iliustruojančia lygiagrečius skirtingų kategorijų testus, skirtus identifikuoti administracinio vartotojo kontrolės aplinkos simptomus. Reguliariam programos testavimui naudojamos dvi aplinkos – eksperimentinis telefonas „Google Nexus 5“, kuriame buvo įvairiais būdais perimama AVK, bei kontrolinis, švarus (t.y. nenulaužtas) telefonas „LG G4“ su gamykliniais nustatymais. Tokiu būdu įgyvendinami du esminiai testavimo kriterijai:

- abiem atvejais programa turi rodyti skirtingus rezultatus, kadangi vienu atveju operacinės sistemos aplinka yra nulaužta, kitu – ne;
- lygiagretus testavimas skirtingose aplinkose leidžia efektyviau identifikuoti programavimo klaidas ar sunkiai prognozuojamus kuriamos programos veiksmus.

3.2. Perimtos AVK aptikimo prototipo eksperimentinė fazė

3.2.1. Pasiruošimas eksperimentavimui ir bandymų planas

Siekiant nustatyti sukurto prototipo efektyvumą eksperimentai vykdomi įvairiose aplinkose, atspindinčiose visus įmanomus Android operacinės sistemos konfigūravimo (ir tuo pačiu – saugumo) būklės variantus. Kiekvienam bandymui nustatomi tikslai, kaip turi reaguoti ir kokius požymius turi aptikti prototipas. Taip pat toje pačioje aplinkoje vykdomi bandymai su kontrolinėmis mobiliosiomis programomis, kurios specialiai atrinktos taip, kad turėtų integruotus, vidinius perimtos AVK aptikimo testus. Kontrolinių programų parinkimo kriterijai detaliau aprašomi kitame skyriuje.

Žemiau pateikiamoje lentelėje išvardinamos visos testavimo fazės bei pateikiami reikalavimai bandymo rezultatams. Kiekviena fazė detaliau aprašoma 3.3.3 skyriuje “Eksperimentavimo eiga”.

3.1 lentelė. Prototipo bandymų planas ir laukiami rezultatai

Eilės nr.	**Bandymo aprašymas**	**Iškelti kriterijai bandymo rezultatams vertinti**
1.	Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje	Aplinka identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų
2.	Tikrinamas švarus, gamyklinis Android atvaizdis įrenginyje, kuriame įdiegtas nestandartinis sistemos atkūrimo skirsnis (angl. *recovery partition*)	Aplinka identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų

3.	Tikrinamas nestandartinis, trečių šalių modifikuotas Android atvaizdis be perimtos AVK	Aplinka identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų
4.	Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje įdiegta AVK perėmimo įranga, bet pats AVK perėmimas dar neįvykdytas	Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų
5.	Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje jau perimta AVK	Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų
6.	Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje jau perimta AVK, bet dirbtinai pašalinta / pervadinta „su“ sisteminė komanda	Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų
7.	Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje jau perimta AVK, bet sistemoje įdiegti aktyvūs AVK paslėpti skirti paketai	Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų
8.	Tikrinamas švarus, gamyklinis Android atvaizdis virtualiame įrenginyje (emuliatoriuje)	Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų

Kiekvienoje eksperimento fazėje atliekami bandymai tiek su prototipu, tiek su pasirinktomis realiomis komercinėmis mobiliosiomis programomis, kuriose yra integruotas perimtos AVK aptikimo funkcionalumas. Šias programas vadinamos *kontrolinėmis*, kadangi jos bus atskaitos tašku vertinant šiame darbe aprašyto metodo pagrindu sukurto prototipo efektyvumą.

Kontrolinės programos specialiai pasirinktos iš įvairaus spektro paslaugų sričių: finansinių paslaugų, intelektualinės nuosavybės teisėmis apsaugotų įrašų, elektroninių mokėjimų, verslo duomenų valdymo bei specializuotų techninių mobiliųjų programų. Žemiau pateikiamoje lentelėje pateikiami esminiai pasirinktų programų aprašai.

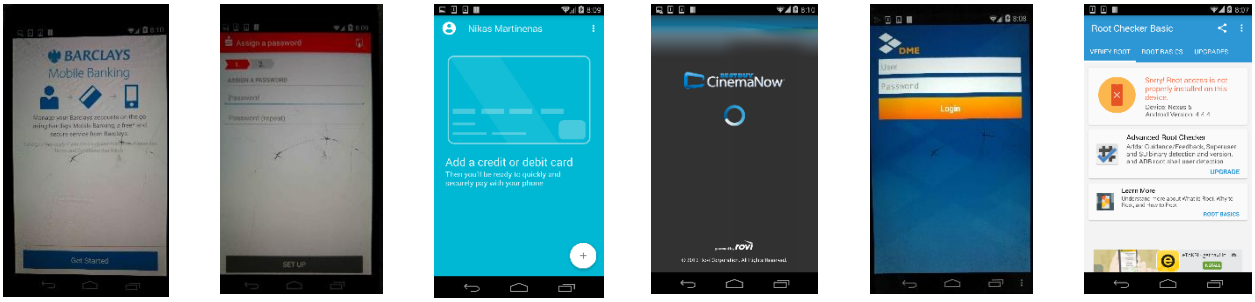
3.2 lentelė. Prototipo efektyvumui palyginti pasirinktos kontrolinės mobiliosios programos

Barclays Mobile Banking	Vieno didžiausių Didžiosios Britanijos bankų Barclays mobilios interneto bankininkystės programa [27]
Star Finanz Sparkasse	Vokietijos banko Star Finanz mobilios interneto bankininkystės programa [28]
Android Pay	Kompanijos „Google“ mobilaus apmokėjimo programa [29]
BestBuy CinemaNow	Vienos didžiausių JAV elektronikos prekių tinklo „BestBuy“ kino filmų pasiūlos internetu mobilioji programa [30]
DME Mail	Kompanijos „Excitor” verslo aplinkai skirta, saugią el. pašto, kalendoriaus, kontaktų aplinką užtikrinanti mobilioji programa [31]
Root Checker	Populiariausia mobilioji programa „Google Play“ virtualioje prekyvietėje, skirta perimtos AVK būklei aptikti [32]

Reikia pažymėti, jog analizės skyriuje minėti „Samsung Knox“ bei „VMWare AirWatch“ produktai nebuvo pasirinkti kaip palyginamieji, nes šių programų bandymams būtų reikalinga pilna produkto infrastruktūra su serveriais, profiliais ir kliento konfigūracija, o tai užtikrinančios licencijos yra mokamos ir brangios. Tiesa, pačią „AirWatch“ vartotojui skirtą mobiliąją programą galima įdiegti per „Google Play“ virtualią prekyvietę, tačiau paleidimo metu programa perimtos AVK netikrina (ir

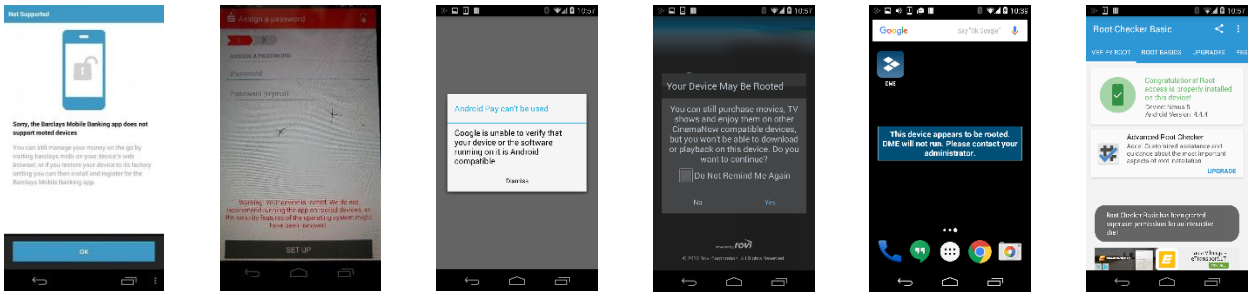
dėl to netinka kontroliniams testams), kadangi jai reikalingas ryšys su centriniu serveriu, kuriame galėtų būti sukonfigūruotos instrukcijos perimtos AVK tikrinimui.

Visos pasirinktos kontrolinės programos prieš pradedant bandymus buvo paleistos perimtos AVK aplinkoje siekiant įsitikinti, jog kiekviena jų turi bent jau bazinį nulaužtos sistemos aptikimo funkcionalumą. Žemiau esančiuose paveikslėliuose pateikiami įrodymai, jog kontrolinės programos tvarkingai veikia švarioje, originalioje „Android“ sistemoje, bet iškart atsisako veikti, jei aptinkama perimta AVK. Programos „Root Checker“ atveju, klaidos pranešimas rodomas, kai nėra aptinkama AVK; ir atitinkamai sėkmės pranešimas rodomas tada, kai AVK aptinkama.



Barclays Mobile Banking | **Star Finanz Sparkasse** | **Android Pay** | **BestBuy CinemaNow** | **DME Mail** | **Root Checker**

3.3 pav. Kontrolinių mobiliųjų programų ekrano kopijos esant švariai OS aplinkai



Barclays Mobile Banking | **Star Finanz Sparkasse** | **Android Pay** | **BestBuy CinemaNow** | **DME Mail** | **Root Checker**

3.4 pav. Kontrolinių mobiliųjų programų ekrano kopijos aptikus perimtą AVK

Kai kurios mobiliosios programos dėl saugumo sumetimų draudžia daryti ekrano kopiją iš pačio telefono, todėl tais atvejais ekranas buvo fotografuotas iš kito telefono, kas sumažino jų vaizdo kokybę. Tolesniuose skyriuose šios ekrano kopijos nebebus vizualiai kartojamos, tiesiog bus fiksuojama ir konstatuojama, kokie buvo konkrečios kontrolinės programos reakcija ir rezultatas.

Eksperimentai vykdomi „Google Nexus 5“ mobiliajame išmaniajame telefone. Kadangi šio magistrinio projekto darymo metu (2016 metais) populiariausia „Android“ operacinės sistemos versija buvo 4.4.4 (kodinis pavadinimas “KitKat”) [33], šios versijos atvaizdis buvo pasirinktas eksperimentavimui. Versijos pasirinkimas didesnės įtakos bandymų rezultatams bet kokiu atveju

neturi, išskyrus tuos atvejus, kai diegiamos mobiliosios programos, nepalaikančios pačių naujausių operacinės sistemos versijų.

3.2.2. Eksperimentavimo eiga

Šiame skyriuje detaliau aprašomi visi bandymų etapai, prieš tai glaustai apžvelgti 3.1 lentelėje. Kiekvieno etapo metu atliekami sistemos paruošiamieji darbai, vykdomos sisteminės komandos siekiant sukurti išsikeltus kriterijus atitinkančią eksperimentavimo aplinką. Taip pat pateikiamos prototipo programos rezultatų ekrano kopijos bei kontrolinių programų rezultatų lentelės. Rezultatai susumuojami ir bendros išvados pateikiamos tolimesniame 3.2.3 skyriuje.

Bandymas Nr.1

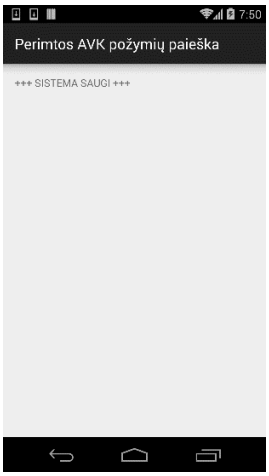
Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje. Aplinka turi būti identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų. Šio bandymo tikslas įsitikinti, jog tiek prototipas, tiek kontrolinės programos korektiškai identifikuoja švarią, nenulaužtą sistemą.

Bandymui pasiruošiama prijungus mobilų įrenginį prie darbinio kompiuterio per USB jungtį. Parsisiunčiamas originalus „Android“ 4.4.4 versijos sisteminis atvaizdis iš oficialios „Google“ svetainės [34]. Naudojantis sisteminėmis komandomis „*adb*“ ir „*fastboot*“ įrenginys perkraunamas į paleidyklės (angl. *bootloader*) režimą ir į įreginį įdiegiami švarūs, originalūs sisteminiai atvaizdžiai:

adb reboot bootloader
fastboot -w update image-hammerhead-ktu84p.zip

Tuomet įrenginys perkraunamas į vartotojo režimą. Per ADB jungtį įdiegiama prototipo programa bei kontrolinės komercinės programos, aprašytos eksperimento plano skyriuje. Visos programos paleidžiamos ir fiksuojami rezultatai.

3.3 lentelė. Eksperimento bandymo nr.1 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Taip	
Android Pay	Taip	
BestBuy CinemaNow	Taip	
DME Mail	Taip	
Root Checker	Taip	

Kaip ir tikėtasi, švarioje operacinėje sistemoje visos mobiliosios programos veikia be jokių sutrikimų ar klaidos pranešimų. Taip pat įsitikinama, kad magistrinio darbo prototipas taip pat korektiškai fiksuoja sistemą be perimtos AVK požymių.

Bandymas Nr.2

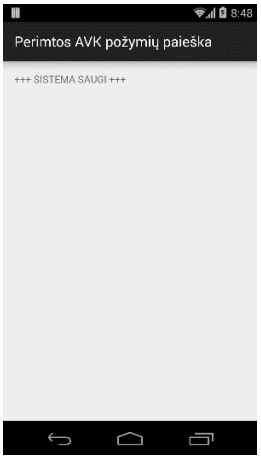
Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kuriame įdiegtas nestandartinis sistemos atkūrimo skirsnis (angl. *recovery partition*). Aplinka turi būti identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų. Šio bandymo tikslas įsitikinti, jog dėl nestandartinio atkūrimo skirsnio sistema nebus laikoma nulaužta, kadangi tokio skirsnio egzistavimas savaime neįrodo sistemos saugumo pažeidimų, nors dažnai ir yra pirmas žingsnis AVK perėmimo procese.

Parsisiunčiamas populiarus sistemos atkūrimo paketas TWRP [35], trečiųjų šalių suprogramuotas kaip alternatyva standartiniams gamykliniams paketams. Kaip ir anksčiau, naudojantis specializuotais sisteminiais įrankiais „*adb*“ bei „*fastboot*“ įdiegiamas sistemos atkūrimo skirsnio paketas:

adb reboot bootloader
fastboot flash recovery twrp-2.8.x.x-xxx.img
fastboot reboot

Operacinė sistema ir toliau užkraunama originali, kokia buvo įdiegta pirmo bandymo metu. Prototipas ir visos kontrolinės programos paleidžiamos, fiksuojami rezultatai.

3.4 lentelė. Eksperimento bandymo nr.2 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Taip	
Android Pay	Taip	
BestBuy CinemaNow	Taip	
DME Mail	Taip	
Root Checker	Taip	

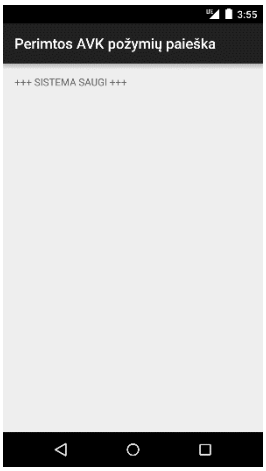
Nei viena iš programų neaptinka pakeisto sistemos atkūrimo skirsnio, arba jei ir aptinka, nelaiko to saugumo pažeidimo požymiu.

Bandymas Nr.3

Tikrinamas nestandartinis, trečių šalių modifikuotas „Android“ atvaizdis (angl. *custom ROM*) be perimtos AVK. Aplinka identifikuojama kaip saugi, neturi būti I tipo (klaidingai teigiamų) klaidų. Šio bandymo tikslas įrodyti, jog nestandartinis Android atvaizdis dar nėra pakankama priežastis operacinę sistemą laikyti nesaugia, jei operacinėje sistemoje nebuvo vykdyti aktyvūs veiksmai perimti AVK.

Bandymui pasirenkamas vienas populiaresnių modifikuotų „Android“ atvaizdžių „ParanoidAndroid“ [36]. Jo privalumas prieš kitus populiarius atvaizdžius, bent jau šio bandymo kontekste, yra tai, kad jame nėra automatiškai įdiegiamas perimtos AVK funkcionalumas. „ParanoidAndroid“ įdiegiamas į mobilų įrenginį pasinaudojant ankstesniame bandyme įdiegtu „TWRP“ sistemos atstatymo skirsniu. Į naują operacinę sistemą įdiegiamos visos kontrolinės programos bei prototipas. Programos paleidžiamos ir fiksuojami rezultatai.

3.5 lentelė. Eksperimento bandymo nr.3 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Taip	
Android Pay	Ne	
BestBuy CinemaNow	Taip	
DME Mail	Taip	
Root Checker	Taip	

Bandymo rezultatai parodo, jog dauguma mobiliųjų programų su įdiegtu AVK požymių patikros funkcionalumu nestandartinio „Android“ atvaizdžio nelaiko saugumo problema. Vienintelė išimtis yra „Android Pay“, kurios saugumo reikalavimai yra tokie griežti, jog programa atsisako veikti aptikusi neoriginalų (t.y. ne „Google“ paruoštą ir patvirtintą) operacinės sistemos atvaizdį.

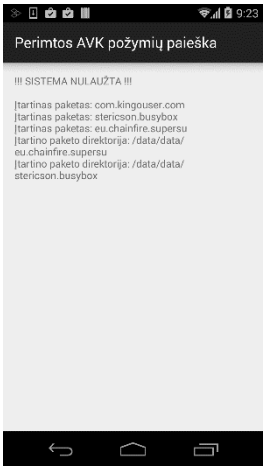
Bandymas Nr.4

Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje įdiegta AVK perėmimo įranga, bet pats AVK perėmimas dar neįvykdytas. Aplinka turi būti identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų. Šio bandymo tikslas įsitikinti, jog yra aptinkami pirmieji požymiai veiklos, nukreiptos į AVK perėmimą. Nors teoriškai pati sistema dar saugi, tačiau AVK perėmimui skirtų paketų buvimas sistemoje rodo nemažą tikimybę, jog sistemos saugumo artimu metu bus pažeistas.

Bandymo metu į įrenginį pasinaudojant „Google Play“ virtualia prekyviete yra įdiegiamos mobiliosios programos „SuperSU“, „BusyBox“, „KingoSuperUser“, kurių funkcionalumas betarpiškai priklauso nuo perimtos AVK teisių prieinamumo. Mobilioji programa „SuperSU“ skirta valdyti perimtos AVK prieigos teises, kurių gali prašyti kitos „Android“ programos. „BusyBox“ paketas įdiegia papildomas sistemos administravimo komandas, kurioms reikalinga administracinio vartotojo

prieiga. „KingoSuperUser“ yra populiaraus AVK perėmimo paketo „Kingo Root“ dalis. Įdiegus minėtus paketus, prototipas ir visos kontrolinės programos paleidžiamos, fiksuojami rezultatai.

3.6 lentelė. Eksperimento bandymo nr.4 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Ne	
Android Pay	Ne	
BestBuy CinemaNow	Ne	
DME Mail	Ne	
Root Checker	Ne	

Tik prototipas ir „Barclays“ mobilioji programa identifikuoja įtartinų programinių paketų egzistavimą sistemoje. Kitos programos arba neturi įdiegto funkcionalumo aptikti šiuos požymius, arba tokius požymius laiko nepakankamais, kad sistema būtų laikoma nesaugia.

Bandymas Nr.5

Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje jau perimta AVK. Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų. Šio bandymo tikslas įsitikinti, kad tiek prototipas, tiek kontrolinės mobiliosios programos sėkmingai identifikuoja perimtą AVK, kai nėra taikomi jokie papildomi būdai paslėpti AVK požymių.

Populiariausias būdas perimti AVK jau turint prieigą prie išplėsto funkcionalumo nestandartinio atstatymo skirsnio „TWRP“ (žr. bandymą nr.2) yra paketo, turinčio „su“ (angl. *superuser*) sisteminę komandą, diegimas. Komanda „su“ leidžia neprivilegijuotam vartotojui perimti administracinio vartotojo teises. Parsisiunčiamas programinis paketas „SuperSU“ [37], kurio vienas iš komponentų yra minėta sisteminė „Android“ komanda. Paketas kopijuojamas į šakninę mobilaus įrenginio direktoriją, ir tuomet vykdomos šios komandos:

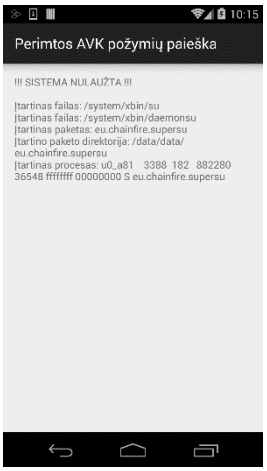
adb reboot recovery
TWRP: Install -> UPDATE-SuperSUv.198

Perkrovus įrenginį, visų pirma įsitikinama, kad „su“ sisteminė komanda sėkmingai įdiegta. Tam panaudojama ADB komandinė eilutė:

adb shell
su –

Įsitikinus, kad „su“ komanda iš komandinės eilutės veikia sėkmingai, paleidžiamas prototipas ir visos kontrolinės programos, fiksuojami rezultatai.

3.7 lentelė. Eksperimento bandymo nr.5 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Taip	
Android Pay	Taip	
BestBuy CinemaNow	Taip	
DME Mail	Taip	
Root Checker	Taip	

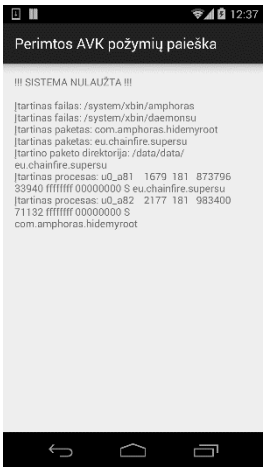
Kaip ir tikėtasi, visos programos sėkmingai aptinka perimtos AVK būseną. Dauguma programų remiasi tik sisteminės komandos „su“ paieška, kuri šio bandymo metu nėra slepiama. Svarbu atkreipti dėmesį, jog šiame bandyme prototipas sėkmingai aptinka daugiau nei vieną požymį: sisteminę komandą „su“, įtartiną paketą pagal registrą, įtartino paketo direktoriją bei procesą.

Bandymas Nr.6

Tikrinamas originalus, standartinis „Android“ atvaizdis, kuriame jau perimta AVK, bet dirbtinai pašalinta ar pervadinta „su“ sisteminė komanda. Aplinka turi būti identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų. Šio bandymo tikslas įsitikinti, kurios programos, siekdamos aptikti perimtą AVK, remiasi tik primityviu testu, skirtu aptikti „su“ sisteminę komandą.

Šio eksperimento paruošiamieji darbai beveik tapatūs bandymui nr.5, tik papildomai yra ištrinama arba pervadinama „su“ komanda. Šį veiksmą galima atlikti arba rankiniu būdu prisijungus prie „Android“ sistemos per ADB komandinę eilutę, arba pasinaudojant tam skirtomis „Android“ mobiliosiomis programomis (pvz. „Hide my Root“ [38]). Įsitikinus, jog minėtais metodais sėkmingai pašalinta ar pervadinta „su“ komanda, paleidžiamas prototipas bei kontrolinės programos, fiksuojami rezultatai.

3.8 lentelė. Eksperimento bandymo nr.6 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Taip	
Android Pay	Taip	
BestBuy CinemaNow	Ne	
DME Mail	Taip	
Root Checker	Ne	

Bandymo rezultatai parodo, jog mobiliosios programos, kurios remiasi tik „su“ komandos egzistavimo patikrinimu, gali būti lengvai apgaunamos komandą paslėpus kitoje direktorijoje ar pervadinus. Prototipas sėkmingai identifikuoja perimtos AVK būklę pagal kitus kriterijus: AVK paslėpimui skirtų paketų įrašus programų registre, įtartinas direktorijas, procesus.

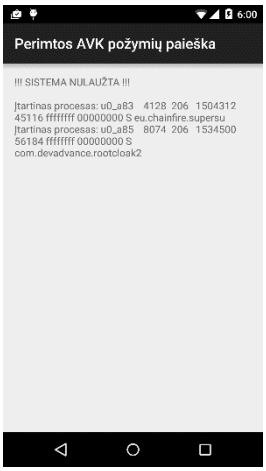
Bandymas Nr.7

Tikrinamas švarus, gamyklinis Android atvaizdis originalioje įrenginio konfigūracijoje, kurioje jau perimta AVK, bet sistemoje įdiegti aktyvūs AVK paslėpti skirti paketai. Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų. Šio bandymo tikslas įvertinti AVK požymių aptikimo efektyvumą, kai taikomos sudėtingesnės perimtos AVK požymių slėpimo priemonės.

Bandymo nr.6 metu buvo elementariai pervadinama „su“ komanda ir žiūrima, kurios mobiliosios programos šio veiksmo neaptiks. Tuo tarpu šio bandymo metu naudojami populiari perimtos AVK požymių slėpimo programų šeima „RootCloak“ [39] [40]. Šios programos specialiai parašytos tam, kad paslėptų AVK požymius nuo kitų tame pačiame įrenginyje įdiegtų mobiliųjų programų. Jos slepia informaciją apie įtartinus paketus, pašalina įvairius įkalčius failų sistemos lygmenyje, koreguoja nestandartinės operacinės sistemos konfigūracinius nustatymus taip, kad jie atitiktų originalius.

Bandymo metu visų pirma įdiegiamas papildomas „Android“ programinis karkasas (angl. *framework*) „Xposed“ [41], kuris reikalingas kitiems diegiamiems komponentams. Šiam bandymui naudojama „Android“ 5.1.1 versija, kadangi ant standartiškai pasirinktos ir kituose bandymuose naudojamos 4.4.4 tvarkingai neveikia „Xposed“ paketas. Tada kaip „Xposed“ modulis diegiama „RootCloak Plus 1.52“ programos versija. Paleidus šią programą, visos kontrolinės programos, įskaitant prototipą, įtraukiamos į „RootCloak“ konfigūraciją, kad perimtos AVK požymiai būtų slepiami. Prototipas ir kontrolinės programos paleidžiamos, fiksuojami rezultatai.

3.9 lentelė. Eksperimento bandymo nr.7 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip	
Star Finanz Sparkasse	Ne	
Android Pay	Taip	
BestBuy CinemaNow	Ne	
DME Mail	Taip	
Root Checker	Ne	

Palyginus šio bandymo rezultatus su bandymo nr.6 rezultatais ir ypač su prototipo rezultatų ekranų kopijomis, matyti, kad „RootCloak“ slepiančioji programa taiko kur kas sudėtingesnius metodus nei tik elementarų „su“ sisteminės komandos pervadinimą. Nuo prototipo slepiami paketų registrai, AVK reikalingi failai, tačiau prototipas vis tiek aptinka įtartinų paketų sisteminius procesus. Taip pat matyti, kad tokios aukštos kokybės programos, kaip „Barcays Mobile Banking“ ar „Android Pay“, irgi savyje turi integruotus sudėtingus AVK požymių aptikimo metodus. Kita dalis kontrolinių programų neberanda perimtos AVK požymių ir klaidingai operacinės sistemos aplinką fiksuoja kaip saugią.

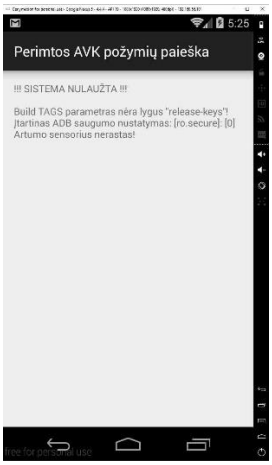
Bandymas Nr.8

Tikrinamas švarus, gamyklinis Android atvaizdis virtualiame įrenginyje (emuliatoriuje). Aplinka identifikuojama kaip nesaugi, neturi būti II tipo (klaidingai neigiamų) klaidų. Šio bandymo tikslas yra palyginti prototipo bei kontrolinių programų rezultatus virtualioje aplinkoje.

Virtuali aplinka jau savaime gali būti laikoma nesaugia, net jei nėra akivaizdžių AVK perėmimo požymių. Taip yra todėl, kad fizinio kompiuterio, kuriame paleista virtuali aplinka, savininkas bet kada gali įdiegti AVK paketus, stebėti ir keisti virtualios mašinos sisteminius nustatymus, mobiliųjų programų procesus. Kitaip tariant, toks asmuo jau turi privilegijuotą (atitinkantį perimtos AVK būseną) priėjimą prie virtualios sistemos, net jei ši ir nėra tiesiogiai nulaužta.

Bandymui pasirenkamas vienas populiariausių mobiliųjų įrenginių virtualizacijos įrankis „Genymotion“ [43]. Įdiegus asmeniniam naudojimui skirtą įrankio versiją, sukuriama virtuali, „Android“ 4.4.4 versijos atvaizdžiu paremta mašina. Kadangi „Genymotion“ savo siūlomus „Android“ atvaizdžius komplektuoja kartu su automatiškai perimta AVK būkle ir AVK funkcionalumą išplečiančiais įrankiais (pavyzdžiui, programa „busybox“), įdiegta virtuali mašina papildomai išvaloma, kad neliktų jokių su AVK susijusių požymių. Bandymo tikslas šiuo atveju yra aptikti išimtinai su virtualizacija susijusius požymius. Sutvarkius ir paruošus virtualią sistemą, įdiegiamos ir paleidžiamos kontrolinės programos bei prototipas, fiksuojami rezultatai.

3.10 lentelė. Eksperimento bandymo nr.8 rezultatai

Mobilioji programa	**Bandymo rezultatas atitinka iškeltus kriterijus**	**Prototipo rezultatų ekrano kopija**
MD prototipas	Taip	
Barclays Mobile Banking	Taip*	
Star Finanz Sparkasse	Taip	
Android Pay	Taip*	
BestBuy CinemaNow	Taip	
DME Mail	Taip*	
Root Checker	Ne	

Iš prototipo rezultatų ekrano kopijos matyti, jog šįkart nesaugi aplinka aptikta pagal visai kitas požymių grupes, nei prieš tai darytuose bandymuose. Pirmąkart aptiktas pats ryškiausias virtualios aplinkos bruožas – neaktyvuoti fiziniai sensoriai. Taip pat aptikti sisteminės konfigūracijos nustatymai, įrodantys administracinio vartotojo prieigos galimybes. Nei viena kontrolinė programa nepasileido sėkmingai, tačiau ne visos programos aktyviai aptiko AVK požymius. Lentelėje žvaigžde pažymėtos programos apskritai nepasileido virtualioje aplinkoje dėl techninio suderinamumo problemų. Tačiau šio bandymo tikslų kontekste laikome, jog pažymėtos kontrolinės programos įveikė testą, kadangi bet kokiu atveju neleido jomis naudotis.

3.2.3. Eksperimentavimo rezultatai ir išvados

Praeitame skyriuje buvo atlikti aštuoni bandymai, skirti imituoti įvairias galimas „Android“ operacinės sistemos būsenas, įdiegtus ir naudojamos komponentus, konfigūracinius nustatymus. Kiekvieno bandymo pabaigoje buvo pateikiama rezultatų apžvalga ir trumpos išvados, ką rezultatai rodo bandymo kontekste. Šiame skyriuje visų bandymų rezultatai susumuojami ir pateikiami vienoje lentelėje. Taip įvertinamas prototipo efektyvumas lyginant su pasirinktomis kontrolinėmis mobiliosiomis programomis.

Žemiau pateiktoje lentelėje pateiktos kiekvienos mobiliosios programos atitiktys išsikeltam bandymo tikslui. Bandymų metu siekta nustatyti, ar testuojama programa aptinka AVK perėmimo požymius įvairiose „Android“ konfigūracijos būklėse, taip pat ar nerodo I tipo klaidų, kai sistema yra laikoma švaria.

3.11 lentelė. Eksperimento visų bandymų rezultatai ir jų palyginimas

	Prototipas	Barclays Mobile Banking	Star Finanz Sparkasse	Android Pay	BestBuy CinemaNow	DME Mail	Root Checker
Bandymas #1 (švari sistema)	+	+	+	+	+	+	+
Bandymas #2 (atstatymo skirsnis)	+	+	+	+	+	+	+
Bandymas #3 (3ios šalies atvaizdis)	+	+	+	–	+	+	+
Bandymas #4 (įdiegta AVK įranga)	+	+	–	–	–	–	–
Bandymas #5 (perimta AVK)	+	+	+	+	+	+	+
Bandymas #6 (primityvus slėpimas)	+	+	+	+	–	+	–
Bandymas #7 (sudėtingas slėpimas)	+	+	–	+	–	+	–
Bandymas #8 (virtuali aplinka)	+	+	+	+	+	+	–

Pagal pasirinktus kriterijus visus bandymus pilnai teigiamai išlaikė tik šio magistrinio projekto metodo prototipas bei „Barclays Mobile Banking“ mobilioji programa. „Android Pay“ produktas taip pat akivaizdžiai yra aukštos kokybės ir sugeba identifikuoti įvairius perimtos AVK požymius, tačiau 3-ių šalių „Android“ sistemos atvaizdžius identifikuoja kaip nesaugius, kas nesutampa su šiame darbe išsikeltu saugios operacinės sistemos apibrėžimu. Dar du komerciniai produktai, bankinio atsiskaitymo programa „Star Finanz Sparkasse“ bei intelektualinę kino filmų nuosavybę sauganti „BestBuy CinemaNow“, nesugebėjo aptikti įtartinų, AVK perėmimui skirtų paketų prieš juos aktyviai panaudojant, taip pat nebuvo atsparios pažengusiai AVK požymių slėpimo technikai naudojant „RootCloak“ šeimos programas. Mobilioji verslo klientų komunikacijų programa „DME Mail“ teigiamai pasirodė daugumoje bandymų, išskyrus AVK paketų aptikimą. Galiausiai, programa „Root Checker“, kaip ir tiketasi, neaptiko AVK požymių 50 proc. bandymų. To ir buvo galima tikėtis, kadangi šios programos funkcionalumas paremtas tik „su“ sisteminės komandos paieška.

Bandymų metu pateiktuose prototipo rezultatų ekrano kopijose matyti, jog daugiasluoksnis, požymių grupėmis paremtas metodas pasiteisina skirtinguose sistemos konfigūravimo scenarijuose. Prototipas buvo specialiai suprogramuotas ekrane parodyti konkretų identifikuotą požymį, todėl kiekvieno bandymo metu yra aišku, kurių testų rezultatai buvo teigiami. Ypatingai pasiteisino sistemos procesų paieška, nuo kurios AVK požymiai nebuvo paslėpti nei viename bandyme.

Emuliatoriaus bandyme pasiteisino prototipo požymių paieškos grupė, skirta aptikti informacinio triukšmo neteikiančius sensorius. Šio konkretaus bandymo metu buvo nerastas artumo (angl. *proximity*) sensorius. Taip pat svarbu atkreipti dėmesį, jog emuliatoriaus bandymo metu kai kurios mobiliosios programos („Barclays“, „Android Pay“, „DME“) apskritai nepasileido ir nerodė jokių papildomų klaidos pranešimų. Todėl sunku įvertinti, ar tai buvo sąlygota sėkmingai aptiktos emuliatoriaus aplinkos testo, ar programa iš esmės nėra suderinama su emuliatoriumi. Bandymo kontekste tai buvo užskaityta kaip teigiamas rezultatas, kadangi šios programos bet kokiu atveju neleido jomis naudotis.

Pažymėtina, jog eksperimento kontrolinės programos buvo pasirinktos kaip populiariausios ir kokybiškiausios virtualioje „Google Play“ prekyvietėje. Pavyzdžiui, „Barclays Mobile Banking“ mobilioji programa, kaip didelio banko produktas, akivaizdžiai suprogramuota profesionalios, IT saugos konsultantų paramą turinčios komandos. Daugumos „Google Play“ prekyvietėje esančių mobiliųjų programų kokybės lygis, ypač informacijos saugos atžvilgiu, bus kur kas žemesnis. Todėl galima teigti, kad atsitiktinai pasirinkus bet kokią kitą grupę AVK požymių ieškančių mobiliųjų

programų ir jas ištestavus pagal tą pačią bandymų metodiką, prototipo efektyvumo atotrūkis būtų kur kas ryškesnis.

4. IŠVADOS

Per pastarąjį dešimtmetį ypač išpopuliarėjo išmanieji mobilieji įrenginiai, kurių didžiąją rinkos dalį užpildo kompanijos „Google“ operacinė sistema „Android“. Kadangi mobilieji įrenginiai iš esmės yra mažų gabaritų kompiuteriai, jiems ne tik aktualios tos pačios informacinių technologijų grėsmės, kaip ir asmeniniams kompiuteriams, bet taip pat ir nauji, su mobiliaja aplinka susiję saugos pažeidimai. Viena iš ryškėjančių saugumo spragų tendencijų yra perimta administracinio operacinės sistemos vartotojo (angl. *root*) prieiga, kuri smarkiai sumažina bendrą mobilaus įrenginio atsparumą saugos pažeidimams. Mobiliųjų programų, kurios dirba su konfidencialia informacija (pavyzdžiui, banko pavedimais), kūrėjai yra suinteresuoti iš anksto identifikuoti tokią nesaugią aplinką. Atlikus akademinės literatūros bei komercinių produktų pasiūlos analizę nebuvo rastas nei vienas pilnavertis, viešai prieinamas ir panaudojamas būdas AVK aplinkai aptikti.

Šiame magistriniame projekte suformuluojamas ir argumentuojamas metodas, leidžiantis aptikti nesaugią, perimtą administracinio vartotojo kontrolę sistemoje. Metodas apibendrina jau egzistuojančius bei siūlo naujus perimtos AVK požymių aptikimo algoritmus. Metodas pagrįstas daugialype įvairių požymių grupių analize siekiant užtikrinti, kad AVK būtų aptikta net tose operacinėse sistemose, kur įdiegta specializuota, požymius slepianti programinė įranga.

Eksperimentiniame šio darbo etape su „Android Studio“ programavimo paketu ir Java programavimo kalba buvo sukurtas pilnai veikiantis ir esminį metodo funkcionalumą realizuojantis prototipas. Prototipo programavimas ir testavimas buvo vykdomas pasitelkus realius „Android“ operacine sistema pagrįstus mobiliuosius telefonus, taip pat buvo naudota virtuali emuliatoriaus aplinka. Šiame etape taip pat buvo atlikta grupė bandymų, skirtų įvertinti prototipo efektyvumą esant įvairioms „Android“ sistemos konfigūracijoms. Buvo vertinamas prototipo sugebėjimas aptikti perimtą AVK arba korektiškai identifikuoti švarią aplinką. Tuo pat metu buvo pasirinkta grupė realių, didelį populiarumą tarp mobiliųjų įrenginių vartotojų turinčių komercinių mobiliųjų programų, kurios turi integruotą perimtos AVK požymių paiešką. Šios kontrolinės grupės rezultatai buvo lyginami su prototipo rezultatais. Galutinės eksperimento išvados patvirtino, kad prototipas įveikė visas bandymų kategorijas, tuo tarpu daugumai kontrolinių programų ne visais scenarijais pavykdavo atitikti išsikeltus bandymų kriterijus.

Viena iš silpnųjų metodo, o tuo pačiu ir jo pagrindu sukurtos prototipo, pusių yra tai, kad dauguma požymių paieškos algoritmų yra paremti konkrečių AVK perėmimo programinių paketų ar specifinių sistemos konfigūracijos parametrų paieška. Laikui bėgant neabejotinai atsiras vis naujų AVK perėmimo programų, tad siekiant, kad metodas išliktų efektyvus, jo požymių paieškos algoritmai turėtų būti reguliariai atnaujinami.

Svarbu pažymėti, kad darbe naudotas prototipas buvo minimalios apimties, tik esminį, bazinį funkcionalumą siūlanti mobilioji programa. Prototipo galimybes galima ir toliau plėsti, pavyzdžiui, paverčiant jį ne savarankiška programa, o atskiru Java klasių rinkiniu, kuris galėtų būtų integruojamas į bet kurią mobiliąją „Android“ programą ir jai teiktų informaciją apie operacinės sistemos saugumo būseną. Taip pat metodo buvo numatytas, bet prototipo pirmajame etape nerealizuotas simptomų svorių įvertinimo algoritmas. Toks metodas leistų kiekvienam simptomui priskirti subjektyvią vertę, o visų testų grupių pabaigoje tokios vertės būtų susumuojamos ir tikrinamos, ar nebuvo viršytas iš anksto numatytas vertės slenkstis. Tokiu būdu pavieniai įtartini simptomai dar nebūtinai galėtų reikšti, jog sistema pilnai nulaužta. Dar vienas funkcionalumo plėtros variantas būtų galimybė metodu paremtoje mobiliojoje programoje įdiegti realaus laiko AVK požymių duomenų bazės atnaujinimą, panašiai kaip tai daroma moderniose antivirusinėse programose. Tačiau tokiu atveju AVK požymių aptikimo produktas nebegalėtų egzistuoti kaip savankiška programa ar Java klasių rinkinys, ir taptų dalimi kompleksinės sistemos, teikiančios naujų AVK požymių paieškos ir jų atnaujinimo internetu paslaugas.

5. LITERATŪRA

[1] „Smartphone user penetration as percentage of total global population,“ [Tinkle]. Available: http://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/. [Kreiptasi 12 01 2016].

[2] „Emerging Markets Drove Worldwide Smartphone Sales to 15.5 Percent Growth in Third Quarter of 2015,“ [Tinkle]. Available: http://www.gartner.com/newsroom/id/3169417. [Kreiptasi 22 12 2015].

[3] OWASP, „OWASP Top 10 Mobile Risks,“ OWASP, [Tinkle]. Available: https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks. [Kreiptasi 14 12 2015].

[4] „An Overview of Android Architecture,“ [Tinkle]. Available: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture. [Kreiptasi 18 02 2016].

[5] J. Boutet, Malicious Android Applications: Risks and Exploitation, SANS, 2010.

[6] K. Labs, „Rooting and Jailbreaking: What Can They Do, and How Do They Affect Security?,“ [Tinkle]. Available: https://blog.kaspersky.com/rooting-and-jailbreaking/1979/. [Kreiptasi 7 01 2016].

[7] „Android Authority,“ [Tinkle]. Available: http://www.androidauthority.com/rooting-for-dummies-a-beginners-guide-to-root-your-android-phone-or-tablet-10915/. [Kreiptasi 03 02 2016].

[8] W. V. Jason Tyler, Android Hacker's Toolkit, John Wiley & Sons, Ltd., 2012.

[9] D. S. Z. Q. Hang Zhang, Android Root and its Providers:A Double-Edged Sword, Riverside: University of California, 2015.

[10] J. R. Ball, Detection And Prevention of Android Malware Attempting To Root The Device, US Air Force Institute of Technology, 2014.

[11] „Enterprise Mobile Threat Report,“ [Tinkle]. Available: https://info.lookout.com/rs/051-ESQ-475/images/Enterprise_MTR.pdf. [Kreiptasi 11 03 2016].

[12] „Implementing Security | Android Open Source Project,“ [Tinkle]. Available: https://source.android.com/security/implement.html. [Kreiptasi 14 02 2016].

[13] W.-B. C. H.-W. L. You Joung Ham, „Mobile root exploit detection based on system events extracted from Android platform,“ Osan, Rep. of Korea, 2013.

[14] A. K. Jindal, „Protecting Android Devices Following BYOD Policy Against Data Security and Privacy Attacks,“ Indraprastha Institute of Information Technology, New Delhi, 2013.

[15] S. Stericson, „RootTools Github,“ [Tinkle]. Available: https://github.com/Stericson/RootTools. [Kreiptasi 27 02 2016].

[16] Samsung, „Samsung Knox,“ Samsung, [Tinkle]. Available: https://www.samsungknox.com/en. [Kreiptasi 25 02 2016].

[17] Wikipedia, „Samsung Knox,“ [Tinkle]. Available: https://en.wikipedia.org/wiki/Samsung_Knox#e-fuse. [Kreiptasi 22 03 2016].

[18] AirWatch, „AirWatch - Detecting Compromised Devices,“ [Tinkle]. Available: http://www.air-watch.com/downloads/brochures/AirWatch_White_Paper_-_Detecting_Compromised_Devices.pdf. [Kreiptasi 11 02 2016].

[19] T. Collyer, „SANS.org - Android, BYOD, and AirWatch MDM,“ [Tinkle]. Available: https://www.sans.org/reading-room/whitepapers/pda/airwatch-mdm-android-policy-technical-review-35372. [Kreiptasi 07 03 2016].

[20] E. Gruber, „Android Root Detection Techniques,“ 2 12 2013. [Tinkle]. Available: https://blog.netspi.com/android-root-detection-techniques/. [Kreiptasi 14 12 2015].

[21] Oracle, „Java Native Interface,“ Oracle, [Tinkle]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/jni/. [Kreiptasi 02 03 2016].

[22] D. Vlasenko, „About Busybox,“ [Tinkle]. Available: https://www.busybox.net/about.html. [Kreiptasi 20 01 2016].

[23] C. Substrate, Cydia, [Tinkle]. Available: http://www.cydiasubstrate.com/. [Kreiptasi 01 04 2016].

[24] „Android OTA Updates,“ Google, [Tinkle]. Available: https://source.android.com/devices/tech/ota/. [Kreiptasi 03 02 2016].

[25] „Vikipedija: Emuliatorius,“ [Tinkle]. Available: https://lt.wikipedia.org/wiki/Emuliatorius. [Kreiptasi 01 05 2016].

[26] S. O. Gasan Awad, „How to Stop Online and Mobile Banking Fraud,“ [Tinkle]. Available: http://www.paymentsjournal.com/WorkArea/DownloadAsset.aspx?id=24381. [Kreiptasi 02 04 2016].

[27] Google, „Android Studio,“ [Tinkle]. Available: http://developer.android.com/tools/studio/index.html. [Kreiptasi 16 01 2016].

[28] „Google Play: Barclays Mobile Banking,“ [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.barclays.android.barclaysmobilebanking. [Kreiptasi 14 03 2016].

[29] „Google Play: Star Finanz Sparkasse,“ Star Finanz, [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.starfinanz.smob.android.sfinanzstatus. [Kreiptasi 14 03 2016].

[30] „Google Play: Android Pay,“ Google, [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.walletnfcrel. [Kreiptasi 14 03 2016].

[31] „Google Play: BestBuy CinemaNow,“ BestBuy, [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.rmh.cinemanowUS. [Kreiptasi 14 03 2016].

[32] „Google Play: Excitor DME Mail,“ Excitor, [Tinkle]. Available: https://play.google.com/store/apps/details?id=dk.excitor.dmemail. [Kreiptasi 14 03 2016].

[33] „Google Play: Root Checker,“ Joey Krim, [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.joeykrim.rootcheck. [Kreiptasi 14 03 2016].

[34] „Google Android Platform Versions,“ [Tinkle]. Available: http://developer.android.com/about/dashboards/index.html. [Kreiptasi 22 04 2016].

[35] „Factory Images for Nexus Devices,“ Google, [Tinkle]. Available: https://developers.google.com/android/nexus/images. [Kreiptasi 02 04 2016].

[36] „TWRP System Recovery,“ [Tinkle]. Available: https://twrp.me/. [Kreiptasi 15 03 2016].

[37] „Paranoid Android Custom ROM,“ [Tinkle]. Available: http://www.paranoidandroid.co/. [Kreiptasi 01 04 2016].

[38] „Chainfire SuperSU,“ [Tinkle]. Available: https://download.chainfire.eu/696/SuperSU/. [Kreiptasi 27 04 2016].

[39] „Google Play: HideMyRoot,“ [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.amphoras.hidemyroot. [Kreiptasi 27 04 2016].

[40] „RootCloak 2,“ [Tinkle]. Available: http://repo.xposed.info/module/com.devadvance.rootcloak2. [Kreiptasi 01 04 2016].

[41] „Google Play: RootCloak Plus,“ [Tinkle]. Available: https://play.google.com/store/apps/details?id=com.devadvance.rootcloakplus. [Kreiptasi 02 04 2016].

[42] „Xposed Framework,“ [Tinkle]. Available: http://repo.xposed.info/module/de.robv.android.xposed.installer. [Kreiptasi 01 04 2014].

[43] „Genymotion Android Emulator,“ Genymotion, [Tinkle]. Available: https://www.genymotion.com/. [Kreiptasi 28 03 2016].