MYKOLO ROMERIO UNIVERSITETAS

VIEŠOJO VALDYMO IR VERSLO FAKULTETAS

GAUDRĖ ONA MARKAUSKIENĖ

SAUGIŲ MOBILIŲJŲ APLIKACIJŲ KŪRIMO IR VALDYMO YPATUMAI: GALIMYBĖS IR IŠŠŪKIAI

Magistro baigiamasis darbas

Vadovas

prof. dr. Tadas Limba

VILNIUS, 2021

MYKOLO ROMERIO UNIVERSITETAS

VIEŠOJO VALDYMO IR VERSLO FAKULTETAS

SAUGIŲ MOBILIŲJŲ APLIKACIJŲ KŪRIMO IR VALDYMO YPATUMAI: GALIMYBĖS IR IŠŠŪKIAI

Kibernetinio saugumo valdymo magistro baigiamasis darbas

Studijų programa 6211LX066

Vadovas

prof. dr. T. Limba

2021 05

Atliko

KSV19-1 gr. stud.

G. O. Markauskienė

2021 05 02

VILNIUS, 2021

TURINYS

LENTELĖS

PAVEIKSLAI

ĮVADAS

Temos aktualumas, naujumas. Verslas jau daugiau nei prieš dešimtmetį suprato, jog teikti paslaugas ir pardavinėti prekes internetu apsimoka – gerokai sumažėja kaštai ir padidėja pardavimai, mat pirkėjui patogu apsipirkti vos prisijungus prie interneto (Enders, Jelassi, 2005). Optimizuojant elektronines paslaugas verslas nuolat stengėsi tobulinti savo sistemas, kuo labiau supaprastinti apsipirkimą ar paslaugų teikimą. Valstybinėms įmonėms tenka nuo to neatsilikti, mat verslo diegiami standartai dėl prie to prisitaikiusių vartotojų lūkesčių pradeda galioti visoms elektroninėms paslaugoms (Norris, Reddick, 2013). Tačiau čia verslo ir technologijų sintezė nesibaigia. Per pastarąjį dešimtmetį įvykusi mobiliojo telefono evoliucija nuo paprasto susisiekimui skirto aparato iki išmaniojo prietaiso savaip pakeitė rinką (Fritz, ir kt., 2017). Šiuo metu jis atlieka daugelį funkcijų, anksčiau išskirtinai priklausiusių kitiems prietaisams: kompiuteriui (naršymas internete, el. laiškų rašymas ir pan.), fotoaparatui, užrašų knygutei, žadintuvui ir t. t. (Fritz ir kt., 2017). Verslo noras savo prekes ir paslaugas kaip įmanoma labiau priartinti prie vartotojo, pagalvoti už jį „į priekį“, paskatino mobiliųjų aplikacijų kūrimą ir išpopuliarėjimą (Nigam, Amirtharaj, 2018). Šiuo metu mobiliųjų aplikacijų (tyrime taip pat vadinamų mobiliosiomis programėlėmis arba tiesiog programėlėmis) yra sukurta visoms gyvenimo sritims: jas turi bankai, elektroninės parduotuvės, viešojo transporto operatoriai, švietimo, sveikatos paslaugas teikiančios įstaigos ir kituose rinkos segmentuose veiklą vykdančios organizaciijos. Daugiausiai mobiliųjų aplikacijų turinčioje mobiliųjų aplikacijų parduotuvėje „Google Play“ 2020 metų pabaigoje buvo patalpinta daugiau nei 3 140 000 programėlių, antroje pagal dydį „App Store“ – kiek daugiau nei 2 000 000 (Statista, 2020 m. IV ketvirtis).

Paradoksalu tai, kad, nors vis daugiau paslaugų keliama į mobiliųjų aplikacijų formatą, tik penktadalis jų neturi nei vienos saugumo spragos (Weir ir kt., 2020). Pastebima, jog jei saugumui ir saugumo testavimui būtų skiriama daugiau dėmesio nuo pat programėlių kūrimo pradžios, rastų spragų šalinimas vyktų ženkliai greičiau ir sklandžiau. (Weir ir kt., 2020, Kellner, 2019). Kibernetinio saugumo srityje besispecializuojanti įmonė „Forrester“ 2020 m. programinės įrangos (prie kurios priskiriama ir mobiliosios aplikacijos) analizėje teigia, jog mobiliosioms aplikacijoms turėtų būti taikomas bendras, daugiau procesų integruojantis požiūris į saugumą, tačiau toks modelis nėra sukurtas procentų (Carielli ir kt., Forrester, 2020). Šiai nuomonei pritariama ir moksliniuose šaltiniuose (Edison ir kt., 2018).

Pastebėtina, jog dvi pagrindinės mobiliųjų aplikacijų parduotuvės: „Google Play“, kurioje talpinamos „Android“ operacinei sistemai pritaikytos programėlės, ir „App Store“, skirta „iOS“ operacinei sistemai pritaikytoms aplikacijoms, nereikalauja iš programėlių kūrėjų, kad jos atitiktų saugumo standartus, o jų reikalavimai pateikiami gairių, kokias saugumo funkcijas programėlės turėtų

atlikti, pavidalu[1] („App Store“, ir „Google Play“ oficiali informacija programėlių kūrėjams, žiūrėta 2021 m. vasarį). Tuo tarpu jau minėta Forrester analizė atskleidė, jog būtent programinės įrangos spragos tampa silpnąja vieta ir savotiškais vartais į įvairių organizacijų informacines sistemas (Carielli ir kt., Forrester, 2020).

Svarbu pastebėti, jog mobilioji aplikacija, nors iš pažiūros gali pasirodyti savarankiška, yra ją sukūrusios organizacijos informacinių sistemų dalis (Fontao ir kt., 2015). Tai – programinė įranga, veikianti į tinklus sujungtų serverių, duomenų kaupyklų ir kitų prietaisų/aparatų aplinkoje (Rupnik, Krisper, 2009). Todėl ir mobiliųjų programėlių saugumas mokslinėje literatūroje dažnai nėra atskiriamas nuo bendrojo kibernetinio saugumo ir apibrėžiamas kaip atsparumas kenkėjiškiems trečiųjų šalių veiksmams: DDoS atakoms, įsiskverbimui į programinę įrangą (taigi ir į programėlę), informacijos gavimui naudojantis programavimo klaidomis ir pan. (Balebako ir kt., 2014, Weir ir kt., 2020, Yusop ir kt., 2016). Taigi natūralu, jog saugi programėlė, kaip ir kita programinė ar aparatinė įranga, turi atitikti ir pagrindinius kibernetinio saugumo kriterijus: konfidencialumo, vientisumo, ir prieinamumo principus (Treacy, McCaffery, 2016).

Temos ištirtumas. Pažymėtina, kad daugelis šioje srityje vykdytų mokslinių tyrimų (Weir ir kt., 2020, Kellner ir kt., 2019 Yusop ir kt., 2016, Balebako ir kt., 2014;) priėjo prie išvados, jog programėlių kūrėjai saugumui skiria per mažai dėmesio. Taip pat pažymima, jog nėra sukurto visuotinio standarto ar modelio, kuriuo remiantis programėlių kūrėjai kurtų visapusiškai saugias programėles (Edison ir kt., 2018, Carielli ir kt., Forrester, 2020).

Mokslinė problema. Visuotinai pripažįstamų standartų ir saugumo reikalavimų nebuvimas (Weir ir kt., 2017; Weir ir kt., 2020; Jha, Mahmoud, 2019) sukuria prielaidas skirtingoms saugių aplikacijų koncepcijų interpretacijoms, subjektyviam saugumo vertinimui, taigi ir prastesnei mobiliųjų programėlių saugumo situacijai. Tikėtina, jog į saugumą orientuotas programėlių kūrimo modelis padėtų programėlių kūrėjams skirti daugiau dėmesio mobiliųjų programėlių saugumui.

Tyrimo objektas: mobiliųjų aplikacijų saugumas

Tyrimo tikslas – pasiūlyti unifikuotą į saugumą orientuoto programėlių kūrimo ir valdymo modelį ir jo pagrindu pateikti rekomendacijas.

Tyrimo uždaviniai:

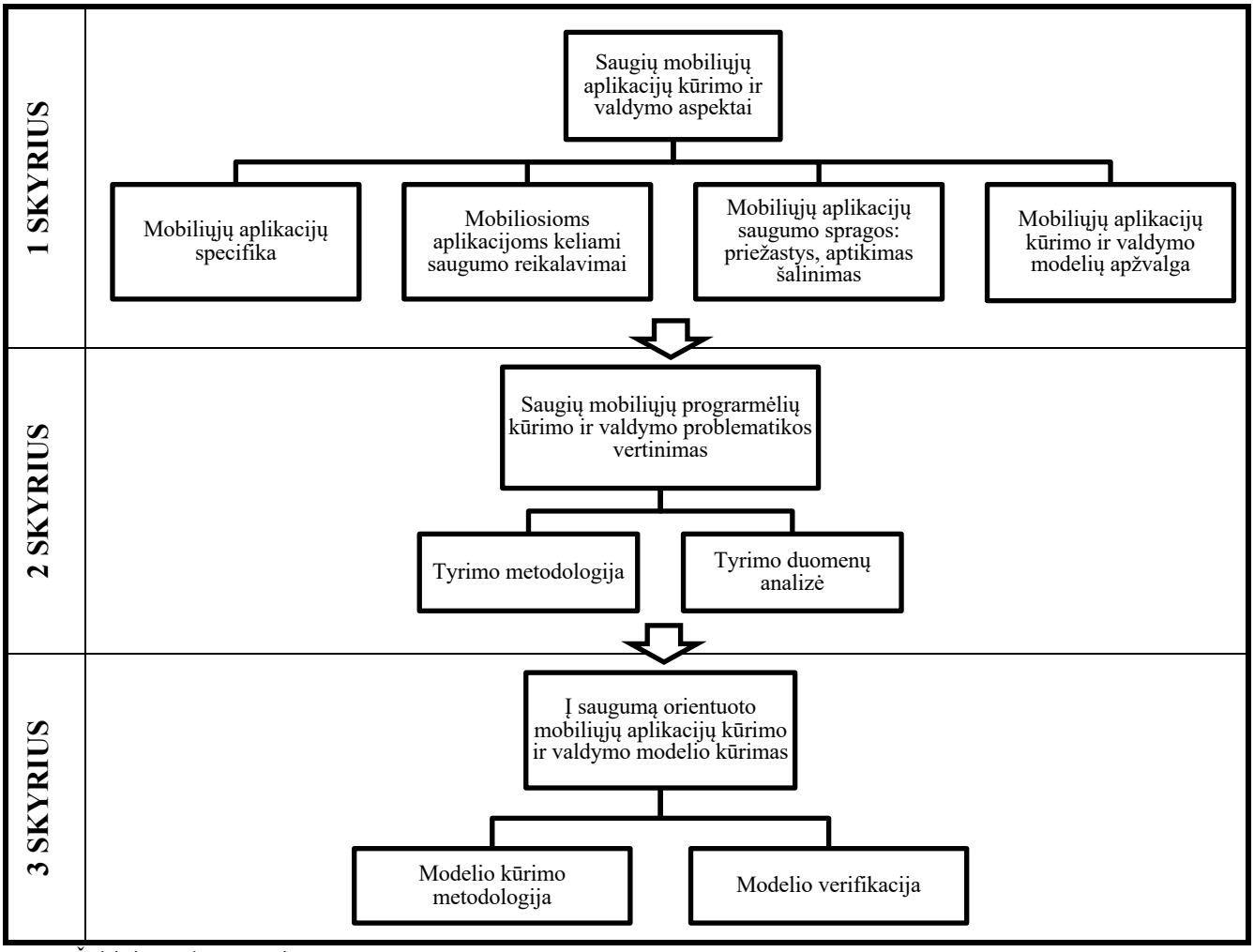
1. Išanalizuoti saugių mobiliųjų aplikacijų kūrimo ir valdymo teorinius aspektus;
2. Atlikti saugių mobilių aplikacijų kūrimo ir valdymo problematikos vertinimą;
3. Sukurti ir verifikuoti į saugumą orientuotą programėlių kūrimo ir valdymo modelį.

[1] Moksliniuose šaltiniuose taip pat randami ir Windows Phone parduotuvių duomenys, tačiau šiame darbe jie neapžvelgiami, nes Microsoft nusprendė nebenaujinti operacinės sistemos ir nebekurti šią operacinę sistemą palaikančių įrenginių, taigi ši parduotuvė tampa neaktualia.

Tyrimo metodai ir šaltiniai. Tyrimo uždavinių atlikimui pasitelkta mokslinės literatūros analizė, normatyvinė analizė, struktūruotas interviu, lyginamoji analizė, turinio analizė, modelio verifikavimas. Mokslinės literatūros analizės, normatyvinės analizės ir turinio analizės metodais siekiama išnagrinėti teorinius saugių mobiliųjų aplikacijų kūrimo ir valdymo aspektus. Atliekant mokslinės literatūros analizę augiausiai remiamasi C. Weir ir kt., N. Yusop ir kt., R. Balebako, A. Kellner ir kt, bei kitų mokslininkų darbais. Normatyvinės analizės metodu nagrinėjami teisiniai reikalavimai saugioms mobiliosioms aplikacijoms. Turinio analizė pasitelkta nagrinėjant mokslininkų sukurtus mobiliųjų aplikacijų kūrimo ir valdymo modelius. Atliekant turinio analizę daugiausia remiamasi H. Edison ir kt., V. Van Cesteren, M. Pamdey ir kt. tyrimais. Struktūruoto interviu metodas pasitelktas tyrimui, kurio tikslas – įvertinti praktinius saugių mobiliųjų programėlių kūrimo ir valdymo aspektus, jų problematiką. Tyrimo rezultatai buvo apdorojami „MaxQDA“ programa, pasitelkiant lyginamąją bei turinio analizę. Modelio verifikavimas atliekamas remiantis Hicks ir kt. rekomendacijomis. Modelis kuriamas remiantis M. Fowler ir kt. sukurta „Agile“ metodologija, C. Weir ir kt. pasiūlytais Dialektikos žingsniais bei N. Newton ir kt. apibrėžtais Kritiniais sėkmės faktoriais.

Darbo struktūra. Darbas sudarytas iš 3 dalių. Pirmojoje dalyje nagrinėjami saugių mobiliųjų aplikacijų kūrimo ir valdymo aspektai: mobiliųjų programėlių specifika, joms keliami saugumo reikalavimai, mobiliųjų aplikacijų saugumo spragų atsiradimo priežastys bei aptikimo ir šalinimo būdai, apžvelgiami mokslinėje literatūroje pateikiami programėlių kūrimo ir valdymo modeliai. Antrojoje dalyje aprašomas ekspertinis tyrimas, kurio tikslas – pasitelkiant ekspertų žinias įvertinti saugių mobiliųjų aplikacijų kūrimo ir valdymo problematiką. Trečiojoje darbo dalyje kuriamas į saugumą orientuotas programėlių kūrimo ir valdymo modelis bei atliekama jo verifikacija. Darbo pabaigoje pateikiamos išvados ir rekomendacijos. Darbo struktūros loginė schema pavaizduota 1 paveiksle.

Tyrimo teorinis ir praktinis reikšmingumas. Darbo reikšmingumą įrodo tai, kad išnagrinėjus teorinius mobiliųjų aplikacijų saugumo aspektus bei pasitelkus ekspertinio tyrimo metu išryškėjusias tendencijas buvo sukurtas unifikuotas į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis, kurio trūkumą pastebėjo darbo problemoje minėti mokslininkai. Modelio tikslas – padėti organizacijoms kurti saugias *by design* programėles, o modelio universalumas užtikrina, jog modelį gali pritaikyti bet kuri mobiliąsias aplikacijas kurianti ar valdanti organizacija.



Šaltinis: sudaryta tyrimo autorės

1 pav. Darbo struktūros loginė schema

1. SAUGIŲ MOBILIŲJŲ APLIKACIJŲ KŪRIMO IR VALDYMO TEORINIAI ASPEKTAI

1.1. Mobiliųjų aplikacijų sąvokos ir raidos analizė

Siekiant kuo tiksliau išsiaiškinti saugių mobiliųjų aplikacijų kūrimo ir valdymo[2] galimybes bei iššūkius svarbu identifikuoti kuo mobiliosios programėlės panašios ir kuo skiriasi nuo kitų programinės įrangos rūšių. Kembridžo žodyne pateikiamas mobiliosios aplikacijos (tyrime taip pat vadinamos mobiliąja programėle arba tiesiog programėle) apibrėžimas nurodo, jog tai – „programinės įrangos rūšis, skirta veikti mobiliajame telefone“ (Cambridge dictionary, žiūrėta 2021 m. vasarį). Analogišką apibrėžimą 2009 metais pasiūlė ir R. Rupnik bei M. Krisper, tačiau šis sąvokos aiškinimas tikriausiai vienas siauriausių, kokį galima rasti atviruose bei moksliniuose šaltiniuose. Kiek platesnį mobiliosios aplikacijos apibrėžimą pasiūlė A. Šuminas ir A. Aleksandravičius (2013), teigiantys, jog mobilioji programėlė yra „[...] programinės įrangos paketas, kurį mobiliųjų įrenginių naudotojai gali parsisiųsti ir įdiegti savo turimuose prietaisuose. [...] mobiliosios programėlės yra skirtos išmaniesiems įrenginiams, tokiems kaip išmanieji telefonai ar planšetiniai kompiuteriai [...]“. Tuo tarpu D. Amalfitano, A. Fasolino, P. Tramontana, ir B. Robbins (2013) akcentuoja mobiliųjų aplikacijų savarankiškumą, atskirumą (angl. *self-containment*) ir specifiškumą: „[mobilioji aplikacija] autonominė programinė įranga, sukurta mobiliajam įrenginiui ir atliekanti specifines mobiliųjų įrenginių vartotojų funkcijas“. Kadangi šiame apibrėžime akcentuojamas programėlės specifinės paslaugos teikimas, specifinio vartotojo poreikio patenkinimas, kurį kaip vieną pagrindinių mobiliųjų aplikacijų skiriamųjų bruožų išskyrė daugelis mokslininkų (Weir ir kt., 2017, Pandey ir kt., 2018 bei kiti), minėtas apibrėžimas šiame tyrime laikytinas pamatiniu.

Visuose apibrėžimuose mobiliosios aplikacijos vadinamos „programine įranga“, kuri Kembridžo žodyne aiškinama kaip „instrukcijos, kuriomis kontroliuojami kompiuterio veiksmai, kompiuterinė programa“ (Cambridge dictionary, žiūrėta 2021 m. vasarį). Taigi, programinė įranga dažniausiai suprantama kaip kompiuteriams skirta programine įranga (Salini, Kanmani, 2016). Tačiau nors mobiliosios aplikacijos laikytinos programinės įrangos rūšimi, tarp jų ir programinės įrangos yra keli fundamentalūs skirtumai (Rupnik, Krisper, 2009):

[2] mobiliosios aplikacijos valdymas tyrimo apimtyje suprantamas kaip jos palaikymas, naujinių kūrimas ir diegimas, kylančių problemų sprendimas ir pan.

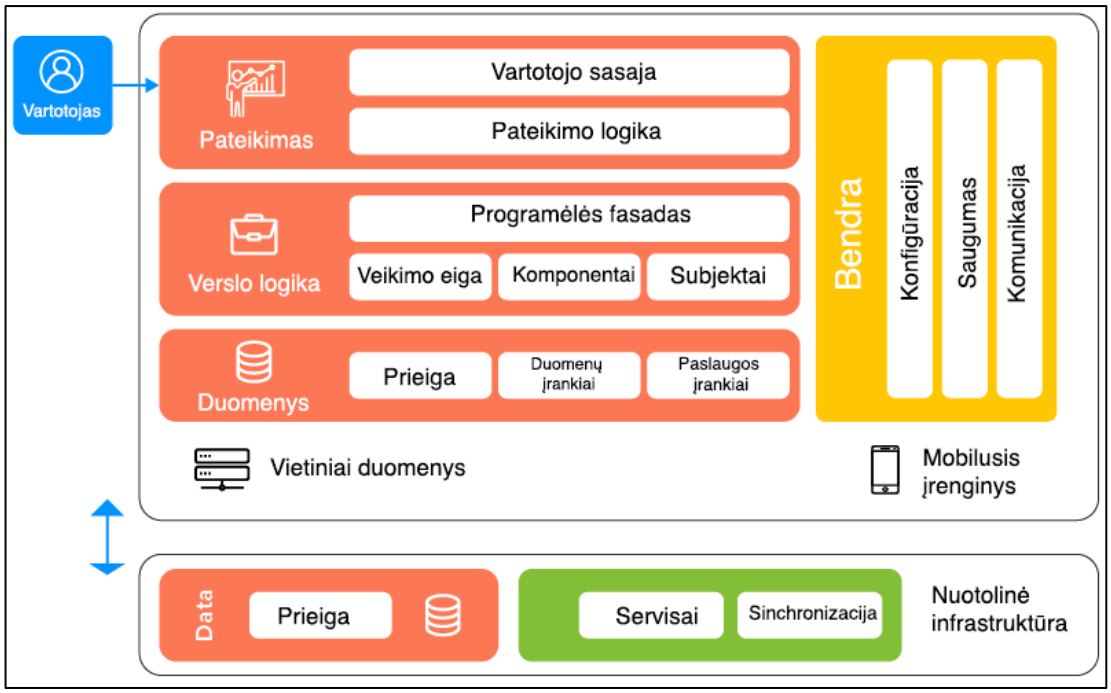
1 lentelė. Programinės įrangos ir mobiliųjų programėlių palyginimas

Programinė įranga	Mobilioji programėlė
Plačios paskirties duomenų ar instrukcijų rinkinys, valdantis aparatinę įrangą (angl. *hardware*)	Konkrečią, specializuotą ar net nišinę informaciją ar paslaugą teikianti programinis paketas
Veikimas nepritaikytas konkrečiam operacinės sistemos kontekstui	Veikia konkrečios operacinės sistemos kontekste
Gali veikti be vartotojo veiksmų.	Orientuota į mobilumą, prisitaikyta vartotojo poreikiams, veikimui būtinas vartotojo veiksmas.
Programinė įranga nebūtinai yra programa/mobilioji programėlė	Mobilioji programėlė yra viena iš programinės įrangos rūšių

Šaltinis: Sudaryta tyrimo autorės remiantis R. Rupnik ir M. Krisper, 2009

Būtent ryški specializacija ir prisitaikymas prie vartotojo poreikių ir yra didžiausi mobiliųjų aplikacijų privalumai, aktualūs tiek verslo organizacijoms tiek vartotojams ir lėmę mobiliųjų programėlių išpopuliarėjimą. (Leung ir kt., 2016).

Mobiliųjų aplikacijų architektūrą įprasta nagrinėti trimis pjūviais: prezentaciniu, verslo logikos bei duomenų (Žr. 1 paveikslą) (Taneja ir kt., 2015) . Prezentacinis lygmuo – tai, ką mato vartotojas, tačiau šiame lygmenyje duomenys tik atvaizduojami. Pati programėlės esmė, tai, kokią ji paslaugą siūlo vartotojui, kuriama verslo logikos lygmenyje. Į šį lygmenį įeina programėlės funkcijos ir darbo eiga (*workflow*), komponentai, subjektai. Duomenų lygmenyje apdorojami vartotojo įvesti duomenys, apsprendžiama, prie kokių duomenų vartotojas gali prieiti, kuriamos sąsajos tarp duomenų, gali būti kuriamos duomenų saugyklos. Savo ruožtu mobilioji programėlė veikia mobiliojo prietaiso aplinkoje bei nuolat sąveikauja su išoriniais informaciniais ištekliais (Taneja ir kt., 2015).

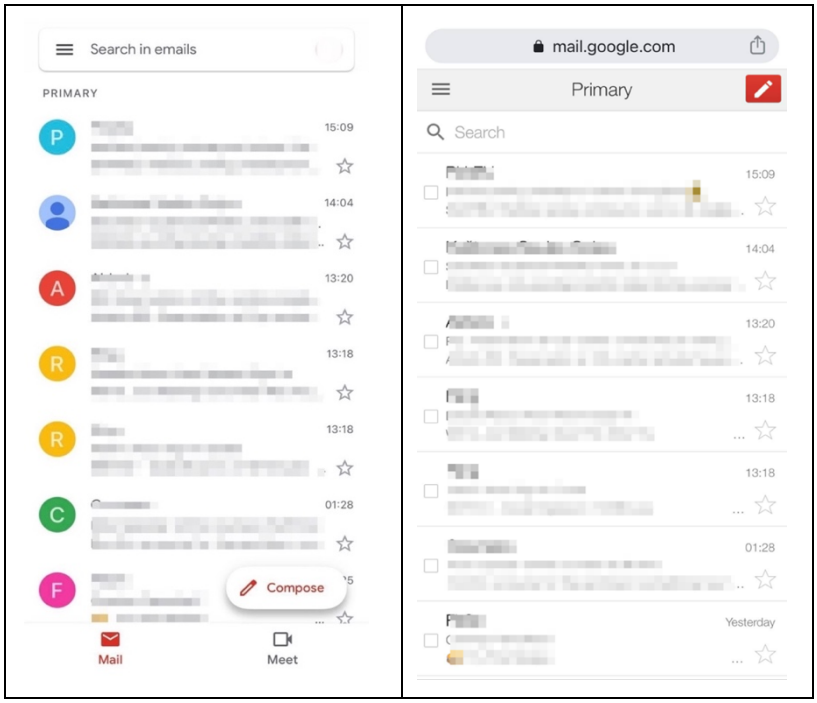


Šaltinis: Apinventiv, originalus šaltinis – IBM

2 pav. Trijų sluoksnių aplikacijos architektūra

Raudona spalva pažymėti ištekliai, kuriuos dažniausiai valdo programėlę valdanti organizacija (mobiliosios aplikacijos valdytojas): t. y., pati programėlė, programėlės duomenų bazė, esanti mobiliajame įrenginyje bei išorinį serverį, kuriame patalpinta išorinė duomenų bazė ir duomenų saugykla (Appinventiv, 2020). Pastebėtina, jog skirtingų programėlių duomenų saugojimo sprendimai skiriasi, pavyzdžiui, ne visos jos saugo duomenis pačioje programėlėje esančioje duomenų bazėje ir yra suprogramuotos taip, kad visą informaciją gauna iš ir siunčia į už telefono ribų esantį serverį, tiksliau jame patalpintą *backendą*, kuriame apsprendžiama visa programėlės veikimo logika. Tokiu atveju programėlės vaidmuo apsiriboja duomenų atvaizdavimu vartotojui ir prieiga prie mobiliojo įrenginio funkcijų (IBM, 2020). Dėl šios priežasties darbe nagrinėjama ne tik mobilioji programėlė, esanti mobiliajame įrenginyje, bet ir su ja susietos duomenų bazės, saugyklos bei komunikacja tarp šių lygmenų. Taip pat darytina išvada, kad programėlė dažnai labai glaudžiai įkorporuojama į programėlės valdytojo informacines sistemas ir nors vartotojui atrodo savarankiška, ją tikslinga nagrinėti kaip visos ją valdančios organizacijos informacinės ekosistemos dalį (de Lima Fontao ir kt., 2015).

Pagal veikimo principą mobiliosiois aplikacijos gali būti skirstomos į tris rūšis: vietines (angl. *native*), naršyklines (angl. *web apps*) arba hibridines, tarpinį variantą tarp vietinės ir naršyklinės aplikacijų (Rakestraw ir kt., 2012). Vietinės ir hibridinės programėlės turi būti įdiegtos į mobilųjį įrenginį, tuo tarpu naršyklinės pasiekiamos per mobiliajame įrenginyje įdiegtą naršyklę. Nors visos trys programėlių rūšys iš esmės atlieka vienodą ar labai panašią funkciją, jos skiriasi tiek savo pajėgumu, tiek prieinamumu, tiek kaina (Leung ir kt., 2016). Vietinė programėlė turi platesnį funkcionalumą,



Šaltinis: sudaryta tyrimo autorės

3 pav. El. pašto „Gmail“ mobilioji aplikacija (kairėje) ir internetinė (web) aplikacija

didesnę prieiga prie mobiliojo įrenginio funkcijų (tokių kaip kamera, kompasas ir pan.), gali rodyti vartotojui pranešimus (angl. *push notifications*), tačiau jos kūrimas brangesnis, nes kainuoja aplikacijų kūrimo įrankiai bei reikia kurti kelias programėlės versijas, pritaikytas skirtingoms operacinėms sistemoms. Tuo tarpu naršyklinės aplikacijos prisitaiko prie bet kurio įrenginio ir dažnai yra susiejamos su internetinėmis svetainėmis, todėl pakeitus informaciją svetainėje ji automatiškai pasikeičia ir naršyklinėje programėlėje. (Lung ir kt., 2016). Dėl to naršyklinės programėlės kūrimas ir palaikymas yra gerokai pigesnis nei vietinės mobiliosios programėlės. Kita vertus, ši programėlė veikia tik kai mobilusis įrenginys yra prisijungęs prie interneto, jos funkcionalumas ribotas, ji neturi priėjimo prie mobiliojo įrenginio aparatinių funkcijų (Rakestraw ir kt., 2012). 3 paveiksle pateikiamas mobiliosios ir naršyklinės programėlės versijų pavyzdys.

Tačiau prieš kelerius metus kartu su programavimo kalbos HTML5 atsiradimu išpopuliarėjo ir trečia aplikacijų rūšis, turinti tiek vietinės mobiliosios, tiek naršyklinės aplikacijų požymių. Tai – lyg į vietinės mobiliosios aplikacijos apvalkalą įvilkta naršyklinė aplikacija. Ją galima parsisiųsti iš mobiliųjų aplikacijų parduotuvės, ji turi prieigą prie mobiliojo įrenginio aparatinių funkcijų, tačiau jos veikimui būtinas interneto ryšys, ji neturi tokio didelio veikimo greičio ir galimo funkcionalumo kaip vietinė mobilioji aplikacija. Tuo pat metu lyginant su vietinėmis mobiliosiomis programėlėmis, ji pasižymi gerokai mažesniais kūrimo bei palaikymo kaštais (Sidana, Medium.com, 2015). Susisteminti visų trijų programėlių rūšių panašumai ir pateikiami 2 lentelėje.

2 lentelė. Mobiliosiųjų vietinės ir hibridinės bei naršyklinės aplikacijų skirtumai

	Vietinė mobilioji aplikacija (*native mobile app*)	Naršyklinė aplikacija (*web app*)	Hibridinė aplikacija (*hybrid app*)
Programėlę reikia parsisiųsti į įrenginį	+	–	+
Programėlė pritaikyta konkrečiai sistemai	+	–	–
Gali veikti neprisijungus prie interneto	+	–	–
Didelis programėlės pajėgumas ir funkcionalumas	+	–	+/–
Nedidelės kūrimo ir valdymo sąnaudos	–	+	+

Prieiga prie įrenginio aparatinių funkcijų	+	–	+
Galimybė siųsti vartotojui pranešimus (*notifications*)	+	–	+

Parengta tyrimo autorės pagal Leung ir kt., 2016 ir Rakestraw ir kt., 2012 informaciją.

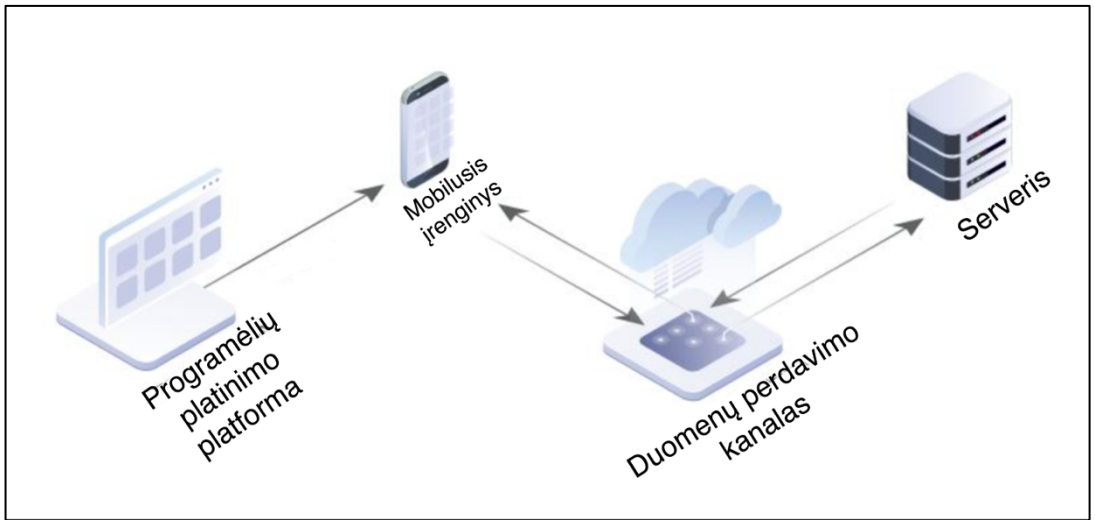
Kadangi tiek vietinė, tiek hibridinė programėlės turi būti parsisiųstos bei turi vienodą priėjimą prie mobiliojo įrenginio aparatinių funkcijų, jų pačių funkcionalumas, taigi ir saugumo iššūkiai panašūs, toliau darbe bus nagrinėjamos šios dvi aplikacijų rūšys, vadinamos bendru pavadinimu: **mobiliosiomis programėlėmis** arba **mobiliosiomis aplikacijomis.**

Kitas pjūvis, kuriuo dažnai nagrinėjamos mobiliosios programėlės – vadinamieji *frontendas* ir *backendas*[3]. *Frontendas* bet kurioje programinėje įrangoje (taigi, ir mobiliosios aplikacijos kontekste) suprantamas kaip vartotojo sąsaja, tai, ką vartotojas mato ir valdo liečiamuoju ekranu, pelyte ar kitais būdais (Cambridge dictionary, žiūrėta 2021 m. vasarį). Tuo tarpu *backendas* – vartotojui nematoma programinės įrangos veikimo logika ir duomenų saugyklos, kurios šios logikos pagalba tampa pasiekiamos ir iš kurių pateikiami duomenys (Cambridge dictionary, žiūrėta 2021). *Frontendas* ir *backendas* dažniausiai yra programuojami skirtingomis kalbomis, pavyzdžiui, *fronendo* kodas dažniausiai rašomas HTML5, CSS, JavaScript, o *backendo* – PHP, Ruby, Python kalbomis (Abdullah ir kt., 2014). Dažnai programuotojai specializuojasi *frontendo* arba *backendo* programavime.

Dauguma mobiliųjų aplikacijų veikia *Client-Server* (klientas-serveris) architektūros principu, kai klientas – į mobilųjį įrenginį parsisiųsta programėlė, o serveris – ją palaikantys serveriai, duomenų kaupyklos ir pan. (Abdullah ir kt., 2014). Norėdamas perskaityti elektroninį laišką, nusipirkti prekę ar susirasti reikiamą adresą navigacijoje vartotojas sąveikauja su programėle. Ši, savo ruožtu, vartotojo užklausas pateikia ir atsakymus gauna iš programėlę valdančios organizacijos serverio. Ši sąveika pavaizduota 4 paveiksle. Todėl nors vartotojas kartais programėlę suvokia kaip savarankišką jo telefone veikiančią programą, svarbu suvokti, jog iš tiesų mobilioji aplikacija veikia susietos programinės ir aparatinės įrangos kontekste (Fontao ir kt., 2015).

Informacija tarp kliento ir serverio *Client-Server* architektūroje perduodama specialia sąsaja, angl. *Aplication Programming Interface*, arba API. API – tai sąsaja leidžianti skirtingoms aplikacijoms keistis duomenimis tarpusavyje, ji nusako duomenų keitimosi logiką ir būdus, t.y., kokie duomenys ir kaip gali būti priimami arba perduodami (Egele ir kt., 2013). Svarbu paminėti, jog būtent API mokslininkų ir ekspertų vertinimu sukuria programėlėse daugiausia saugumo spragų (Weir ir kt., 2020; Veracode, 2020, S. Carielli ir kt., Forrester, 2020 ir kiti).

[3] Lietuviškų atitkimenų šiems dviems terminams nėra pasiūlyta, todėl čia ir toliau darbe bus naudojami šie anglicizmai.



Šaltinis: Positive technologies, 2019

4 pav. Mobiliosios programėlės sąveikos su serveriais schema

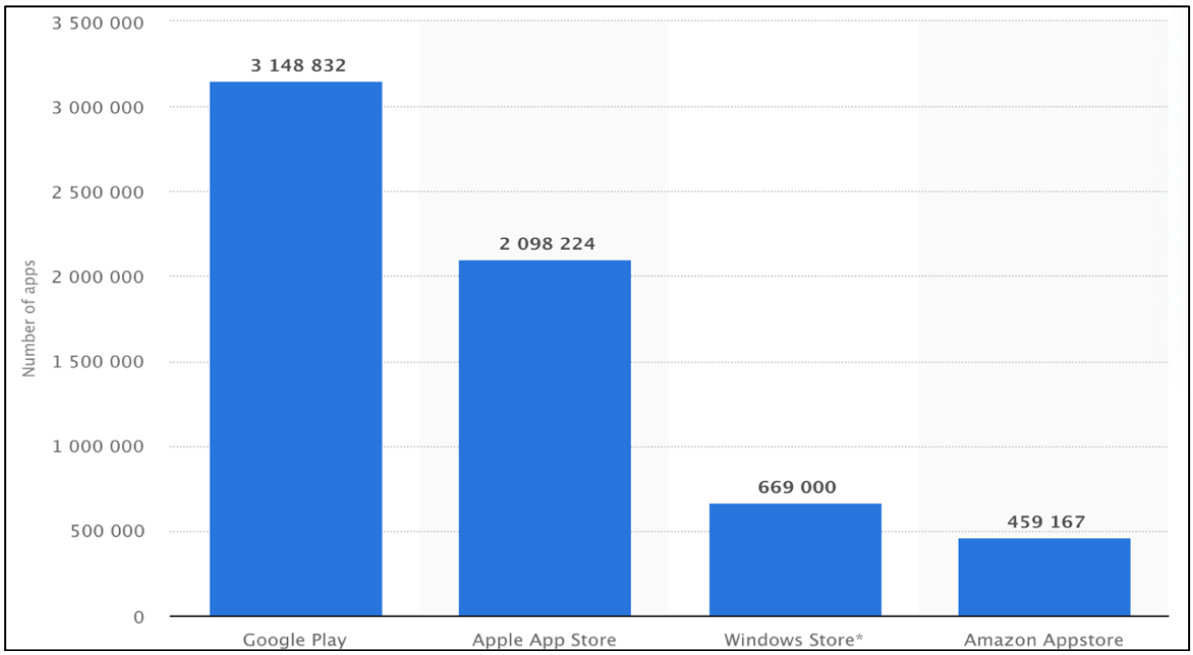
1.2. Saugių mobiliųjų aplikacijų plėtros problematikos analizė

Prieš nagrinėjant programėlių saugumo aspektus svarbu apibrėžti, kas yra programėlių saugumas. Mobiliųjų aplikacijų saugumas mokslinėje literatūroje aiškinamas kaip atsparumas trečiųjų šalių veiksmams (Weir ir kt., 2020;, Balebako ir kt., 2014): DDoS atakoms, informacijos perėmimui ar nutekinimui įsiskverbiant į programinę įrangą ir pan. (Yusop ir kt., 2016). Tačiau programėlių vartotojų saugumui ne mažiau aktualus programėlės valdytojų požiūris į asmens duomens bei sąžiningumas prieš vartotojus, jam atskleidžiant kokiais tikslais ir kokie jo duomenis bus naudojami (Utz ir kt., 2019; Wottrich ir kt., 2018), kitaip tariant – korektiškumas vartotojo asmens duomenų atžvilgiu. Tuo tarpu kibernetinio saugumo ekspertai, tinklaraštininkai May Ying Tee ir Marting Zhang (2018) išskyrė ir trečią saugios programėlės kriterijų – programėlė neturi būti savaime kenkėjiška. Pastarojo tipo programėlės yra sukurtos taip, kad jas parsisiuntus mobiliajame įrenginyje paleidžiamas kenkėjiškas kodas, kurio pagalba įrenginio vartotojas gali būti slapta sekamas, siekiama išgauti finansinę ar asmeninio pobūdžio informaciją apie vartotoją ir pan. (May Ying Tee, Marting Zhang, 2018). Kadangi kenkėjiško tipo programėlės yra nesaugios vartotojui nuo pat jų iniciacijos, t. y., jos kuriamos kenkėjiškais tikslais, darbe jas nagrinėti nėra tikslinga. Todėl **saugi mobilioji aplikacija šio tyrimo apimtyje suprantama kaip atspari kenkėjiškiems trečiųjų šalių veiksmams bei korektiška vartotojo asmens duomenų atžvilgiu**.

Pagrindinis tiek moksliniuose, tiek specializuotuose publicistiniuose šaltiniuose siūlomas patarimas vartotojams, siekiantiems apsaugoti savo telefoną bei joje esančią informaciją – siųstis

programėles tik iš gerai žinomų mobiliųjų aplikacijų parduotuvių. Programėlių parduotuvės teigia, jog prieš patalpindamos aplikacijas savo erdvėje jas testuoja ir priima tik tas, kurios yra saugios, korektiškos ir sukuria vertę vartotojui. Tačiau tiek specializuotoje spaudoje, aprašančioje technologines naujienas ir aktualijas, tiek ir mokslinėje literatūroje, apstu informacijos jog jau nuo 2011 m. šie aplikacijų parduotuvių veiksmai yra nesunkiai apeinami (Campbell, 2014, NowSecure, 2019).

Nors kalbant apie mobiliųjų programėlių platinimo vietas įprasta vartoti žodį „parduotuvė“, svarbu paminėti, jog dauguma mobiliųjų programėlių parduotuvėje turi nemokamas versijas, kurios monetizuojamos programėlės vartotojams rodomomis reklamomis. Tuo tarpu mokamų programėlių kaina – vos keli doleriai. „Statista“ duomenimis vidutinė „App Store“ programėlės kaina – 1,02 USD (Statista, 2021). Šiuo metu pasaulyje veikia kelios dešimtys mobiliųjų aplikacijų parduotuvių, tačiau didžiausią rinkos dalį užima „Google Play“, skirta „Android“ operacinei sistemai sukurtoms programėlėms, „App Store“, kurioje patalpintos „iOS“ operacinei sistemai pritaikytos aplikacijos ir „Windows Store“, skirta Windows operacinės sistemos programėlėms. Kaip jau minėta, „Windows Store“ valdanti kompanija „Microsoft“ nusprendė nebetęsti „Windows phone“ ir jam skirtų aplikacijų parduotuvės vystymo, todėl „Windows Store“ šiame darbe nenagrinėjama nebus. Taip pat paminėtinos ir ketvirta pagal populiarumą pasaulyje programėlių parduotuvė „Amazon“ („Android“), korporacijos „Tencel“ valdoma „MyApp“ („Android“), Kinijoje užimanti 25% rinkos, bei Lietuvoje 2004 metais įkurta „GetJar“ (įvairioms operacinėms sistemoms pritaikytos programėlės), ekspertų vertinama kaip viena iš didžiausių alternatyvių mobiliųjų programėlių parduotuvių (Business of Apps, 2021).



Šaltinis – Statista, duomenys – 2020 m. IV ketvirtis.4 pav.

5 pav. Mobiliųjų aplikacijų parduotuvėse patalpintų aplikacijų skaičius

5 paveiksle galima matyti, jog „Google Play“ ir „App Store“ šiuo metu – ryškūs mobiliųjų programėlių parduotuvių rinkos lyderiai, taigi programėlių kūrėjams gali diktuoti savo sąlygas, nustatyti standartus. Dėl to šiame darbe didžiausias dėmesys skiriamas šioms dviem parduotuvėms.

Vis garsiau skamba ekspertų perspėjimai, kad programėlių parduotuvių įvaizdis sukuria vartotojams „apgaulingą saugumo jausmą“ ir jie nepakankamai atidžiai vertina galimas su programėle susijusias rizikas (Kellner ir kt., 2019). Nepaisant deklaratyvių aplikacijų parduotuvių kalbų apie programėlių testavimą, pro jų filtrus visgi prasprūsta kenkėjiško tipo programėlės (Ying Tee, Zhang, 2018). Google Play dėl „Android“ operacinės sistemos didesnio paplitimo (maždaug 90 proc. visų mobiliųjų įrenginių veikia viena iš „Android“ operacinės sistemos versijų), atviro kodo, didesnės operacinės sistemos „laisvės“ (palyginus su Apple operacine sistema, „Android“ leidžia daugiau funkcinių ir kitų nustatymų keisti pačiam vartotojui, kas potencialiai gali sumažinti įrenginio saugumą) ir kitų savybių – labiau pažeidžiama (Mutchler ir kt., 2015). Spaudoje nuolat pasirodo informacija apie Google Play parduotuvėje atsiradusią ir daug kartų parsisiųstą kenkėjišką ar kenkėjišku kodu infiltruotą programėlę, o kartais tokios programėlės net atsiranda tarp populiariausių (C. Welch, Theverge.com, 2014).

Viena dažniausių „Android“ programėlių problemų – perteklinis programėlių vykdomas vartotojo lokacijos sekimas (Mutchler ir kt., 2015). Šį trūkumą operacinės sistemos kūrėjai stengiasi pašalinti kartu su sistemos atnaujinimais, leidžiančiais išjungti atskiroms programėlėms lokacijos sekimą (Z. Doffman, Forbes, 2019), tačiau tam, kad šie atnaujinimai veiktų, vartotojai patys turi suvokti savo asmens duomenų konfidencialumo problemą ir imtis papildomų veiksmų – neleisti programėlėms sekti jų lokacijos. Kiti veiksmai, kurių ėmėsi „Android“ pagrindinė vystytoja, kompanija „Google“ – „App Defence Alliance“ kūrimas su dar trimis kompanijomis ir naujo produkto „Google Play Protect“[4] kūrimas ir vystymas. Tačiau kibernetinio saugumo ekspertai nemano, jog tai išspręs visas problemas, nes veiksmų, jų nuomone, buvo imtasi gerokai per vėlai, kai nuo iš šios parduotuvės parsisiųstų programėlių jau yra nukentėję daug žmonių, maža to, kenkėjišką programinę įrangą kuriantys programišiai taip pat nuolat tobulėja, naujų įsiskverbimo būdų nuolat daugėja (Z. Doffman, Forbes, 2019; J. Hilderbrand, AndroidCentral, 2020 ir kiti).

„App Store“ požiūris į saugumą kiek kitoks. Visų pirma, „iOS“ nėra atviro kodo operacinė sistema. Antra, „App Store“ prieš padarydama aplikaciją visiems prieinama naudoja vadinamas smėlio dėžes (angl. s*andbox*) – savotišką karantiną naujoms programėlėms ir saugią aplinką jos kodo testavimui („App Store“ informacija, žiūrėta 2021 vasarį). Tai padeda ženkliai sumažinti kenkėjiškų programėlių kiekį. Nepaisant to, nesaugiai suprogramuotas mobiliąsias aplikacijas abiem aplikacijų parduotuvėms

[4] „Google Play Protect“ – programėlė, kuri identifikuoja nesaugias telefone esančias programėles ir, jei jos kenkėjiško pobūdžio, jas ištrina arba apie jas kaip apie nesaugias praneša vartotojui.

pavyksta aptikti vienodai. „Google Play“ patalpintos programėlės, kurių saugumo spragos kelia didelę grėsmę 2019 metais siekė 43%, „App Store“ – 38% (Positive Technologies ataskaita, 2019 m.).

Nepaisant šios statistikos bei potencialios saugumo spragų keliamos grėsmės visų mobiliųjų programėlių vartotojams, nei „Google Play“, nei „App Store“ neturi konkrečių viešai prieinamų saugumo standartų, kuriuos turi atitikti jų parduotuvėse platinamos programėlės ir kuriais galėtų remtis programėlių kūrėjai. Pateikiamos tik rekomendacijos, kuriomis vadovaudamiesi programuotojai turėtų sukurti saugias mobiliąsias aplikacijas („Google Play“ ir „App Store“ pateikiama oficiali informacija, žiūrėta 2021 m. vasarį). Kitaip tariant, programėlių kūrėjams, programuotojams, paliekama kūrybos laisvė, bet tuo pačiu ir visa atsakomybė už programėlių, o kartu ir jų vartotojų, saugumą.

Nors „App Store“ puslapyje, skirtame programuotojams, teigiama, jog programėlės kūrėjai turi užtikrinti visą jos vartotojų asmens duomenų konfidencialumą, tačiau kokiais techniniais sprendimais tai būtų galima pasiekti, nepataria. Puslapyje daugiau akcentuojamas programėlės nekenkėjiškumas, pavyzdžiui, reikalaujama, kad aplikacijose nebūtų kenksmingo kodo, kad programėlė veiktų „konteinerio“ principu – nenuskaitinėtų ir neįterpinėtų informacijos už savo konteinerio ribų, ir pan. Kiek detaliau akcentuojamas programėlių vartotojų duomenų saugumo politika. Tarp kitų, parduotuvė pateikia tokias rekomendacijas:

1. Aiški ir išsami aplikacijos privatumo politika, su kuria vartotojas turi būti supažindintas prieš naudojantis programėle;
2. Programėlė turi gauti vartotojo leidimą naudotis mobiliojo įrenginio funkcijomis (pavyzdžiui, kamera) bei įrenginyje saugoma informacija (pavyzdžiui, nuotraukomis);
3. Turi būti laikomasi duomenų minimalumo principo, nereikalauti vartotojų prisijunginėti ir kurti paskyrų, jei tai nėra būtina jos funkciniam veikimui;
4. Pirmam nuotraukų kėlimui, mikrofono ir kameros naudojimui būtinas atskiras vartotojo sutikimas;
5. Draudžiama sekti vartotojų slaptažodžius, lokaciją, kitą informaciją jiems patiems to nežinant. Už tokius pažeidimus „App Store“ žada sankcijas („App Store“ informaciija, žiūrėta 2021 vasarį).

Šalia šių rekomendacijų „App Store“ reklamuoja savo programėlių kūrimo įrankius, pateikia programinio kodo pavyzdžius. Taip pat yra teikiama techninė pagalba, jei nepavyksta pritaikyti programėlės „iOS“ operacinei sistemai. Pažymėtina ir tai, kad programėlės testuojamos „App Store“ testuotojų, taigi nors konkrečių reikalavimų jie neteikia, jei programėlė turi spragų, ji į parduotuvę nepraleidžiama, o jos kūrėjams pranešama, ką reikia sutvarkyti („App Store“ informacija, žiūrėta 2021 m. vasarį).

Minėtas rekomendacijas dėl programėlės korektiškumo vartotojo duomenų, jo mobiliojo aparato bei mobiliųjų programėlių atžvilgiu savo informacinėje medžiagoje pabrėžia ir „Google Play“. Tačiau

šioje parduotuvėje, skirtingai nei „App Store“, pateikiamas ir išsamų programėlės kūrimo proceso aprašymas: nuo ko pradėti, į ką atkreipti dėmesį programuojant, kaip testuoti programėlę, kas svarbu kiekviename proceso žingsnyje. Nepaisant to, išskyrus tam tikrus konkrečius techninius reikalavimus, pavyzdžiui, dėl jau minėtų API sąsajų, šie žingsniai taip pat galėtų būti prilyginami patarimams ar gairėms, nes yra gana abstraktūs. („Google Play“ informacija, žiūrėta 2021 m. vasarį).

1.3. Mobiliosioms programėlėms taikomų reikalavimų analizė

1.3.1. Reikalavimai infrastruktūrai

Vartotojo mobiliajame įrenginyje atvaizduojama programinės įrangos dalis yra tik viena iš visos aplikacijos ekosistemos dalių (Rupnik, Kremer, 2009). Būtent dėl šios priežasties mobiliosioms programėlėms galioja tie patys informacinio ar kibernetinio saugumo principai, kaip ir kitai programinei įrangai (Yusop ir kt., 2016). Todėl pagrindiniai programėlės saugumo tikslai – konfidencialumas, prieinamumas, vientisumas (į šį terminą gali įeiti autentiškumas ir neatsisakomumas (*non-repudiation*) ir konfidencialumas) (Treacy, McCaffery, 2016). Taip pat kadangi programėlė veikia tinkle, svarbu užtikrinti ir tinklo apsaugą. Mobilioji aplikacija bus tiek saugi, kiek bus apsaugota kiekviena jos dedamoji (de Lima Fontao, 2015). Atskiriems aplikacijos elementams yra taikomi tam tikri saugumo reikalavimai, kai kurie net reglamentuoti teisės aktuose.

Pažymėtina, jog bet kuri Europos sąjungoje veikianti ir mobiliąją aplikaciją valdanti organizacija tampa ir skaitmeninių paslaugų teikėja, todėl jai galioja 2016 m. liepos 6 d. Europos Parlamento ir Tarybos direktyva (ES) 2016/1148 „Dėl priemonių aukštam bendram tinklų ir informacinių sistemų saugumo lygiui visoje Sąjungoje užtikrinti“ bei su ja susiję šalių, kuriose veiklą vykdo organizacija, įstatymai. Dėl to organizacijos informacinės sistemos, tarp jų ir mobiliosios aplikacijos bei ją aptarnaujantys duomenų centrai, turi užtikrinti:

1. Sistemų ir įrenginių saugumą;
2. Tinkamą incidentų valdymą;
3. Veiklos tęstinumo valdymą;
4. Stebėseną, auditą ir bandymus;
5. Sistemų atitiktį tarptautiniams standartams (Europos Parlamento ir Tarybos direktyva, (ES) 2016/1148)

Tai reiškia, jog organizacijos turi ne tik nuolat stebėti korektišką savo mobiliosios aplikacijos veikimą, bet ir testuoti visas savo sistemas bei užtikrinti jų saugumą.

Pamatinė kibernetinio saugumo taisyklė – atgrasymas, kitaip tariant, sistemos kūrimas taip, kad jos laužyti užpuolikams neapsimokėtų: būtų per brangu (pavyzdžiui, dėl naudojamos specialios

programinės įrangos) išgauta informacija neatneštų naudos (pavyzdžiui, dėl šifravimo) ir pan. (Kramer, Teplinsky, 2013). Tai galioja ir mobiliąsias aplikacijas kuriančioms ir valdančioms organizacijoms, taigi, mobiliųjų aplikacijų saugumas neatsiejamas nuo organizacijų saugumo lygio.

Viena dažniausiai taikomų informacinio ir kibernetinio saugumo metodologijų organizacijose – CIS (organiacijos „Interneto saugumo centras“ arba „*Center for Internet Security“*) kritiniai kibernetinio saugumo kontrolės vektoriai, dar žinomi kaip CIS20 arba CIS CSC (*Critical Security Controls*) (Woods ir kt., 2017).

CIS CSC sudaro 20 praktikų ir patarimų, kuriuos įgyvendinus organizacija tampa geriau pasiruošusi atgrasyti ar apsiginti nuo kibernetinių atakų. Kontrolės vektorius organizacija skirsto į tris grupes: pagrindinius, pamatinius ir organizacinius.

Pagrindinės:

1. Organizacijos įrenginių kontrolė ir inventorizacija;
2. Organizacijos programinės įrangos kontrolė ir organizacija;
3. Nuolatinis pažeidžiamumų valdymas (analizė ir šalinimas);
4. Kontroliuotas administratoriaus teisių naudojimas;
5. Saugi įrenginių, programinės įrangos, mobiliųjų įrenginių, nešiojamų kompiuterių, darbo vietų ir paslaugų konfigūracija;
6. *Logų* (istorinių sistemos įrašų) priežiūra, monitoringas ir auditas;

Pamatinės:

7. Elektroninio pašto sistemų ir interneto naršyklių apsauga;
8. Apsauga nuo kenkėjiškų programų (antivirusinių programų diegimas)
9. Tinklo prievadų, protokolų ir paslaugų prieigos apribojimas ir kontrolė;
10. Duomenų atkūrimo galimybių užtikrinimas;
11. Saugi tinklo prietaisų: ugniasienių, maršrutizatorių ir jungiklių konfigūracija;
12. Perimetro apsauga
13. Duomenų apsauga
14. „Būtina žinoti“ principo įvedimas
15. Bevielės prieigos kontrolė
16. Paskyrų monitoringas ir kontrolė

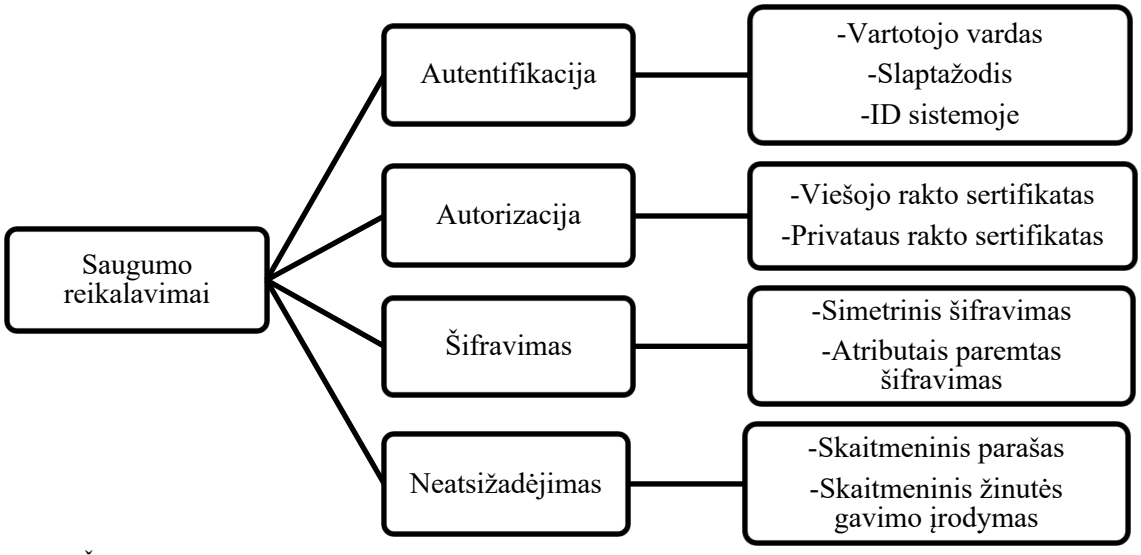
Organizacinės:

17. Saugumo mokymų programos įdiegimas organizacijoje
18. Aplikacijų
19. Incidentų ir atsako į juos valdymas
20. Įsiskverbimo testų bandymai, siekiant identifikuoti organizacijos saugumo spragas. (CIS CSC, žiūrėta 2021 vasarį).

Šių priemonių ir praktikų įgyvendinimas organizacijoje sukuria kontekstą ir prielaidą saugių produktų, taigi ir saugių mobiliųjų aplikacijų kūrimui.

1.3.2. Nefunkciniai reikalavimai

Dėl programėlės specifinės veiklos nišos didžiausias vartotojo dėmesys skiriamas jos funkcionalumui. Tačiau nemažiau svarbūs ir nefunkciniai reikalavimai, apibrėžiami kaip aukšto lygmens savybių rinkinys, parodantis programėlės kokybę (Jha, Mahmoud, 2019), t. y., jei programėlė šias savybes turi, ji mėgstama vartotojų ir todėl lengviau išsilaiko rinkoje. Nefunkciniais reikalavimais įprasta laikyti tokias programėlės charakteristikas kaip *usability*, arba tinkamumą/lengvumą naudoti, prisitaikymą prie įvairių įrenginių (*scalaiblity*), pajėgumas (*performance*), saugumas bei kitas (Jha, Mahmoud, 2019). Tačiau būtent saugumo nefunkcinis reikalavimas tiek mokslininkų (Yusop ir kt., 2016, Salini, Kanmani, 2016), tiek ekspertų (Guardsquare, 2021, Veracode, 2020) nuomone, yra svarbiausias.



Šaltinis: Yusop ir kt., 2016

6 pav. Saugumo pagrindiniai saugumo reikalavimai ir jų atributų pavyzdžiai

Pagrindiniai saugios aplikacijos reikalavimai – saugi autentifikacija, autorizacija, šifravimas ir neatsižadėjimo užtikrinimas (Yusop ir kt., 2016). Reikalavimai ir jiems priskiriami saugumo atributai pateikiami 5 paveiksle.

1. Autentifikacija. Autentifikacijos metu programėlė patvirtina vartotojo ar kitų programų, su kuriomis sąveikauja, tapatybę. Autentifikacijos atributai – vartotoją ar programą identifikuojantys duomenys: vartotojo vardas, ID sistemoje, slaptažodis ir pan.
2. Autorizacija. Patvirtinusi vartotojo ar programos tapatybę, mobilioji aplikacija viešojo arba privataus rakto pagalba patikrina, kokios teisės vartotojui priskirtos, ir priklausomai nuo to suteikia

prieigą prie savo resursų. Pavyzdžiui, elektroninės parduotuvės programėlės autentifikuotas vartotojas gali prieiti prie savo paskyros informacijos, pirkimų istorijos ir bendro parduotuvės produktų sąrašo, tačiau, su sąlyga kad autentifikacija ir autorizacija veiks korektiškai, jis nematys kitų vartotojų paskyrų informacijos ir pirkimų istorijos.

3. Šifravimas. Pagal nutylėjimą, visa programėlės komunikacija su serveriu turi būti šifruota. Tai vienas iš būdų užtikrinti programėlės informacijos vientisumą – jog tranzito tarp programėlės ir serverio metu ji nebuvo pakeista, išliko autentiška.
4. Neatsižadėjimas. Tai svarbus reikalavimas, nes jis taip pat padeda užtikrinti informacijos vientisumą ir autentiškumą. Neatsižadėjimo atributais laikomi skaitmeniniai parašai, dažniausiai su laiko žymomis, fiksuojantys tam tikrus vartotojo veiksmus (pavyzdžiui, mygtuko „sutinku“ paspaudimą po privatumo politika) ar žinutės gavimo įrodymus (pavyzdžiui, kai vartotojo atidaryta žinutė pažymima kaip perskaityta) (Yusop ir kt., 2016).

Minėti nefunkciniai saugumo reikalavimai yra tapę rinkos standartu (Balebako ir kt., 2014, Weir ir kt., 2020, Yusop ir kt., 2016), tačiau vėlgi, šie reikalavimai yra užduoties, siekiamybės lygmenyje, o ne konkrečių veiksmų, sprendimų sąrašas. Sprendimo saugumo reikalavimų užtikrinimui savo išgalėmis dažniausiai ieško jau patys mobiliųjų programėlių kūrėjai ar valdytojai (Balebako ir kt., 2014, Weir ir kt., 2020).

1.3.3. Bendrajame duomenų apsaugos reglamente apibrėžti reikalavimai

Tikėtina, jog programuotojų darbą iš dalies palengvino bendrasis duomenų reglamentas (toliau tekste – BDAR), Europos Parlamente priimtas 2016 metais, bei analogiškos paskirties dokumentai kitose pasaulio šalyse, pavyzdžiui, „Kalifornijos vartotojų privatumo aktas“ arba CCPA (*California Consumer Privacy Act*), 2021 sausio mėn. įsigaliojęs visoje JAV, Jungtinės karalystės „Duomenų apsaugos aktas“ arba DPA (*Data Protection Act*), įsigaliojęs 2018 m. ir įgavęs savarankiško teisės akto statusą Jungtinei Karalystei palikus Europos sąjungą (ES), Kinijoje žadamas priimti „Asmeninės informacijos apsaugos įstatymas“ arba PIPL (Personal Information Protection Law). Prie nefunkcinių reikalavimų, prijungus asmens duomenų apsaugos teisės aktus programuotojai gali geriau įsivertinti galimas saugumo spragų rizikas, taip pat yra aiškesni reikalavimai šifravimui, duomenų nuasmenimui ir pan. (Truong, 2016). Tačiau pažymėtina, jog nors šie dokumentai atlieka panašų ar net tapatų vaidmenį – reglamentuoja asmens duomenų rinkimo, tvarkymo ir saugojimo tvarką, jie turi ir nemažai skirtumų, todėl programuotojams, kuriantiems programėles kitos šalies rinkoms, svarbu susipažinti su toje šalyje galiojančiais asmens duomenų apsaugos įstatymais.

BDAR pripažįstamas kaip pirmasis ir didžiausią įtaką asmens duomenų apsaugai padaręs dokumentas, kurio pavyzdžiu buvo priimti kitų šalių teisės aktai (Goddard, 2017, Truong, 2016), dėl to

jis aktualus ne tik Lietuvos ar ES, bet ir platesniame kontekste. Pagal reglamentą sąvoka „asmens duomenys“ apima, bet neapsiriboja tokiais duomenis kaip asmens vardas, pavardė, asmens tapatybės kortelės ar paso numeris (Lietuvos atveju – ir asmens kodas), pajamos, kultūriniai ypatumai, interneto protokolo (IP) adresas, elektroninio pašto adresas, telefono numeris, sveikatos duomenys (BDAR, 2016). Taigi, programėlių, kurios atlieka autentifikaciją ar kitu būdu renka bet kuriuos iš paminėtų asmens duomenų, valdytojai tampa asmens duomenų tvarkytojais ir/arba valdytojais, kuriems galioja BDAR. Todėl kuriant saugias mobiliąsias aplikacijas būtina remtis pagrindiniais reglamento principais:

1. Teisėtumo, sąžiningumo ir skaidrumo. Asmens duomenys turi būti tvarkomi tik su asmens sutikimu, prieš tai jam pranešant kokie jo duomenys renkami, ir valdomi, kokiais tikslais tai daroma ir kur bei kiek laiko jie bus saugomi;
2. Tikslingumo. Duomenys turi būti renkami konkrečiam tikslui ar užduočiai ir nebetvarkomi, kai užduotis ar tikslas – įvykdyti ir nebeaktualūs;
3. Duomenų minimalumo. Turi būti renkama kaip įmanoma mažiau asmens duomenų, pirmiausia ieškant būdų, kaip tą patį funkcionalumą vartotojui pasiūlyti nerenkant asmens duomenų;
4. Duomenų tikslumo. Asmens duomenys turi būti tikslūs, jo prašymu – patikslinami ne ilgiau nei per 30 dienų;
5. Duomenų saugojimo apribojimai. Duomenys turi būti renkami, tvarkomi ir saugomi tik tokį laiką, koks yra būtinas tinkamam paslaugos teikimui, o pasibaigus paslaugos teikimo faktui ar sutarčiai, ištrinami arba nuasmeninami;
6. Duomenų vientisumas ir konfidencialumas. Tai – konkrečiai su informaciniu saugumu susijęs principas, kuris teigia, jog duomenys pasitelkiant visas organizacines ir teisines priemones turi būti tvarkomi ir saugomi taip, kad būtų apsaugoti nuo neteisėto tvarkymo ar perėmimo, praradimo, sunaikinimo ar sugadinimo (BDAR, 2016).

Kaip jau minėta, „App Store“ ir „Google Play“ reikalauja, jog visi šie principai būtų tiek įgyvendinti programėlės techniniais sprendimais, tiek išdėstomi privatumo politikoje, o vartotojų duomenys tvarkomi tik su nedviprasmiškai išreikštu vartotojo sutikimu. Mokslininkai N. Momen, L. Fritsch ir M. Hatamian (2019), atlikę išsamų tyrimą, konstatavo, jog bendra mobiliųjų aplikacijų saugumo situacija po BDAR įsigaliojimo yra šiek tiek pagerėjusi. Tačiau vis ryškėja tendencija, jog vartotojai privatumo politikų neskaito (arba beveik neskaito) ir stengiasi kuo greičiau su jomis sutikti tam, kad galėtų pradėti naudotis programėle ar kita paslauga. Šį reiškinį mokslininkai vadina „neinformuotu sutikimu“ (*uninformed consent*) (Nunan, Yenicioglu 2014, Schairer ir kt., 2018, Utz ir kt., 2019). Moksliniuose šaltiniuose taip pat teigiama, jog privatumo politikos turi būti supaprastinamos ir pateikiamos taip, kad jas būtų lengva skaityti (Utz ir kt., 2019) bei konstatuojama, kad „neinformuotas sutikimas“ nekuria ryšio tarp vartotojo ir paslaugos teikėjo (šiuo atveju mobiliosios programėlės kūrėjo

ar valdytojo) (Nunan, Yenicioglu, 2014), todėl organizacijos, kurioms rūpi jų vartotojai, turėtų savo privatumo politikai ir vartotojo duomenų apsaugai skirti daugiau dėmesio (Schairer ir kt., 2018).

Vienas svarbiausių BDAR tikslų – duomenų subjektų įgalinimas sekti kur, kaip ir kodėl naudojami jų duomenys (Europos Parlamento ir Tarybos direktyva, (ES) 2016/1148). Mokslininkės V. M. Wottrich, E. A. Van Reijmersdal ir E.G. Smit (2018) atliko tyrimą norėdamos išsiaiškinti, ar po BDAR vartotojai pradėjo daugiau dėmesio skirti savo asmens duomenų saugumui. Tyrimo rezultatai parodė, jog vartotojai gerai supranta savo asmens duomenų svarbą, supranta jų pažeidžiamumą, jaučia nerimą dėl privatumo, todėl yra linkę mažinti riziką renkantis programėles (pavyzdžiui, nenaudoti programėlės, jei ji iš pažiūros prašo per didelės prieigos prie mobiliojo įrenginio aparatinių funkcijų, atidžiau rinktis programėles parduotuvėse). Tačiau paradoksalu tai, kad jei vartotojams programėlė atrodo svarbi, ji atlieka jam svarbią funkciją, vartotojų dėmesys saugumui sumažėja arba vartotojai daro kompromisą ir aukoja savo privatumą dėl galimybės naudoti svarbią programėlę (Wottrich ir kt., 2018). Panašią tendenciją pastebėjo ir mokslininkai, tyrę socialinio tinklo „Facebook“ programėlės diegimo procesą ir su tuo susijusius vartotojų nuogąstavimus dėl saugumo. Tyrimo metu buvo pastebėta, jog vartotojai linkę pasitikėti programėlės kūrėjais, taip pat kad vartotojų elgesys saugumo klausimais tiesiogiai priklauso nuo to, ar programėlė jiems svarbi – t. y., kuo vartotojui programėlė svarbesnė, tuo jo privatumas tampa mažiau svarbus (Eling ir kt., 2013).

Apibendrinant pažymėtina, jog programėlės, nors yra laikomos savarankiškais programinės įrangos paketais, veikia tiek techninėje, tiek teisinėje aplinkose, dėl to jų saugumui aktualūs tiek programėlėse tvarkomų duomenų saugojimo ir kitos infrastruktūros, tiek saugaus veikimo sprendimai, taigi ir taikomi reikalavimai. Kadangi programėlėje dažnai tvarkomi asmens duomenys, jos saugumui aktualus ir BDAR.

1.4. Programėlių saugumo spragos ir jų identifikavimo analizė

1.4.1. Mobiliųjų aplikacijų spragos ir jų atsiradimo priežastys

Mobiliosiose aplikacijose parduotuvių, o kartu ir rinkos, globalumas didina konkurenciją ir tikimybę, kad panašią programėlę gali sugalvoti kas nors kitas. Tai tampa vienu iš svarbiausių faktorių, dėl ko skubama kuo greičiau aplikacijas patalpinti į parduotuves (Nigam ir kt., 2018). Nors kibernetinio saugumo padėtį stebintys asmenys pastaraisiais metais pastebi, jog programėlių kūrėjai skiria vis daugiau dėmesio saugumui bei testavimui (Veracode, 2020, S. Carielli ir kt., Forrester, 2020 ir kiti), pagrindinė tendencija išlieka tokia pati: didžiausias dėmesys skiriamas patrauklaus dizaino kūrimui, tinkamam aplikacijos funkcionalumui, kuo geresnės vartotojo patirties (angl. *user experience* arba UX) kūrimui, o saugumas lieka antrame plane ir dažnai saugumo spragos būna taisomos jau po to kai

programėlės būna patalpintos aplikacijų parduotuvėse (Yusop ir kt., 2016, Nigam ir kt., 2018). „Veracode“ savo ataskaitoje pastebi, jog net 70 procentų mobiliųjų programėlių, turinčių atviro kodo atributų ar įskiepių, iš jų paveldi po bent vieną saugumo spragą.

N. Yusop, M. Kamalrudin, M. M. Yusof ir S. Sidek (2016) atliko tyrimą, siekdami išsiaiškinti, kaip pradedantys mobiliųjų programėlių programuotojai geba iš aplikacijos funkcionalumo (pavyzdžiui, prisijungimas, paieška, įsigijimas) išskirti svarbiausius saugumo atributus (pavyzdžiui, vartotojo vardas, slaptažodis, sertifikatai ir pan.). Tyrimo rezultatai parodė, kad tik 4 procentai respondentų gali teisingai išskirti visus saugumo atributus (Yusop ir kt., 2016). Pagrindinė tokios situacijos priežastis ta, kad saugumo klausimu dauguma programuotojų yra savamoksliai – informacijos apie saugių programų kūrimą jie ieško internete arba klausia patarimo kolegų (Balebako ir kt., 2014). Taip pat pažymėtina, jog programuotojai apie kodo saugumą mokęsi iš mokslinės medžiagos demonstravo geresnius rodiklius nei jų kolegos, mokęsi iš internete rastų patarimų (Acar ir kt., 2016). C. Weir, B. Hermann ir S. Fahl (2020), siekdami išsiaiškinti, kodėl mobiliosiose aplikacijose atsiranda tiek daug programavimo klaidų ir su tuo susijusių saugumo spragų, apklausė 335 jau patyrusius ir sėkmingai savo srityje dirbančius programuotojus. Apklausos rezultatai parodė, kad tik mažiau nei ketvirtadalis programuotojų turi galimybę pasikonsultuoti su saugumo ekspertais, tik trečdalis reguliariai vykdo kodo testavimus, o programėlių pakeitimas ar tobulinimas įsigaliojus BDAR buvo daugiau kosmetiniai, nei esminiai (Weir ir kt., 2020). Šie tyrimai parodo, jog nors programuotojai supranta bendras dėl saugumo spragų tiek programėlės vartotojams, tiek jas valdančioms įmonėms galinčias kilti grėsmes (Weir ir kt., 2020, Acar ir kt., 2016, Weir ir kt., 2017), jiems trūksta specifinių kibernetinio saugumo žinių ir tinkamo saugumo eksperto palaikymo (Yusop ir kt., 2016, Balebako ir kt., 2014, Acar ir kt., 2016). Minėti mokslininkai vieningai tvirtina, jog šią situaciją išspręstų strateginis požiūris į kuriamų mobiliųjų aplikacijų saugumą jam teikiant didesnį prioritetą, į kūrimo procesą įtraukiant kibernetiniame saugume besispecializuojančius asmenis (Yusop ir kt., 2016, Weir ir kt., 2017) ir automatizuojant testavimo procesus (Acar ir kt., 2016, Weir ir kt., 2020, Weir ir kt., 2017).

Pagrindiniai programėlių saugumo trūkumai gali būti skirstomi į tris grupes: API spragos, SSL sertifikatų saugumo spragos ir privačių duomenų nutekėjimas (Weir ir kt., 2020, Veracode, 2020, NowSecure, 2019). Siekiant išsiaiškinti programišių atakos vektorius ir šių spragų keliamas grėsmes tikslinga kiekvieną šių grupių aptarti atskirai.

API spragos. Ankstesniame poskyryje trumpai aprašyta API sąsaja sulaukia itin daug ekspertų ir mokslininkų dėmesio. Ji dažnai identifikuojama kaip labiausiai pažeidžiama mobiliųjų programėlių dalis (Egele ir kt., 2014, Mutchler ir kt., 2015, Weir ir kt., 2020, Balebako ir kt., 2014, Ravitch ir kt., 2015). Tai taip pat ir bene dažniausiai programišių taikinyje atsirandanti mobiliųjų aplikacijų ekosistemos dalis (S. Carielli ir kt., Forrester, 2020). API programišių gali būti pasiekiamas trimis būdais:

1. Programinės įrangos, tinklų ir prietaisų saugumo spragos ar sutrikdymai. Mobiliojo telefono operacinės sistemos, ryšio sutrikdymai (pavyzdžiui, DDOS atakomis) gali suteikti programišiams prieigą prie tose sistemose ar tinkluose veikiančių programėlių API;
2. Įsiterpimas į API ir serverio komunikaciją. Vadinamos „*man in the middle*“ (pažodžiui – žmogaus viduryje, kitaip – įsiterpusio trečiojo asmens) atakos, kai programišius apsimesdamas *proksiu* (tinklo tarpininku) įsiterpia į API komunikaciją su serveriu užgrobdamas sesiją, perimdamas serverio siunčiamą ar gaunamą informaciją ar neteisingai sukonfigūruodamas saugumo nustatymus. Pažymėtina, jog programėlėse pasitelkiant trečiųjų šalių serverius ar kitas paslaugas, jų API būna sukurti iš anksto, todėl kai kurias saugumo spragas programėlė „paveldi“ ne dėl jos kūrėjų klaidų;
3. Išnaudojamos pačios programėlės funkcionalumo API spragos: įterpiamas kenksmingas kodas, pasiekiama nepakankamai užšifruota jautri informacija (pavyzdžiui, sesijos raktas) (Egele ir kt., 2013).

SSL sertifikatų saugumo spragos. SSL (dar vadinamas TSL) sertifikatas – elektroninis dokumentas, saugomas programėlės serveryje, užtikrinantis komunikacijos tarp vartotojo ir serverio šifravimą ir, tuo pačiu, konfidencialumą. Sertifikatas turi viešąjį raktą, patvirtinantį abiejų komunikacijos pusių tapatybę bei privatų raktą, kurio pagalba kliento serveriui ir atvirkštine kryptimi siunčiamos žinutės užšifruojamos ir iššifruojamos. SSL požymiai naršyklėje – užrakintos spynelės piktograma arba žodis „*secure*“ (saugu) prie internetinio puslapio adreso ir adreso pradžia „https“. Jei svetainėje SSL sertifikatas neįdiegtas, jos adresas prasidės raidėmis „http“, bus pavaizduota atrakinta spynelė arba bus parašyta „*not secure*“ (nesaugu). Į puslapius be SSL sertifikato nepatartina vesti jokios jautrios informacijos, nes ji bus nešifruojama ir ją gali matyti tretieji asmenys (Clark ir kt., 2003). Svarbus SSL sertifikatą naudojančių programų komponentas – nepatikimų, galimai sufabrikuotų SSL sertifikatų atpažinimas ir pranešimas apie juos vartotojui. Ši funkcija plačiai naudojama naršyklėse, tačiau jos pritrūksta kai kurios mobiliosiose aplikacijose (Hubbard ir kt., 2014). Neturėdamos kaip pranešti vartotojui apie įtartiną SSL sertifikatą daugelis programėlių sesiją nutraukia, tačiau kai kurios programėlės priima bet kurį joms pateiktą sertifikatą, tokiu atveju minėtam į komunikaciją įsterpusiam programišiui atskleisdamos konfidencialią sesijos informaciją (Hubbard ir kt., 2014).

Privačių duomenų nutekėjimas. Vartotojų privatūs duomenys gali būti nutekinami ir be *„man in the middle“* įsiterpimo – dėl programuotojų klaidų ar netinkamo duomenų saugojimo. 2015 metais kilo didelis sąmyšis, kai paaiškėjo, jog daugelio programėlių duomenų kopijos patalpintos viešai prieinamose talpyklose (FRPT- Software Snapshot. 6/21/2015). Žinant talpyklos adresą suinteresuotam asmeniui pakaktų išgauti ar atspėti slaptažodį ir visas talpyklos turinys patektų į jo rankas. Analogiška situacija įvyko 2021 m. vasarį Lietuvoje, kai viename internetiniame forume buvo paskelbti netinkamai saugotų „CityBee“ duomenų kopijų įrašų rinkiniai. Pastaruoju atveju paaiškėjo, jog duomenys buvo ne

tik patalpinti viešai prieinamame serveryje, bet dar ir nešifruoti, arba šifruoti labai minimaliai. Tarp nutekėjusių duomenų – vartotojų vardai, pavardės, asmens kodai, silpnai koduoti slaptažodžiai, vairavimo teisių duomenys ir kita jautri informacija. Netrukus po šio incidento pasirodė ir daugiau nutekintų duomenų iš lažybų paslaugas teikiančios įmonės „Orakulas“, pažinčių svetainės „Darni pora“ ir kitų.

Be minėtų trijų pažeidžiamumų grupių mokslinėje bei ekspertinėje literatūroje dažnai minimas ir OWASP dažniausių mobiliųjų saugumo spragų sąrašas, kurio pagalba programuotojai gali atkreipti dėmesį į kritines programėlių ir mobiliųjų įrenginių vietas ir joms skirti ypatingą dėmesį (Weir ir kt., 2020; Salini, Kanmani, 2016, Veracode, 2020, NowSecure, 2019). OWASP arba *Open Web Application Security Project* (Atviro tinklo programų saugumo projektas) – nepelno siekianti organizacija, šviečianti programų kūrėjus ir valdytojus, kad jų programos būtų kuo saugesnės vartotojams. Mobiliųjų rizikų sąrašas susideda iš 10 punktų, apimančių tiek mobiliojo prietaiso programinę įrangą, tiek pačios programėlės programavimo bei duomenų valdymo ir saugojimo aspektus:

1. Netinkamas mobiliojo įrenginio operacinės sistemos funkcijų ir saugos valdiklių nustatymų naudojimas, pavyzdžiui, prastai suprogramuota biometrinė autentifikacija (piršto antspaudo ar veido atpažinimo), pernelyg platūs ir nepakankamai reglamentuoti leidimai prieiti prie mobiliojo prietaiso funkcijų bei duomenų (kameros, mikrofono, nuotraukų galerijos, vietos sekimo ir pan.);
2. Nesaugios duomenų saugyklos mobiliajame telefone, kurias gali perskaityti kitos į mobiliųjį įrenginį įdiegtos aplikacijos ar įrenginį perėmęs asmuo;
3. Nepakankamai apsaugoti „keliaujantys duomenys“ (angl. *data in transit*), kitaip – komunikacija tarp kliento (šiuo atveju – mobiliajame įrenginyje įdiegtos programėlės) ir serverio, iš kurio aplikacija pasiima duomenis;
4. Nesaugi autentifikacija. Dėl mobiliųjų aplikacijų patogumo vartotojui, programėlės vartotojo tapatybės patvirtinimui dažnai naudoja ne ilgus slaptažodžius, o trumpus pin kodus ar biometrinius duomenis (piršto antspaudą, veido nuskaitymą), autentifikacijos saugumą tiesiogiai surišant su mobiliojo įrenginio saugumu. Tačiau nesaugiai sukurtos autentifikacijos sesijos gali būti perimtos;
5. Nepakankamas jautrių duomenų šifravimas;
6. Perteklinė autorizacija, kai programėlė prašo prieigos prie jos funkcijai nereikalingų vartotojo duomenų ar mobiliojo įrenginio aparatinių funkcijų, pavyzdžiui mikrofono, kontaktų sąrašo, lokacijos sekimo ir pan.;
7. Prasta programėlės kodo kokybė. Nepakankamai ištestuotos, kodo klaidų turinčios mobiliosios programėlės kelia daug rizikų tiek vartotojams, tiek ir programėlės savininkui bei jo IT infrastruktūrai;

8. Programėlės kodo gadinimas (angl. „*tamper*“) jį modifikuojant, manipuliuojant skirtingais programėlės veikimo scenarijais ar pasitelkiant kenkėjiškas programas;
9. Programėlės atkūrimas atvirkštinės inžinerijos būdu, siekiant atrasti ir išnaudoti jos silpnąsias vietas ar nukopijuoti funkcionalumą bei kitus sprendimus;
10. Perteklinis funkcionalumas. Programėlėje turi veikti tik tos funkcijos ir turi būti pasiekiama tik tie duomenys, kurie yra būtini korektiškam aplikacijos veikimui, kitaip tariant, programėlė turi veikti mažiausios galimos privilegijos principu. Šios funkcijos ar duomenys yra kurie dažnai nematomi vartotojui, bet gali būti nesunkiai perimti (OWASP Mobile Top 10, žiūrėta 2021 m. vasarį).

Tačiau tiek duomenų nutekėjimo, tiek kitų spragų atsiradimo galima išvengti tinkamai jas aptikus ir laiku pašalinus.

1.5. Mobiliųjų aplikacijų saugumo spragų aptikimas ir šalinimas

Tiek kibernetinio saugumo ekspertų publikacijose, tiek moksliniuose šaltiniuose daugiausiai pateikiami du pasiūlymai, kaip aptikti saugumo spragas bei apskritai pagerinti aplikacijų saugumą: rizikų valdymas ir valdymas bei programėlių testavimas (Weir ir kt., 2020, Carielli ete al, Forrester, 2020, Veracode, 2020, NowSecure, 2019). Pastarasis savo ruožtu pagal charakteristikas gali būti skirstomas į statinį ir dinaminį; automatizuotą ir atliekamą rankiniu būdu (Weir ir kt., 2017).

Testavimas

Mokslinėje literatūroje dažnai pabrėžiama **statinio** mobiliųjų programėlės testavimo (Static Aplication Security Testing), arba SAST nauda (Weir ir kt., 2020, Li ir kt., 2017 ir kiti). Tai – programėlės kodo testavimas, jis dažniausiai naudojamas iš karto baigus kurti programėlę, patikrintas, ar jos kodas neturi klaidų atrandant kodo klaidas. Šio testavimo metu atrastas klaidas lengva ir nebrangu ištaisyti. Tačiau statinis testavimo metodas netestuoja jau veikiančių aplikacijų, o kai kurie trūkumai ne visada atsispindi programėlės kode (Veracode, 2020). Dėl to ekspertai pataria kartu su SAST naudoti ir **dinaminį** testavimą (Dynamic Aplication Security Testing), arba DAST. DAST proceso metu testuojama veikianti programėlė, prie jos kodo neprieinama. Šį metodą rekomenduojama taikyti jau prieš pat programėlę patalpinant į aplikacijų parduotuvę, bet taip pat ir reguliariai po paleidimo, siekiant laiku pastebėti galimai nepastebėtas ar dėl naujinių atsiradusias saugumo spragas. Kadangi šis metodas taikomas vėlesnėse aplikacijos gyvavimo ciklo fazėse, jų aptiktų saugumo trūkumų šalinimas – brangesnis, tačiau kibernetinio saugumo ekspertai rekomenduoja SAST ir DAST naudoti kartu (Veracode, 2020).

Tiek SAST, tiek DAST gali būti tiek automatizuoti, t. y., kai testavimą atlieka speciali programinė įranga ar specialiai parašyti kodai, vadinami testais (Carielli ete al, Forrester, 2020), tiek rankiniai:

programėlės kodo testavimas kodo peržiūros (arba *code review*) metodu, kai vienas programuotojas peržiūri kito programuotojo parašytą kodą (SAST); bei veikiančios mobiliosios programėlės tikrinimas kai ją testuoja patys programuotojai arba specialiai šiam tikslui suburta kokybės užtikrinimo (*Quality Assurance* arba QA) komanda ar dedikuotas žmogus (DAST) (Weir ir kt., 2017).

Rizikų valdymas

Nacionalinis standartų ir technologijos institutas (*National Institute of Standards and Technology* arba NIST) rizikų valdymą apibrėžia kaip procesą, kurio metu „nustatoma rizika, įvertinama rizika ir imamasi priemonių rizikai sumažinti iki priimtino lygio“ (Wang JA ir kt., 2009). Teoretikai akcentuoja rizikų vertinimo svarbą, teigdami, jog apskaičiavusios galimas rizikas organizacijos yra linkusios priimti teisingesnius sprendimus (Hopkin, 2010), tarp jų – ir dėl saugumo. Rizikų valdymo metodologiją patartina taikyti kiekvieną kartą, kai organizacijoje planuojami didesnio masto pokyčiai ar pradedama tvarkyti naujus asmens duomenis (Hopkin, 2010). Kadangi mobilioji aplikacija dažnai yra susijusi su asmens duomenimis, be to, ji, kaip jau minėta, yra organizacijos ekosistemos dalis, prieš ją kuriant ar prieš atliekant reikšmingesnius atnaujinimus taip reikėtų atlikti rizikos valdymo procesą (Salini, Kanmani, 2016). Rizikos valdymo metodologijos dažiausiai yra panašios, tačiau jau minėta organizacija OWASP pateikia būtent programinės įrangos rizikai sukurtą vertinimo metodologiją. Ši metodologija skirta padėti programuotojams įsivertinti kiekvienam programėlės elementui kylančias grėsmes bei numatyti jų šalinimo būdus (OWASP informacija. Žiūrėta 2021 m. vasarį). **Rizikos lygis** (R) pagal šią metodologiją apskaičiuojamas dauginant tam tikros saugumo spragos įšnaudojimo **tikimybę** (T) ir **žalos**, kurią incidentas padarytų (Ž), mastą (R=T*Ž). Metodologija susideda iš penkių žingsnių: 1) rizikos identifikavimo; 2) įsilaužimo tikimybės nustatymo, konkrečiais įverčiais įvertinant tiek galimus veikėjus, norinčius įsilaužti į sistemas, ir jų gebėjimus bei galimybes tai padaryti, tiek saugumo spragas, kurios gali būti išnaudotos; 3) galimos žalos – tiek techninės, tiek verslo/reputacinės, įvertinimo; 4) rizikos lygio apskaičiavimo dauginant 2 bei 3 žingsnyje priskirtus tikimybės ir žalos įverčius bei 5) atrinkimo, kuri saugumo spraga bus taisoma pirmiausia (OWASP metodologija, žiūrėta 2021m. vasarį). Pažymėtina, tinkamam rizikų valdymui svarbi vertintojo kompetencija, turimos žinios šioje srityje (Hanson, 2014), todėl ankstesniuose skyriuose apžvelgtos problematikos kontekste galima teigti, jog programuotojai visapusiško rizikų valdymo proceso atlikti nebūtų pajėgūs. Tai dar kartą patvirtina, jog programuotojams būtinas saugumo ekspertų palaikymas o tuo pačiu tai, kad mobiliųjų programėlių kūrėjams ir valdytojams jų programėlių saugumas turi būti vienu iš prioritetų, nes tokio eksperto įdarbinimas turi įtakos ir finansinių resursų paskirstymui.

Dialektika

Kadangi bendras mobiliųjų aplikacijų saugumo standartas nėra sukurtas, dauguma programėles kuriančių ar valdančių įmonių turi tam tikrus sąrašus, gerųjų praktikų aprašus – į ką turi būti atkreiptas dėmesys išleidžiant naują programėlę ar jos atnaujinimą, kokios dažniausia pasitaikančios klaidos ir kaip

jų išvengti (Weir ir kt., 2017, Yusop ir kt., 2016, Acar ir kt., 2016). Tačiau kai kurie mokslininkai ir ekspertai laikosi nuomonės, kad tokie sąrašai visgi nėra tokie efektyvūs, kaip būtų galima tikėtis (Weir ir kt., 2017, Guardsquare, 2020). Kaip to sprendimą C. Weir, A. Rashid ir D. Noble (2017) siūlo dialektikos – polemizuojančio tipo dialogo su kolegomis taikymą. Mokslininkų siūlomos metodologijos pagrindu skirtinguose programėlės kūrimo etapuose galimas problemas išryškina kitas programėlę kuriančio asmens komandos (ar kitos komandos tos pačios įmonės viduje) narys. Metodologija susideda iš penkių dialektikos technikų:

1. Minčių lietaus (*brainstorm*) sesija, kurioje siekiama išgryninti visų galimų sistemos užpuolikų profilius: jų gebėjimus, motyvaciją, kokiomis spragomis jie greičiausiai pasinaudotų. Šiame etape turėtų dalyvauti produkto (mobiliosios aplikacijos ar tam tikro jos atnaujinimo/ pakeitimo) kūrėjų komanda (programuotojai, testuotojai, analitikai), pageidautina kuo skirtingesnių požiūrių ir kvalifikacijų.
2. Derybos dėl saugumo sprendimų. Šiame etape minčių lietaus sesijoje identifikuotos rizikos iškomunikuojamos vadovybei (ar klientams), ieškoma kompromisų tarp verslo tikslų pasiekimo ir saugumo kriterijų, siekiama vadovybei/klientams paaiškinti, kodėl programėlės saugumas turi būti laikomas prioritetiniu.
3. Diskusija apie produkto saugumą su kita komanda organizacijos viduje programavimo metu arba iškart jam pasibaigus. Šio etapo metu pasitelkiama prie kito projekto ar produkto dirbanti komanda, prašant jų įvertinti ar pakomentuoti diegiamus saugumo sprendimus. Šis etapas gali būti atliktas neformaliai, jo sėkmė priklauso nuo tinkamo mikroklimato organizacijos viduje, kai žmonės nebijo atvirai, bet draugiškai ir konstruktyviai išsakyti kritiką savo kolegoms. Šis etapas svarbus ir dėl to, kad vidinės ar tarpkomandinės komunikacijos metu išryškėja atsakomybės, mat dažnu atveju yra tikimasi, jog dėmesį į tam tikrą problemą atkreips kas nors kitas, t. y., yra neaiški atsakomybė dėl skirtingų produkto elementų kokybės.
4. Saugumo komandos įsitraukimas. Prieš paleidžiant produktą vartotojams svarbu įsitikinti, jog jis saugus. Todėl baigus programėlės kūrimo darbus ji yra peržiūrima specialiai suburtos kokybės užtikrinimo (dar vadinama QA, *quality assurance*) komandos ar dedikuoto asmens bei kitų programuotojų. Tai atliekama tiek veikiančios programėlės (dinaminiu) testavimu, tiek statiniu, kodo peržiūros (angl. *code review*) metodu. Siekiant maksimalaus rezultato į šį etapą, kaip ir į pirmąjį, svarbu įtraukti žmones su skirtingomis žiniomis ir kompetencijomis.
5. Automatizuotas testavimas. Tai – ketvirtojo etapo papildymas ar net pakeitimas. Mažesnės organizacijos, startuoliai ar savarankiškai dirbantys programėlių kūrėjai dažnai neturi pakankamai finansinių išteklių dedikuotų QA žmonių samdymui. Didelės organizacijos taip pat nėra linkusios švaistyti žmogiškųjų išteklių testavimui, kurį galima atlikti automatizuoto

testavimo pagalba. Svarbu, kad būtų pasitelkiami tiek statinis, tiek dinaminis automatizuoti testavimo būdai.

6. Reagavimas į atsiliepimus. Dėl programėlių specifikos jų kūrėjai ne visada mato, jei ji veikia nekorektiškai. Dėl to itin svarbu gauti grįžtamąjį ryšį iš klientų, registruoti bet kokius funkcinius ar saugumo kriterijų neatitikimus pirminiam planui bei leisti juos taisančius atnaujinimus (Weir ir kt., 2017).

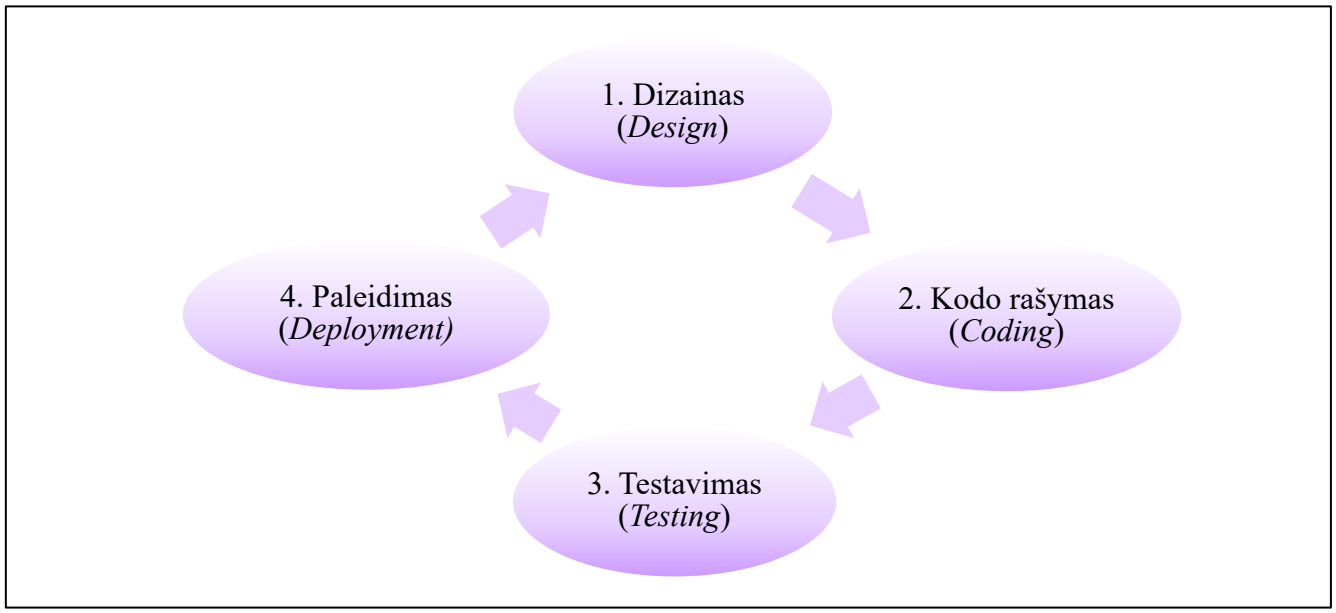
Dialektikos taikymo metodologija apima tiek aukščiau aprašytus testavimo metodus, tiek neformalų rizikų vertinimą, kai rizikos nėra matuojamos konkrečiais įverčiais, bet į jas atkreipiamas dėmesys, jas stengiamasi iškomunikuoti vadovybei ar klientams. Tikėtina, jog kai kurie dialektikos metodologijos etapai, greičiausiai – automatizuotas ar rankinis testavimas ir pasitarimai su kolegomis yra taikomi daugumoje įmonių – tokią išvadą galima daryti apibendrinus mokslinius darbus, kuriuose buvo tiriamos programėlių kūrėjų saugumo žinios bei kokiais būdais jie jas gilina (Balebako, 2014, Acar ir kt., 2016, Weir ir kt., 2020). Tačiau lygiai taip pat mokslininkai pastebi, jog dėmesys saugumui programėlių kūrimo procese atsiranda gerokai per vėlai. Nuosekliai taikant Dialektikos metodologiją tikėtina, to būtų išvengta, nes saugumo klausimai šios metodologijos dėka yra keliami pačioje proceso pradžioje.

1.6. Mobiliųjų aplikacijų kūrimo modelių analizė

1.6.1. Saugios programinės įrangos kūrimo ir palaikymo modelis

Kadangi viena iš darbo tikslo dalių yra sukurti unifikuotą į saugumą orientuoto programėlių kūrimo ir valdymo modelį, svarbu apžvelgti, kokie mobiliųjų aplikacijų kūrimo ir valdymo modeliai jau yra sukurti, kuo jie skiriasi, kokių privalumų ir trūkumų turi. Kadangi saugumas darbe suprantamas ne kaip tam tikri konkretūs veiksmai, o procesas, tikslinga koncentruotis į procesinius programėlių kūrimo modelius.

Kaip jau apibrėžta 1.1. poskyryje, mobiliosios programėlės – viena iš programinės įrangos rūšių, todėl žiūrint abstrakčiai, joms taikomi panašūs reikalavimai, jų kūrimo ir palaikymo žingsniai iš esmės vienodi (Pandey ir kt., 2018). Supaprastintas saugios programinės įrangos kūrimo ir palaikymo modelis vertintinas kaip ciklas, nes yra nesibaigiantis, produktas (programinės įrangos paketas) nuolat tobulinamas (Salini, Kanmani, 2016). Jis susideda iš keturių pagrindinių fazių (žr. 7 paveikslą): 1) dizaino, 2) kodo rašymo, 3) testavimo ir 4) paleidimo, po kurio vyksta atnaujinimų ir tobulinimų kūrimas, kodo rašymas ir t. t. (Salini, Kanmani, 2016).



Šaltinis: Salini, Kanmani, 2016

7 pav. Saugios programinės įrangos kūrimo ir palaikymo ciklas

Tačiau daugelis mokslininkų visgi laikosi nuomonės, jog mobiliųjų programėlių kūrimui negali būti taikomi tie patys modeliai, kaip ir kompiuteriams skirtoms aplikacijoms. Anot jų, mobiliosios programėlės keičiasi itin dažnai, dėl to ir jų kūrimo modeliai turi būti adaptyvūs, lankstūs ir lengvai bei greitai pritaikomi (Pandey ir kt., 2018; Joorabchi ir kt.., 2013; Flora ir kt., 2014 ir kiti). Mokslininkai Mamta Pandey, Ratnešas Litorija, Pratikas Pandey, apibendrinę moksliniuose darbuose pateikiamus skirtumus tarp mobiliųjų bei kompiuteriams skirtų aplikacijų, pateikė tokį jų sąrašą:

1. Mobiliųjų aplikacijų kūrimui reikia platesnes kompetencijas turinčių žmonių komandos, nei kompiuteriams skirtų programų;
2. Nuolat besikeičiantys reikalavimai. Mobiliųjų aplikacijų rinka (tiek vartotojo, tiek techninėje/technologinėje plotmėse), o kartu ir aplikacijoms keliami reikalavimai keičiasi dažniau nei kompiuteriams skirtoms aplikacijoms;
3. Ryšys tarp verslo logikos architektūros ir techninio sprendimo mobiliosiose programėlėse yra daug glaudesnis nei kompiuteriams skirtose programose;
4. Mobiliųjų aplikacijų vartotojai yra daugiau diversifikuoti, naudoja skirtingus prietaisus, todėl mobiliųjų programėlių vartotojo sąsajos dizaino sprendimai turi būti daug lankstesni ir daug geriau apgalvoti nei kompiuteriams skirtų programų vartotojo sąsaja.
5. Mobiliųjų aplikacijų kūrimo ir valdymo ciklas – daug trumpesnis nei kompiuteriams skirtų programų;
6. Viena iš svarbiausių mobiliųjų aplikacijų savybių – greitaveika, nes vartotojams greitis yra svarbu, jie linkę prie aplikacijų praleisti sąlyginai nedaug laiko (kartais tik apie 20 sekundžių).

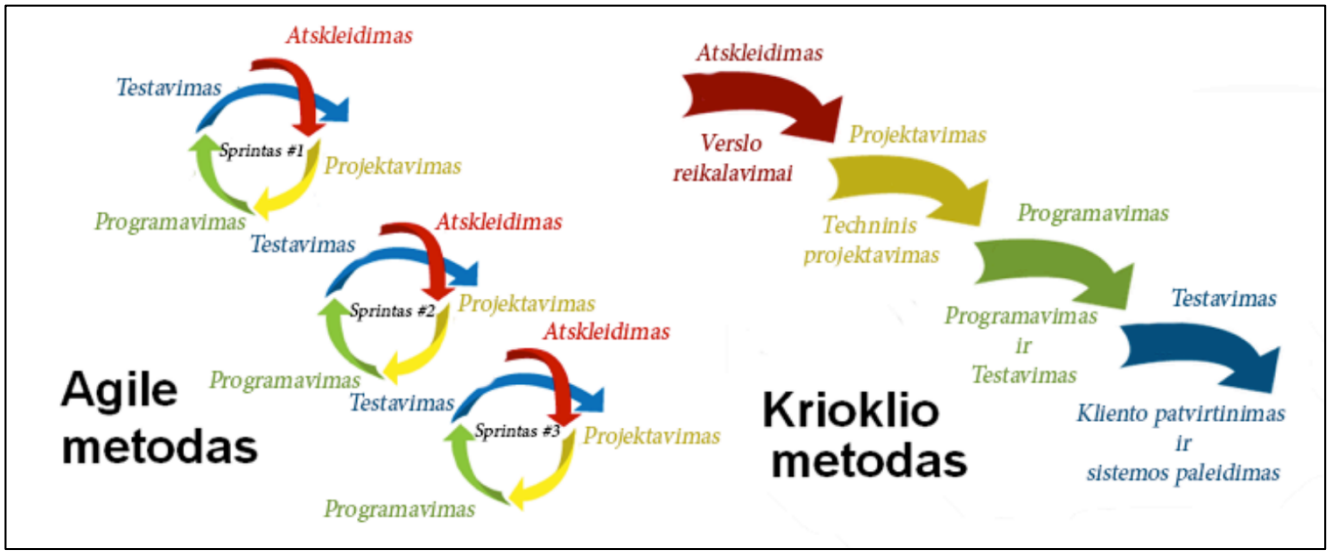
Tuo tarpu kompiuteriams skirtoms programoms tai yra mažiau aktualu, nes vartotojai linkę su jomis dirbti ar kitaip sąveikauti ilgiau;

7. Skirtinga prietaisų architektūra ir pajėgumai. Kompiuteriai turi gerokai daugiau vidinės atminties, todėl jų programos gali būti galingos, saugoti didelius kiekius informacijos. Mobiliosios programėlės turi būti gerokai „lengvesnės“ dėl mobiliųjų įrenginių specifikos;
8. Ekrano dydis. Mobilieji įrenginiai turi mažesnius ekranus, todėl programėlių kūrėjams reikia strategiškai apgalvoti visus vartotojo patirties (*user experience* arba UX) aspektus. Tai nėra taip aktualu kompiuteriams skirtoms programoms;
9. Mobiliojo prietaiso energijos ištekliai ir sąnaudos. Mobiliosios programėlės, greitai iškraunančios mobilųjį prietaisą, bus blogai įvertintos vartotojų ir galiausiai nebe bus parsisiunčiamos;
10. Programėlės kūrimo biudžetas dažnai yra gerokai mažesnis nei kompiuteriams skirtos programos (Pandey ir kt., 2018).

Taigi, lyginant su kompiuteriams skirtomis programomis, mobiliosios programėlių kūrimo ir palaikymo procesas yra daug dinamiškesnis. Dėl šios priežasties daugelis mokslininkų akcentuoja „Agile“ metodologijos tinkamumą mobiliųjų programėlių kūrimui (Edison ir kt., 2018; Pandey ir kt., 2019; Flora ir kt., 2014).

1.6.2. „Agile“ ir „Krioklio“ modeliai

„Agile“ (angl. judrus, lankstus, paslankus) modelis, sukurtas 2000 m. M. Fowler, D. Highsmith, J. Kern, J. Sutherland ir K. Schwaber, dažnai pateikiamas kaip priešprieša „Waterfall“ arba „Krioklio“ modeliui, kurį 1970 metais pristatė W. W. Rois (Almasri, 2016).



Šaltinis: Stalnionytė, 2013, parengta pagal Reich, 2012

8 pav. „Krioklio“ ir „Agile“ metodologijos ir jų skirtumai

Tiek „Agile“, tiek „Krioklio“ metodai susideda iš tokių pačių žingsnių: reikalavimų atskleidimo, dizaino projektavimo, programavimo, testavimo ir testavimo (Reich, 2012). Kituose šaltiniuose prieš reikalavimų surinkimą dar minimas iniciacijos, o po testavimo – paleidimo žingsniai (Van Cesteren, 2017, Flora ir kt., 2014). Tačiau nors žingsniai arba fazės abiejose metodologijos tapačios, jų proceso eiga skiriasi. „Krioklio“ metodologija pasižymi linijiškumu, t. y., kita fazė prasideda prieš tai buvusiai fazei pasibaigus, fazės nepersidengia (Almasri, 2016; Pandey ir kt., 2018). Tuo tarpu „Agile“ metodologija daugiausia dėmesio skiria darbuotojų sąveikai, komunikacijai bei veiklos optimizavimui (Cohen ir kt., 2003), yra pritaikyta trumpiems ciklams ir orientuojasi į funkcionalumų planavimą bei prioritetų eilės sudarymą (Highsmith, Cockburn, 2001; Van Cesteren, 2017). Susistemintas „Krioklio“ ir „Agile“ modelių tinkamumas priklausomai nuo organizacijos, kliento norų ir vykdomo projekto pobūdžio pateikiamas 3 lentelėje:

3 lentelė. Susisteminti „Krioklio“ ir „Agile“ metodologijų skirtumai

Daugiau tinkamas „Krioklio“ metodas	Daugiau tinkamas „Agile“ metodas
Formali, hierarchinė organizacijos struktūra, informacijos organizacijoje kryptis iš viršaus į apačią	Neformali organizacinė kultūra, informacijos organizacijoje kryptis abipusė
Klientas (tiek organizacijos viduje, pavyzdžiui, vadovybė, tiek išorinis) produktą gauna programinės įrangos kūrimo proceso pabaigoje	Klientas dalyvauja produkto kūrimo procese
Kliento atsakomoji reakcija (grįžtamasis ryšys, atsiliepimai) gaunama pasibaigus produkto kūrimo procesui.	Kliento atsakomoji reakcija (grįžtamasis ryšys, atsiliepimai) gaunama produkto kūrimo metu, dažniausiai kas mėnesį
Produktas paleidžiamas visų produkto kūrimo fazių pabaigoje	Produktas paleidžiamas dažnai (vidiniai paleidimai), dažnai kas mėnesį
Skirtingi padaliniai dažniausiai nekooperuoja	Skirtingi padaliniai kooperuoja siekdami bendro tikslo
Komandų pasiekti rezultatai pristatomi retai	Darbuotojai jaučia spaudimą pristatyti projekto progresą kas tam tikrą laikotarpį (kas savaitę, kas dvi ir pan.)

Šaltinis: Dima, Maassen, 2018

Svarbu pažymėti, jog „Agile“ metodologija taip pat gali skirstoma į skirtingas metodikas: „Extreme programming“ (XP) (Ekstremalus programavimas), „Scrum“, „Crystal“, „Adaptive Software

development“ (ASD) (Adaptyvus programinės įrangos kūrimas), „Lean Software Development“ („Lieknas“ programinės įrangos kūrimas), „Feature Driven Development“ (FDD) (Funkcionalumais paremtas kūrimas) ir kiti. (Abrahamsson ir kt., 2019). Šios metodikos skiriasi vartotojo įsitraukimo į produkto kūrimą lygiu (pavyzdžiui, „XP“ – reikalauja aukšto vartotojų įsitraukimo, o, „Scrum“ vartotoją atstoja produkto vadovas komandos viduje), komandų narių tarpusavio bendravimo pobūdžiu (pavyzdžiui, „XP“ ir „Scrum“ metodikoms būdingi neformalūs kasdieniai susirinkimai, o FDD ir DSDM bendraujama informacijos ir dokumentų dalijimosi lygmenyje), projektų dydžiu (XP ir ASD labiau tinkami mažiems projektams), iteracijos ciklo dydžiu.

1.6.3. „Scrum“ modelis

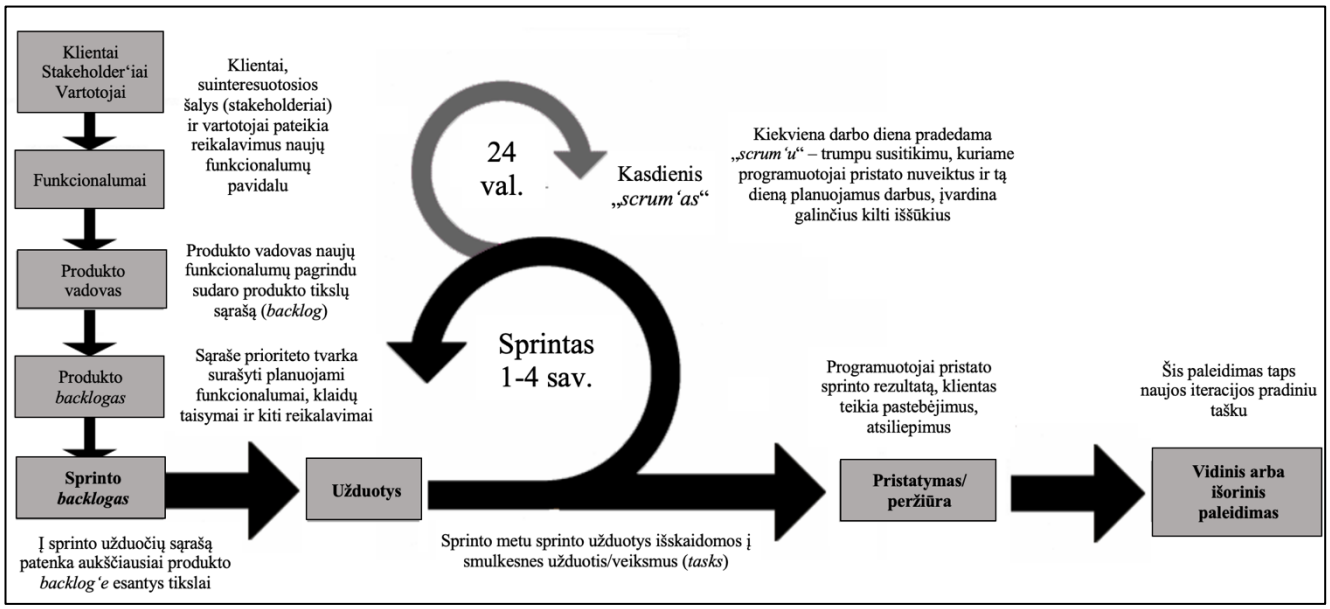
Harlyn Flora, Svati V. Čande, ir Šiaofeng Vangas atliko tyrimą, siekdami išsiaiškinti, kurią iš šių metodikų dažniau linkę naudoti mobiliųjų programėlių kūrėjai ir valdytojai. Dauguma iš 130 tyrime dalyvavusių respondentų teigė, jog naudoja „Scrum“ metodiką (Flora ir kt., 2014). Šio tyrimo rezultatus patvirtina ir „State of Agile“ raportai, kuriuose Scrum metodika jau kelerius metus iš eilės neužleidžia pozicijų (The 14th annual State of Agile Report, 2020).

„Scrum“ metodiką 1995 metais sukūrė Džefas Suterlandas (Jeff Sutherland) ir Kenas Švaberis (Ken Schwaber) (vėliau su dar trimis kolegomis pristatę ir „Agile“ metodologiją). Metodikai būdinga 1) tikslą suskaidyti į užduotis (programėlės ar jos naujinių kūrimo darbus arba *tasks*) ir jas vykdyti prioriteto tvarka (pagal prieš tai sudarytą prioritetinę eilę); 2) atnaujinimus kurti ir paleisti ciklišk ai, vadinamais sprintais arba iteracijomis[5], kurių trukmė 1-4 savaitės (kuriant naują programėlę ciklai taip pat išlieka, tačiau paleidimas iki galutinio programėlės varianto būna ne išorinis, į programėlių parduotuves, o vidinis); bei 3) kasdien rengti trumpus komandų susirinkimus (Van Cesteren, 2017). Scrum modelis paprastai turi keturis komponentus:

1. Produkto neatliktų darbų sąrašas (*Product backlog*):
    a. Klientai, vartotojai, programėlę kuriančios ir/ar palaikančios organizacijos vadovybė ir kitos suinteresuotos šalys sudaro programėlės funkcinius reikalavimus (kokias funkcijas programėlė turi turėti, kokią paslaugą ir kaip turi atlikti. kuriame prioriteto tvarka surašyti produkto kūrimo ar tobulinimo tikslai;
    b. Produkto vadovas paverčia reikalavimus tikslais ir juos surašo į produkto tikslų sąrašą;
    c. Sąrašas sudėliojamas pagal prioritetinę eilę. Jame surašyti visi funkcionalumai, kurie turi būti įgyvendinti, klaidų taisymai, reikalavimai;

[5] Iteracija pagal lietuvių kalbos žodyną – „procedūros kartojimas, remiantis ankstesnės procedūros rezultatu“ (https://www.lietuviuzodynas.lt/terminai/Iteracija).

2. Sprinto/iteracijos užduočių sąrašas (*sprint backlog*):
    a. Į iteraciją patenka aukščiausiai neatliktų darbų sąrašo prioritetinėje eilėje esantys tikslai;
    b. Tikslai suskaidomi į užduotis ir padalijami komandoms;
3. Vykdoma iteracija:
    a. Atliekamos užduotys;
    b. Iteracijos metu kasdien vyksta susitikimai, kuriuose aptariami nuveikti ir būsimi darbai, galintys iškilti iššūkiai;
    c. Sprinto rezultatai pristatomi suinteresuotoms šalims, gaunamas grįžtamasis ryšys;
4. Darbinis programinės įrangos prieaugis:
    a. Jei iteracijos rezultatas atitinka nustatytus reikalavimus, jos užduotis pažymima kaip atlikta;
    b. Įvyksta išorinis (į aplikacijų parduotuves) arba vidinis paleidimas (įmonės viduje, pristatymas kitoms komandoms);
    c. Šis rezultatas tampa kitos iteracijos pradiniu tašku;
    d. Kai pasiekiamas nepadarytų darbų sąrašo (*Project backlog*) tikslas (*Definition of Done*), vertinama, jog įvyko darbinis programinės įrangos prieaugis (Van Cesteren, 2017).



Šaltinis: Van Cesteren, 2017

9 pav. „Scrum" modelis

Scrum modelis mėgstamas dėl komandiškumo ir darbo matomumo – dažnai pristatinėjant progresą, jis labiau matomas, negu kai pristatomas tik galutinis rezultatas. Kita vertus cikliški sprintai ar iteracijos gali ne visiems pasiteisinti, todėl šio modelio pritaikomumas nėra absoliutus (Van Cesteren, 2017).

1.6.4. Binsaleh ir Hassan modelis

Mokslininkai H. Edison, R. Jabangwea ir A. Nguyen-Duc (2018) atliko sisteminę mobiliųjų aplikacijų kūrimo modelių analizę. Savo tyrime jie matavo modelių pritaikomumą, tinkamumą veiklos šakai, pasitelkdami *rigor/relevance* principą. Iš dvidešimties pirminę analizės fazę įveikusių ir vėliau tirtų modelių tyrimo kriterijus atitiko tik du (Edison ir kt., 2018). Pirmasis, M. Binsaleh ir S. Hassan (2011) modelis buvo sukurtas apjungiant „Agile“ metodologiją, XP, Scrum ir FDD modelius. Tai – mobiliųjų aplikacijų kūrimo modelis, pritaikytą e-parduotuvėms bei panašaus funkcionalumo mobiliosioms programėlėms (Edison ir kt., 2018). Modelis paremtas daugybinėmis iteracijomis, kurios savo ruožtu sudarytos iš 5 fazių:

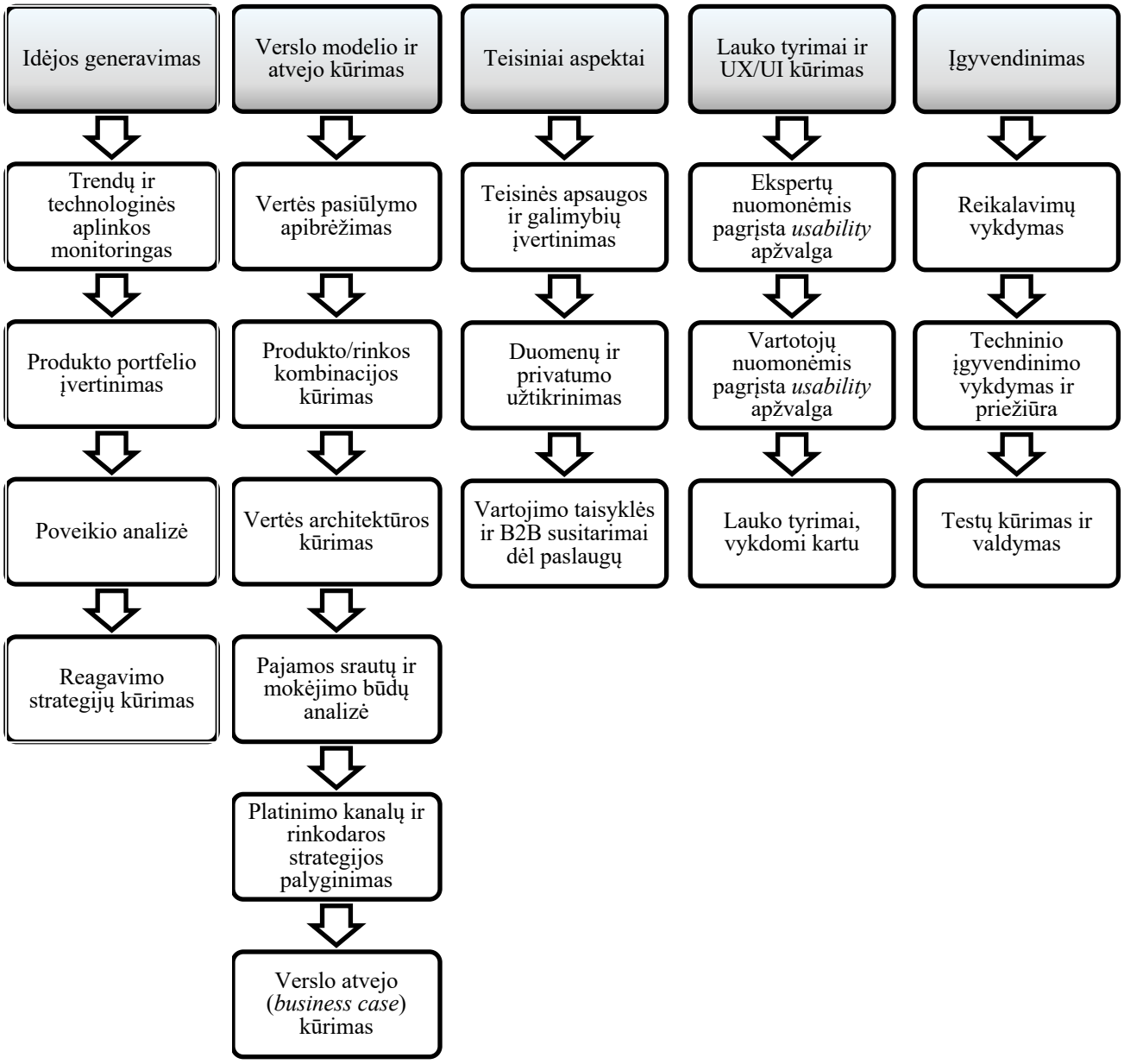
1. Tyrinėjimas
   - Verslo reikalavimų nustatymas
   - Sistemos metaforos nustatymas (sistemos metafora „Agile“ kontekste – terminų suvienodinimas, susitarimai dėl reikšmių)
2. Produkto planavimas
   - Programavimo užduočių pritaikymas prie programėlės vartotojo istorijos (vartotojo istorija „Agile“ kontekste – neformalus, apibendrintas paaiškinimas, kaip veikia viena ar kita funkcija; mažiausias darbo vienetas (užduotys dalinamos į vartotojo istorijas)).
   - Apibrėžti, kurios vartotojo istorijos bus įgyvendinamos pirmiausia ir kada bus paleidžiamos.
3. Iteracija
   - Iteracijos planavimas
   - Iteracijos vykdymas (programavimas)
4. Produkto dalies paleidimas
   - Testavimas: ar produktas atitinka reikalavimus?
   - Paleidimas
5. Palaikymas
   - Palaikymas ir evoliucija
   - Naujų produkto dalių paleidimo planavimas (Binsaleh, Hassan, 2011).

Pastebėtina, jog šis modelis fokusuojasi į *user experience* (vartotojo patirtį) ir *usability* (patogumą naudoti), dar vadinamus UX/UI. Modelis platus, tačiau kiek per abstraktus ir ne per daugiausiai nutolęs nuo „Agility“ metodologijos, todėl kvestionuotinas jo naujumas ir išskirtinumas bei pritaikomumas. Modelyje paliekama daug laisvės jį taikančiai organizacijai ar asmeniui, tačiau tai tuo pačiu vertintina

ir kaip silpnybė – jis nepakankamai detalus, dėl to neatneša daug vertės organizacijai, kuri nėra pažengusi mobiliųjų programėlių kūrime.

1.6.5. Zelder, Kittl ir Petrovic modelis

Antrasis H. Edisono ir kitų kartu tyrimą vykdžiusių mokslininkų reikalavimus atitikęs modelis – C. Zeidler, C. Kittl, ir O. Petrovic (2008) sukurtas mobiliųjų aplikacijų kūrimo ir palaikymo modelis (9 paveikslas). Modelio autoriai, vėlgi, siūlo iteracinį modelį, taigi daugiau priskirtiną prie „Agile“ metodologijos. Jo tikslas – sukurti produktą, didinantį vertę tiek vartotojui, tiek jį sukūrusiai organizacijai (Zeidler, 2008).



Šaltinis: Edison ir kt., 2018

10 pav. Mobiliųjų aplikacijų kūrimo modelis pagal Zeidler ir kt., 2008

Modelį sudaro 5 fazės: 1) Idėjos generavimo, kurioje atliekami rinkos tyrimai, produkto ir jo poveikio analizė bei kuriamos reagavimo į poveikį strategijos; 2) Verslo modelio ir atvejo kūrimo, kurioje išgryninama produkto vertė, kitoniškumas, pajamų srautai, marketingo strategijos; 3) Teisinių klausimų sprendimų, kurioje apsprendžiami teisiniai santykiai su partneriais ir vartotojais bei teisinio saugumo klausimai; 4) Lauko tyrimai ir *Usability* ir vartotojo patirties *User experience* (UX/UI) kūrimo, kurios metu produkto prototipas išbandomas tikslinėse grupėse ir koreguojamas siekiant užtikrinti kuo geresnę vartotojo patirtį; bei 5) įgyvendinimą, kurioje vyksta techniniai produkto kūrimo ir testavimo darbai (Zeidler, 2008).

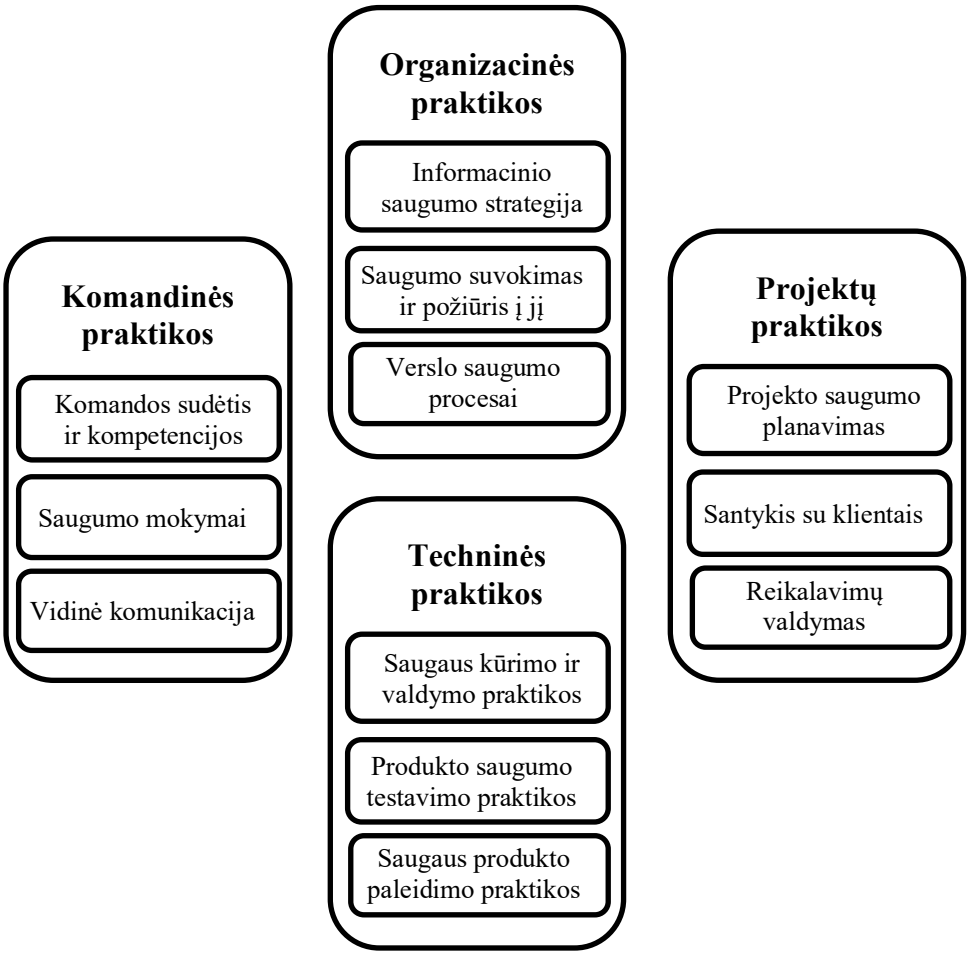
Modelis itin detalus analizės ir planavimo fazėse, kas aktualu organizacijoms, kurios neturi daug patirties mobiliųjų programėlių kūrime ir palaikyme. Tačiau produkto vykdymui ir testavimui modelyje nėra skiriama pakankamai dėmesio, o produkto paleidimas ir palaikymas išvis nėra minimi. Todėl vertintina, kad modeliui trūksta išbaigtumo, cikliškumo, kas, anot mokslininkų (Almasri, 2016; Pandey ir kt., 2018; Cohen, 2014 ir kitų) yra būtina nuolat besikeičiančioms programėlėms.

1.6.6. Newton, Anslow ir Drechsler kritiniai sėkmės faktoriai

Nors minėtieji modeliai ir metodologijos, tokios kaip Scrum ar XP, gali būti lanksčiai pritaikyti skirtingų organizacijų, pažymėtina, jog nei vienas jų nėra aiškiai orientuotas į saugumą. Tačiau N. Newton, C. Anslow ir A. Drechsler (2019) išskyrė Kritinius sėkmės faktorius (KSF), kurie padeda organizacijoms tiek pačioms būti atsparesnėms kibernetinių rizikų atžvilgiu, tiek kurti saugesnius produktus ir tokiu būdu gali būti vertintini kaip programėlių kūrimo ir valdymo papildymas saugumo klausimais. KSF mokslininkai suskirstė į keturias grupes pagal organizacijų sritis, lygius, kuriuose faktoriai veikia: 1) organizacines praktikas, apimančias organizacijos informacinio saugumo strategiją, saugumo suvokimą ir požiūrį į jį, bei verslo saugumo procesus; 2) komandines praktikas, į kurias sudaro dėmesys komandos sudėčiai ir kompetencijoms, saugumo mokymams bei veiksmingai vidinei komunikacijai; 3) projektų praktikas, apimančias projekto saugumo planavimą, santykį su klientais ir projekto reikalavimų valdymą; bei 4) technines praktikas, akcentuojančias saugų produktų kūrimą ir valdymą, testavimą bei paleidimą (Newton ir kt., 2019).

Ši metodologija darbo kontekste aktuali, nes apibrėžia faktorius, kurie svarbūs tam, kad „Agile“ metodologija besivadovaujančios įmonės dėl didelės produkto dinamikos neprarastų fokuso į saugumą. Modelio esmė – užtikrinti saugumą per organizacinę kultūrą, kurioje saugumas suvokiamas kaip strategijos dalis, didelis dėmesys skiriamas komandos narių kompetencijos (taip pat ir saugumo klausimais) ir dalijimasis žiniomis, projektai kuriami saugūs *by design* (pagal dizainą, pagal nutylėjimą, nuo pradžių saugūs), vadovaujamasi gerosiomis praktikomis. Taigi, ši metodologija nepaaiškina, kaip

sukurti saugią mobiliąją programėlę, bet sukuria saugumo kontekstą organizacinėje aplinkoje, kas yra svarbu kuriant saugius produktus, o šiuo atveju – saugias mobiliąsias aplikacijas.



Šaltinis: Newton ir kt., 2019

11 pav. Kritiniai organizacijų, veikiančių pagal „Agile“ metodologiją, saugumo faktoriai

Skyriaus apibendrinimas

Stengiantis perprasti visus saugių mobiliųjų aplikacijų kūrimo ir valdymo procesuose kylančius iššūkius buvo apžvelgta mobiliųjų aplikacijų specifika, programėlėms keliami reikalavimai bei dažniausiai pasitaikančios aplikacijų saugumo spragos. Paaiškėjo, jog mobiliųjų aplikacijų parduotuvės nėra sukūrusios saugumo standartų, kuriuos turi atitikti mobiliosios aplikacijos, tik pateikia rekomendacijas, todėl visa atsakomybė dėl programėlių saugumo sprendimo radimo ir įgyvendinimo tenka programuotojams. Pastarieji, nors ir supranta aplikacijų saugumo svarbą, dažniausiai saugumo srityje yra savamoksliai, jiems trūksta kibernetinio saugumo žinių. Dėl to dažniausiai saugumo spragų atsiradimą sąlygoja programavimo klaidos, kitaip tariant, jos atsiranda dėl programuotojų padarytų klaidų. Šias spragas atrasti ir įsivertinti padeda automatizuotas ir rankinis testavimas ir rizikų vertinimas, taip pat dialektikos taikymas.

Kadangi darbo tikslas – pasiūlyti į saugumą orientuotą mobiliųjų aplikacijų kūrimo ir valdymo modelį, buvo apžvelgti iki šiol sukurti mobiliųjų aplikacijų modeliai. Paaiškėjo, jog dažniausiai mobiliųjų aplikacijų kūrimui taikoma „Agile“ metodologija bei skirtingi šios metodologijos atmainos, o populiariausia jų – „Scrum“ metodologija. Remiantis šiomis teorinėje dalyje padarytomis išvadomis toliau darbe bus konstruojamas tyrimas bei į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis.

2. MOBILIŲ APLIKACIJŲ SAUGUMO PROBLEMATIKOS KŪRĖJAMS IR VALDYTOJAMS VERTINIMAS

2.1. Tyrimo metodologija

Analizuojant mokslinę literatūrą buvo išskirtos dažniausiai pasitaikančios ir didžiausią grėsmę keliančios mobiliųjų aplikacijų saugumo spragos, taip pat išnagrinėti mobiliosioms programėlėms taikomi saugumo standartai, taikomi reikalavimai. Pažymėtina, jog nors tiek mobiliųjų programėlių parduotuvės, tiek skirtingi teisės aktai pateikia abstrakčius techninius bei vartotojų duomenų apsaugos reikalavimus, sprendimus kaip šiuos reikalavimus įgyvendinti turi atrasti patys programėlės kūrėjai. Taip pat pastebėtina, jog dalis problemų yra „paveldimos“ kartu su trečiųjų šalių kurtais įskiepiais ar valdomomis paslaugomis. Nagrinėjant mokslinę literatūrą taip pat buvo prieita prie išvados, jog programuotojams dažnai trūksta kibernetinio saugumo žinių, kas trukdo jiems visapusiškai įvertinti jų kuriamų mobiliųjų programėlių aspektus.

Tyrimo metodika. Siekiant įvertinti saugių mobiliųjų aplikacijų kūrimo ir valdymo problematiką buvo atliekamas tyrimas, kurio metu bus apklausiami didelę patirtį sukaupę mobiliųjų programėlių kūrėjai. Šio tyrimo metu gauti duomenys bus sujungiami su teorine darbo medžiaga bei jų pagalba bus kuriamas į saugumą orientuotas mobiliųjų aplikacijų kūrimo modelis. Kadangi tyrimo metu siekiama giliau ištirti saugių mobiliųjų programėlių kūrimo iššūkius, t. y., teoriją siekiama sugeneruoti iš tyrimo metu gautų duomenų, tikslinga pasitelkti kokybinį tyrimo metodą (Gaižauskaitė, Valavičienė, 2016). Siekiant iš kokybinio tyrimo rezultatų daryti apibendrinančias išvadas svarbu, kad surinktą medžiagą būtų galima susisteminti ir apdoroti (Gaižauskaitė, Valavičienė, 2016), tačiau siekiant sumažinti galimą tyrėjo ribotumą, klausimai turi būti pakankamai atviri, tyrimas turėtų vykti kuo labiau organiškai, turi būti atkreipiamas dėmesys į kontekstą (Žydžiūnaitė, Sabaliauskas, 2017). Dėl šių priežasčių pasirinktas tyrimo instrumentas – struktūruotas interviu, kuris vėliau bus vertinamas per interpretatyvinę poziciją.

Tyrimo tikslas: įvertinti mobilių aplikacijų saugumo problematiką, kylančią programėlių kūrėjams ir valdytojams.

Tyrimo uždaviniai:

1. Apžvelgti mobiliųjų aplikacijų kūrimo eigą, procesus;
2. Atskleisti iššūkius, kylančius mobiliųjų aplikacijų kūrėjams, kuriant saugias programėles;
3. Įvertinti kibernetinio saugumo žinių ir saugumo metodų taikymo svarbą mobiliųjų programėlių saugumui;
4. Remiantis ekspertų patirtimi išskirti svarbiausius aspektus, kurie turėtų būti įtraukti kuriant ir verifikuojant į saugumą orientuotą mobiliųjų aplikacijų kūrimo ir valdymo modelį.

Tyrimo atranka ir patikimumas. Remiantis sociologinių tyrimų metodologija, kokybinio tyrimo patikimumui gal būti taikomi trys kriterijai: tikrumas/tikėtumas (angl. *credibility*), perkeliamumas (angl. *transferability*) bei įtikinamumas (angl. *trustworthiness*) (Golafshani, 2003, Žydžiūnaitė, Sabaliauskas, 2017). Siekiant atitikti **tikrumo/tikėtumo** kriterijų būtina pasirinkti tinkamus ekspertus: svarbu vadovautis nuostata, jog tyrimui aktualūs asmenys turi gerai išmanyti situaciją, turėti su ja susijusios patirties ir gebėti atsakyti į interviu metu jam užduodamus klausimus (Žydžiūnaitė, Sabaliauskas, 2017). Laikantis šio principo tam, kad būtų galima giliau ištirti mobiliųjų programėlių kūrėjams ir valdytojams kylančius iššūkius, buvo pasitelkta **kriterinė** atranka: tyrimo ekspertais buvo pasirinkti su mobiliosiomis programėlėms dirbantys asmenys, turintys ne mažesnę nei 5 metų patirtį būtent mobiliųjų programėlių kūrimo srityje. Tyrimo **perkeliamumo** kriterijų tyrimas taip pat potencialiai atitinka: tai, kad apklausiami ekspertai – Lietuvoje kuriamų programėlių atstovai, perkeliamumui neigiamos įtakos neturės, nes mobiliųjų aplikacijų rinka yra globali. Kitaip tariant, mobiliosios programėlės, taip pat ir kurtos Lietuvoje, yra daugiausiai platinamos dviejose visame pasaulyje prieinamose virtualiose parduotuvėse, o visoms šiose parduotuvėse platinamoms aplikacijoms yra taikomi vienodi reikalavimai nepaisant jų kūrimo geografijos. Todėl galima teigti, jog visos parduotuvėse patalpintos programėlės yra lygiavertės rinkos dalyvės. Pažymėtina ir tai, kad Lietuva plačiai pripažįstama kaip gana aukštu informacinu raštingumu pasižyminti šalis, todėl labai tikėtina, jog ir mobiliųjų aplikacijų srityje Lietuvos kūrėjai ir valdytojai atitinka Vakarų valstybių bei globalią rinką. Siekiant atitikti **įtikinamumo** kriterijų svarbu pasirinkti tinkamą tyrimo imtį. Remiantis klasikine testų teorija darytina išvada, kad kokybinio tyrimo ekspertų skaičių ir tyrimo rezultatų patikimumą sieja greitai gęstantis netiesinis ryšys ir grupėje nuo 5 iki 9 respondentų atsakymo patikimumas pasiekia maksimumą (Libby, Blashfield, 1978; Baležentis, Žalimaitė, 2011). Siekiant maksimalaus tyrimo patikimumo ir įtikinamumo, tyrimui pasirinkti 9 ekspertai.

Mobiliąsias aplikacijas kuriančioms bei valdančioms įmonėms buvo išsiųsti kvietimai dalyvauti tyrime (pabrėžiant pagrindinį ekspertų kriterijų, t. y., ne mažesnė nei 5 metų specifinė darbo su mobiliosiomis programėlėmis patirtis), iš 15 sutiko dalyvauti 3. Vėliau sniego gniūžtės principu, pagal ekspertų rekomendacijas buvo atrinkti kiti 10 ekspertų, iš kurių dalyvauti tyrime sutiko 6.

Kokybinio tyrimo patikimumas taip pat gali būti sustiprintas ir trianguliacijos principu (Patton, 2002, Žydžiūnaitė, Sabaliauskas, 2017). Trianguliacijos principas tyrimo imtyje atsiskleidžia trimis aspektais: 1) skirtingo lygmens pareigybių, darant prielaidą, jog skirtingo lygmens pareigybes užimantys asmenys procesus gali vertinti skirtingai (didesnę svarbą skirti skirtingiems proceso etapams), taip siekiant kuo plačiau atskleisti programėlių kūrimo procesą; 2) įmonės dydžio, darant prielaidą, jog nuo įmonės dydžio kinta ir komandų, kuriančių programėles, sudėtis bei aplikacijų kūrimo proceso trukmė; bei 3) specializacijos, darant prielaidą, jog skirtingos specializacijos ekspertai atkreips dėmesį į skirtingus, taigi vertinant bendrai – platesnio spektro programėlių saugumo iššūkius.

Tyrimo ekspertai: siekiant kuo didesnio tyrimo patikimumo, buvo siekiama atrinkti kuo sėkmingesnių mobiliųjų aplikacijų kūrėjus ir valdytojus. Taip pat buvo kreipiamas dėmesys į mobiliųjų aplikacijų teikiamų paslaugų pobūdį, stengiantis rinktis skirtingus mobiliųjų programėlių kūrėjus ir valdytojus. Be organizacijų ar asmenų, kurių darbo pobūdis – kurti ir valdyti daug mobiliųjų aplikacijų, ekspertai atstovauja sveikatos apsaugos, švietimo, verslo administravimo, žiniasklaidos ir mados veiklos sektorius.

Ekspertas Nr. 1 – UAB „Visma Lietuva“ programuotojas, dirbantis su „Android“ platformai pritaikyta Norvegijos darželių aplikacija. Aplikacija plataus funkcionalumo, apima vaikų ugdymo, lankomumo informacijos valdymą, turi tėvų ir mokytojų modulius. Pažymėtina, jog nors UAB „Visma Lietuva“ teikia gana platų internetinių sprendimų spektrą ir turi daug skirtingų klientų, dėl įmonės struktūros, t. y., suskirstymo dedikuotomis komandomis, vienu metu dirbančiomis su vienu projektu, šiuo atveju šis ekspertas priskiriamas prie tų, kurie kuria ir palaiko vieną projektą, o ne dirba su daug projektų vienu metu.

Ekspertas Nr. 2 – UAB „Mediapark“ asocijuotasis partneris, atliekantis vyresniojo projektų vadovo funkcijas. Šio eksperto komanda išleidžia po naują programėlę vidutiniškai kas 6-12 mėnesių. Tarp „Mediapark“ kurtų programėlių – tokios kaip „Spark“, „Viada“, „Redbull“ mobiliosios aplikacijos.

Ekspertas Nr. 3 – IBM dukterinės įmonės IBM iX mobiliųjų aplikacijų architektas. IBM iX Lietuvos padalinys per metus išleidžia 11-12 programėlių. Tarp IBM iX Lietuva kurtų mobiliųjų aplikacijų – mobiliosios programėlės tokioms kompanijoms kaip „Porche“ ir „BMW“.

Ekspertas Nr. 4 – UAB „Delfi“ IT skyriaus vadovas (CTO), atsakingas už organizacijos programėlės *backendą*, užduočių trečiajai šaliai, teikiančiai organizacijai hibridinės programėlės sprendimą, kūrimą (verslo poreikio iškomunikavimą bei techninių reikalavimų sudarymą).

Ekspertas Nr. 5 – startuolio UAB „Gigsis“, kuriam priklauso programėlė „Pingin“, direktorius. „Pingin“ – B2B (*business to business* arba verslas verslui) principu veikianti programėlė, siūlanti plataus spektro išmaniųjų namų ir prietaisų sistemas biurams, bendradarbystės erdvėms (angl. *hub*), viešbučiams ir pan.

Ekspertas Nr. 6 – laisvai samdomas „Android“ platformoms skirtų programėlių kūrėjas, besispecializuojantis *frontend* programavime ir naują programėlę išleidžiantis vidutiniškai kas 6-12 mėnesių.

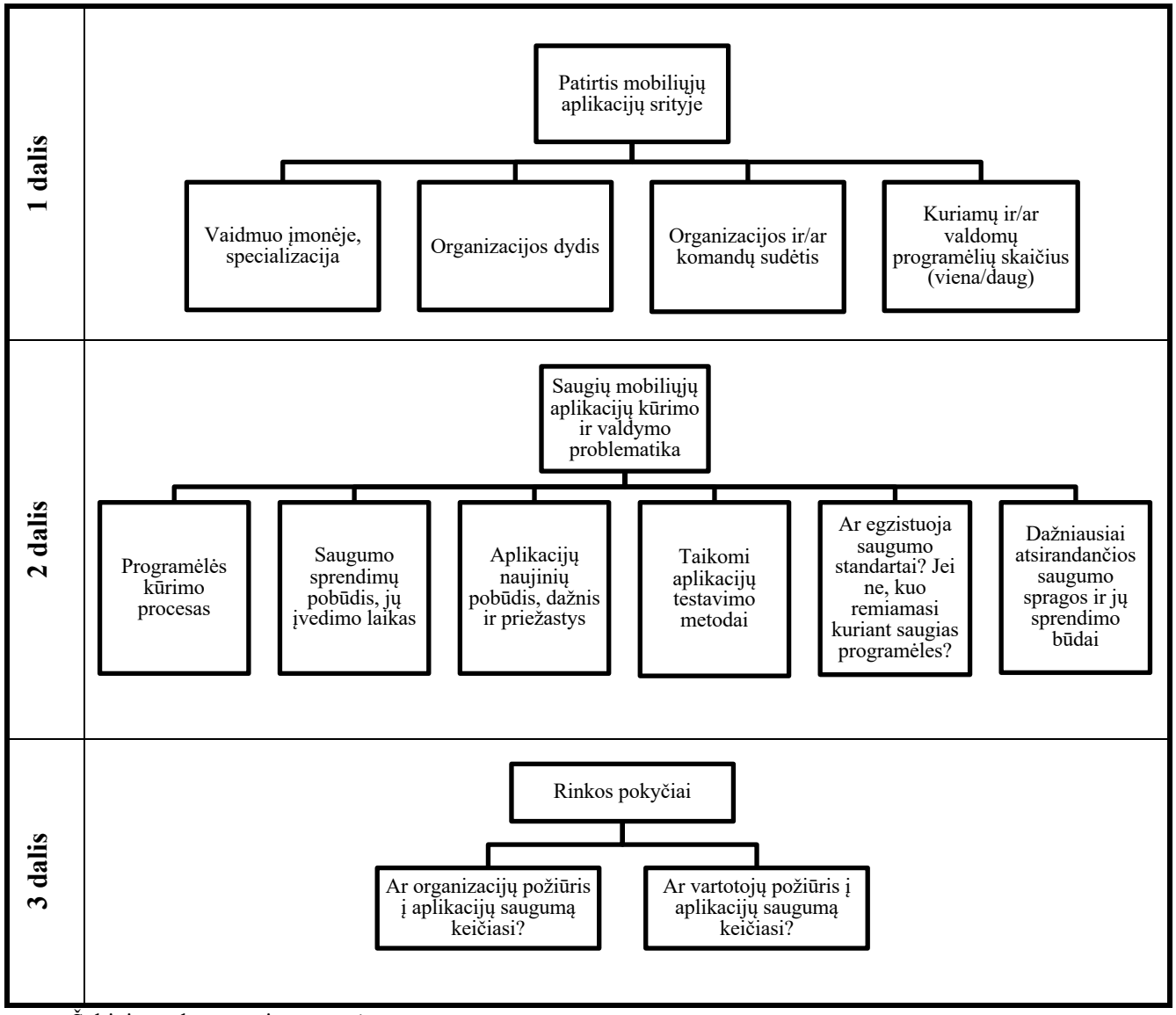
Ekspertas Nr. 7 – programėlės „Trafi“ programuotojas, besispecializuojantis *backend* programavime.

Ekspertas Nr. 8 – startuolio-programėlės „Mindletic“ IT skyriaus vadovas (CTO), atsakingas už korektišką programėlės veikimą. „Mindletic“ – emocinės higienos ir psichologinės sveikatos gerinimo įrankis.

Ekspertas Nr. 9 – UAB „Vinted“ Saugumo direktorius, vadovaujantis informacinio saugumo komandai ir prisidedantis prie visų įmonės produktų kūrimo.

Struktūruoto interviu modeliavimas

Informantams buvo pateikta 20 atvirų klausimų, siekiant išgauti kuo autentiškesnę respondentų nuomonę bei aprėpti kuo daugiau konteksto. Struktūruoto interviu klausimus tikslinga skirstyti į tris dalis: 1) klausimai, atskleidžiantys respondento patirtį ir vaidmenį mobiliųjų aplikacijų kūrime; 2) pagrindiniai klausimai, padėsiantys giliau ištirti saugių mobiliųjų aplikacijų kūrimo ir valdymo procesus, juose iškylančius iššūkius bei jų įveikimo būdus; ir 3) papildomi klausimai, atskleidžiantys ekspertų požiūrį į mobiliųjų aplikacijų rinkos pokyčius. Tyrimo klausimai pateikiami 1 priede, o schematiškai pavaizduoti 12 paveiksle:



Šaltinis: sudaryta tyrimo autorės

12 pav. Struktūruoto interviu klausimų loginė schema

Tyrimo etika

Struktūruotas interviu buvo atliekamas moksliniais tikslais, magistro baigiamojo darbo apimtyje. Tyrimo dalyviai jame dalyvavo laisva valia, neatlygintinai. Dėl pandeminės situacijos šalyje, visi interviu vyko nuotoliniu būdu, naudojantis programa „Zoom“. Kadangi visi informantai atstovauja verslo organizacijas, o kai kurie net gali būti traktuojami kaip netiesioginiai konkurentai, tyrime itin svarbu laikytis **konfidencialumo principo.** Todėl analizuojant tyrimo rezultatus informantų atsakymai pateikiami apibendrinta forma, o cituojant – nuasmeninami. Klausimus iliustruojančiose suvestinėse informantui priskirtas pavadinimas (pavyzdžiui, Ekspertas Nr. 1) nėra rodomas, o **informantai keičiami vietomis** (išskyrus pirmos dalies klausimus, kurie ir už tyrimo ribų yra laisvai prieinami), kad būtų išlaikomas anonimiškumas (pavyzdžiui, viename paveiksle pirmo informanto atsakymai gali būti vaizduojami pirmame stulpelyje, kitame paveiksle – septintame ir pan.). Interviu buvo įrašinėjamas programoje „Zoom“ įdiegtu įrankiu bei tyrėjo telefonu, siekiant sumažinti duomenų praradimo riziką. Visi šie principai buvo pristatyti kiekvienam tyrimo dalyviui interviu pradžioje, taip pat buvo teiraujamasi, ar tyrimo dalyvis leidžia minėti jo darbovietę ir pareigas. 8 atvejais iš 9 šis leidimas buvo duotas, kitu atveju bet kokia ekspertą identifikuoti galinti informacija nebuvo atskleista. Tyrimo rezultatai buvo apdorojami programa „MaxQDA“. Tyrimo iliustracijoms naudojamos ištraukos iš programos pagalba atvaizduotų kokybinio tyrimo rezultatų suvestinių.

2.2. Tyrimo duomenų analizė

Pirmojoje klausimų grupėje informantams buvo pateikiami klausimai apie jų darbo patirtį, rolę kuriant ar palaikant mobiliąsias aplikacijas, patirtį šioje srityje, komandos sudėtį. Šiais klausimais, kaip jau minėta, buvo siekiama kategorizuoti informantus pagal jų pareigybės lygį, įmonės dydį, specializaciją, veiklos pobūdį (ar ekspertas dirba su viena, ar kuria bei valdo daug programėlių vienu metu) ir darbo mobiliųjų programėlių kūrimo ir valdymo srityje patirtį, darant prielaidą, jog šie kriterijai gali daryti įtaką tiek ekspertų pasakojimui apie programėlių kūrimo ir valdymo procesą, tiek paties proceso pobūdžiui. Susistemintos respondentų charakteristikos pateikiamos 4 lentelėje. Kadangi tam tikros charakteristikos apie ekspertus paaiškėjo tik tyrimo metu, ekspertų pasiskirstymas skirtingų trianguliacijos principo aspektų požiūriu nėra vienodas.

4 lentelė. Struktūruoto interviu informantų-ekspertų charakteristikos ir kriterijai

	Ekspertas Nr. 1	Ekspertas Nr. 2	Ekspertas Nr. 3	Ekspertas Nr. 4	Ekspertas Nr. 5	Ekspertas Nr. 6	Ekspertas Nr. 7	Ekspertas Nr. 8	Ekspertas Nr. 9
Aukšto lygmens vadovas				+	+			+	+
Vidutinio lygmens vadovas		+	+						
Vykdančios funkcijos	+					+	+		
Didelė/vidutinė įmonė	+	+	+	+			+		+
Maža įmonė/startuolis					+			+	
Savarankiškai dirbantis						+			
Specializacija: *frontend*	+					+			
Specializacija: *backend*				+	+		+		
Specializacija: *backend* ir *frontend*		+	+					+	+
Veiklos pobūdis: programėlių kūrimo agentūros principas		+	+			+			
Veiklos pobūdis: vienos programėlės palaikymas	+			+	+		+	+	+
Programėlių kūrimo patirtis 5-7 m.	+	+							+
Programėlių kūrimo patirtis 8-10 m.				+		+		+	
Programėlių kūrimo patirtis daugiau nei 11 m.			+		+		+		

Šaltinis: sudaryta tyrimo autorės

13 paveiksle pateikiamos diagramos, kuriose grafiškai pavaizduotas ekspertų pasiskirstymas pagal 1) pareigybės lygį, 2) atstovaujamos įmonės dydį bei 3) specializaciją. Ketvirtoje diagramoje atsiskleidžia ekspertų pasiskirstymas pagal pagrindinį atrankos kriterijų – patirtį mobiliųjų aplikacijų kūrimo ir valdymo srityje. Ekspertu tyrime laikomas asmuo, kuris tokios darbo patirties yra sukaupęs ne mažiau nei 5 metus. Visi tyrime dalyvavę ekspertai šį kriterijų atitinka.

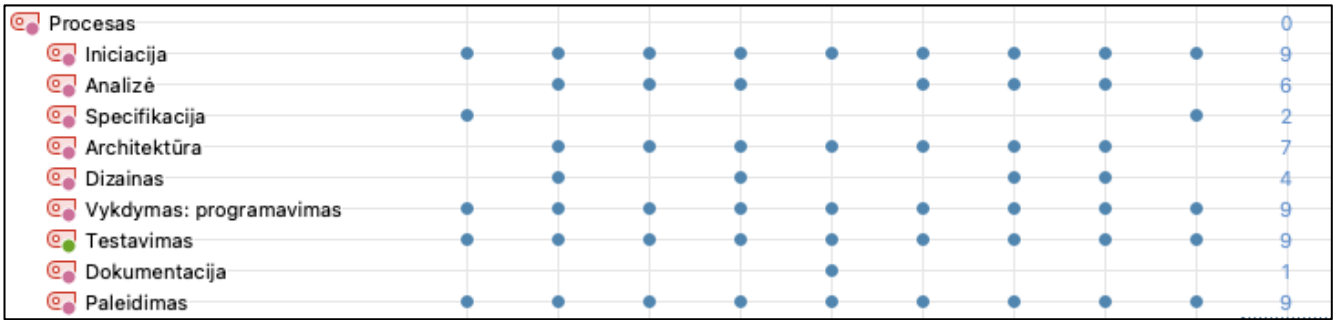
Šaltinis: sudaryta tyrimo autorės

13 pav. Tyrimo informantų pasiskirstymas pagal svarbiausias charakteristikas

Pagrindinių klausimų dalyje – 11 klausimų. Pirmiausia ekspertų buvo prašoma trumpai nupasakoti jų įmonėje ar veikloje taikomą programėlių ar jų atnaujinimų kūrimo procesą. Apibendrinus informantų atsakymus gauti tokie proceso žingsniai (čia ir toliau skliausteliuose rašoma kiek ekspertų iš 9 minėjo šį proceso žingsnį, pavyzdžiui, 2/9 - 2 ekspertai iš 9):

1. Iniciacija (9/9)
2. Analizė (7/9)
3. Architektūra (7/9) (kitų respondentų įvardinta kaip specifikacija (2/9))
4. Vizualiniai dizaino sprendimai (7/9)
5. Programavimas (9/9)
6. Testavimas (9/9)
7. Patalpinimas į parduotuves (9/9)

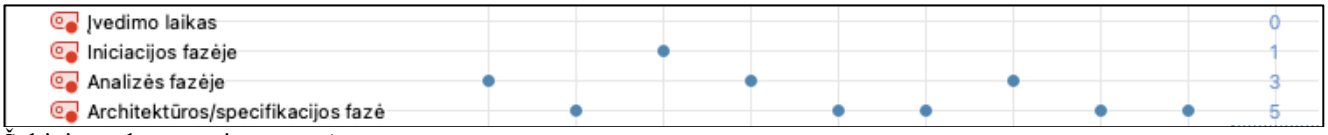
Gauti rezultatai leidžia daryti išvadą, jog programėlių ir jų naujinių kūrimo procesas – gana standartizuotas ir skiriasi ne esminiai, o techniniai proceso aspektai, pavyzdžiui, kai kuriose įmonėse dizaino fazėje yra daroma vartotojo sąsajos schema (*wireframe*), kitose iš karto daromi vizualinių dizaino elementų kombinavimai ir sprendimai. Pažymėtina, jog vienas informantas atlieka tik programavimo darbus, todėl jis apie procesą pasakojo jau po dizaino fazės. Informantų atsakymų pasiskirstymas atsispindi 14 paveiksle:



Šaltinis: sudaryta tyrimo autorės

14 pav. Mobiliosios programėlės kūrimo proceso žingsniai pagal tyrimo ekspertus

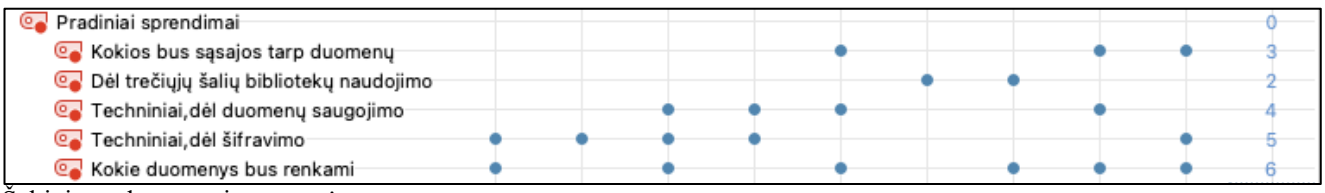
Paklausti, kuriame programėlių ar jų naujinių kūrimo proceso etape dažniausiai įvedami saugumo sprendimai, 5 informantai iš 9 teigė, jog saugumo sprendimų pradedama ieškoti architektūros ar specifikacijos fazėje (šios fazės pagal daugelį modelių priskiriamos prie programėlės ar jos naujinio techninio ir vizualinio dizaino, todėl tyrime yra sugrupuotos). 3 ekspertai teigė, jog apie saugumo sprendimus pradedama kalbėti analizės fazėje, tuo tarpu 1 ekspertas teigė, jog apie saugumo sprendimų pradedama ieškoti dar iniciacijos fazėje. Informantų atsakymai iliustruojami 15 paveiksle:



Šaltinis: sudaryta tyrimo autorės

15 pav. Saugumo sprendimų įvedimo į procesą laikas

Paprašyti padetalizuoti, kokio tipo sprendimai tai dažniausiai būna, dauguma teigė pirmiausia priimantys sprendimus, su kokiais duomenimis bus dirbama (6/9), kokie techniniai duomenų šifravimo (5/9) ir saugojimo (4/9) sprendimai bus naudojami, apgalvojamos būsimos sąsajos tarp naujų ir jau turimų duomenų (3/9), bei kokios trečiųjų šalių bibliotekos bus naudojamos (2/9). Ekspertų atsakymų pasiskirstymas schematiškai pavaizduotas 16 paveiksle:



Šaltinis: sudaryta tyrimo autorės

16 pav. Saugumo sprendimai, ekspertų priimami pirminėje fazėje

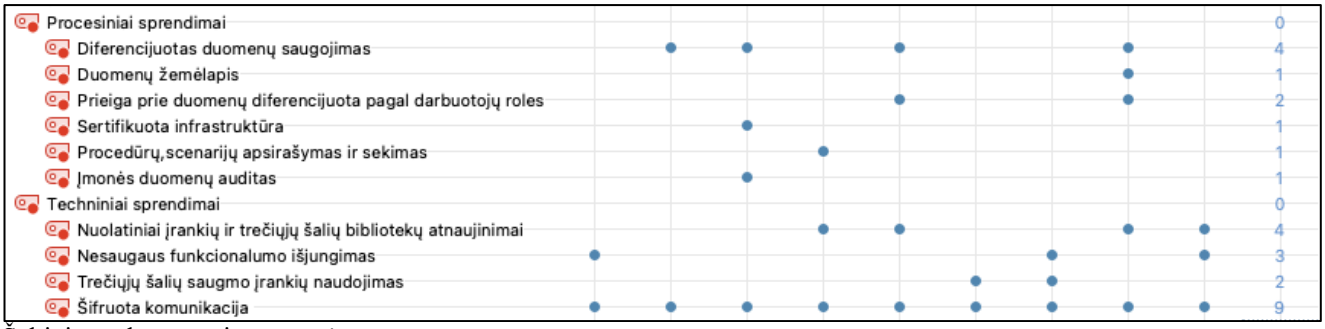
7 ekspertai teigė, jog priimant programėlių ar jų naujinių saugumo sprendimus atsižvelgiama į tai, kokie duomenys bus tvarkomi, tuo tarpu 2 ekspertai savo klientams siūlo standartizuotą paslaugą ir bet kurie nauji sprendimai neturi įtakos pamatiniam, strateginiam saugumo lygiui, kurį jie tiek techniniais, tiek procesiniais sprendimais stengiasi išlaikyti itin aukštą. Su šiuo teiginiu buvo susijęs ir kitas klausimas: „kokių sprendimų imatės, kad jūsų programėlės būtų saugios?“, kurio atsakymus ir tikslinga skirstyti į procesinius ir techninius. Ekspertų organizacijose taikomi procesiniai saugumo sprendimai:

1. Diferencijuotas duomenų saugojimas (skirtinguose serveriuose patalpintose duomenų saugyklose) (5/9);
2. Prieigos prie duomenų diferencijavimas organizacijos mastu (kokie darbuotojai kokius duomenis gali matyti) (2/9);
3. Procedūrų ir scenarijų aprašų naudojimas (2/9);
4. Duomenų žemėlapio naudojimas (1/9);
5. Sertifikuotos IT infrastruktūros naudojimas (1/9);
6. Organizacijos turimų duomenų auditas (1/9);

Informantų minimi techniniai saugumo sprendimai buvo šie:

1. Šifruota komunikacija (9/9);
2. Diferencijuota sistema su galimybe atjungti blogai veikiantį ar saugumo spragų turintį funkcionalumą (4/9);
3. Nuolatiniai programėlėse naudojamų trečiųjų šalių įrankių ir bibliotekų atnaujinimai (4/9);
4. Trečiųjų šalių saugumo įrankių naudojimas (pavyzdžiui, „Android Firebase“ naudojimas vartotojo autentifikacijai) (3/9);

Susisteminti ekspertų atsakymai pateikiami 17 paveiksle.

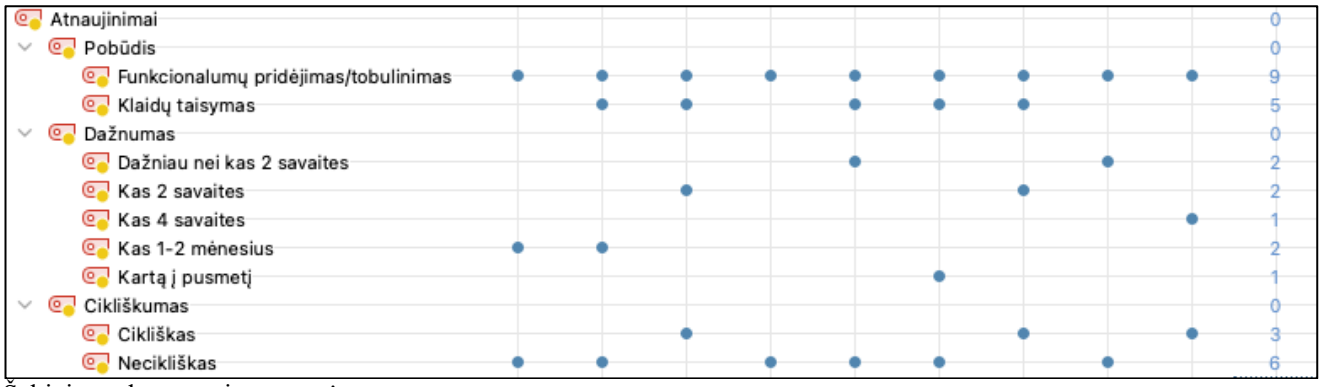


Šaltinis: sudaryta tyrimo autorės

17 pav. Sprendimai, padedantys kurti ir valdyti saugias programėles

Ekspertų taip pat buvo teiraujamasi, kas kiek laiko jie paprastai leidžia programėlių naujinius ir kokio pobūdžio atnaujinimai tai paprastai būna (naujų funkcionalumo diegimai, funkcionalumo klaidų taisymai, saugumo spragų lopymai ar pan.). 3 ekspertai iš 9 atsakė, jog programėlių atnaujinimai jų organizacijose/veikloje leidžiami ciklais: 2 organizacijų programėlių ar jų naujinių kūrimo ciklas trunka 2 savaites, 1 – 4 savaites. Visi trys informantai atstovavo dideles įmones. Tuo tarpu 6 ekspertai naujinius

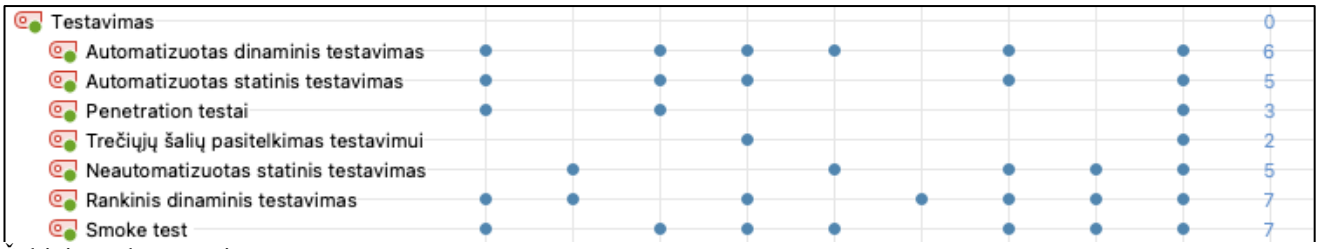
leidžia pagal poreikį, o jų dažnis skiriasi. 2 iš minėtų 6 informantų organizacijose/veiklose naujiniai leidžiami kas 1-2 mėnesius, 2 – dažniau nei kas 2 savaites. Pažymėtina, jog taip dažnai atnaujinimus leidžia tyrime dalyvavusių ekspertų darbovietės, turinčios startuolio statusą. 1 ekspertas teigė, jog atnaujinimus daro kartą į pusę metų, tačiau tai – ekspertas, dirbantis tik su aplikacijos *frontendu*, todėl tikėtina, jog jis neturi visų žinių apie sistemos, kuriai priklauso programėlė, *backendo* atnaujinimus. Visi ekspertai teigė, jog naujiniais dažniausiai diegia naują ar tobulina esamą funkcionalumą, klaidų taisymą daro rečiau (jį paminėjo tik 5 ekspertai iš 9). Paklausti, ar daro spragų lopymą, visi teigė, jog jei spragų atsiranda, jas taiso kaip įmanoma greičiau, tačiau tai atsitinka retai, nes jie dėmesio skiria saugumo spragų prevencijai. Susisteminta informacija apie informantų atstovaujamų organizacijų/veiklų atliekamus programėlių atnaujinimus pateikiama 18 paveiksle.



Šaltinis: sudaryta tyrimo autorės

18 pav. Ekspertų atstovaujamų organizacijų/veiklų atliekami mobiliųjų programėlių atnaujinimai

Kitas klausimas, kurio buvo teiraujamasi ekspertų – kaip testuojamos programėlės ir jų atnaujinimai. 7 ekspertai iš 9 (t. y., 78 proc. informantų) atsakė, jog jų atstovaujamose organizacijose/veiklose programėlės. 6 informantai iš 9 naudoja automatinius, specialiai sukurtus testus (pasitelkiant automatizuotą dinaminį testavimo metodą). Po 5 ekspertus taip pat teigė, jog jų organizacijose/veiklose atliekamas automatizuotas statinis testavimas (t. y., automatizuotai tikrinamas programėlės kodas) bei neautomatizuotas statinis testavimas – kai programuotojas peržiūri kito kolegos parašytą kodą. 3 informantai taip pat pažymėjo, jog programėlių saugumą padeda užtikrinti trečiųjų šalių vykdomas įsiskverbimo testavimas (*penetration tests*, kai bandoma įsilaužti į programėlę, specialiai suvesti netinkamus duomenis ir pan., siekiant atrasti imituoti įsilaužėlių atakas), o 2 ekspertai pasitelkia trečiasias šalis visapusiškam programėlės ištestavimui. 7 ekspertai iš 9 taip pat atlieka vadinamąjį *smoke test* – jau paleistos ir vartotojams prieinamos programėlės testavimą, dažniausiai prieš paleidžiant naują funkcionalumą, siekiant įsitikinti, jog su atnaujinimu programėlė veiks korektiškai. Ekspertų atstovaujamų organizacijų/veiklų naudojamų mobiliųjų aplikacijų testavimo metodų suvestinė pateikiama 19 paveiksle:

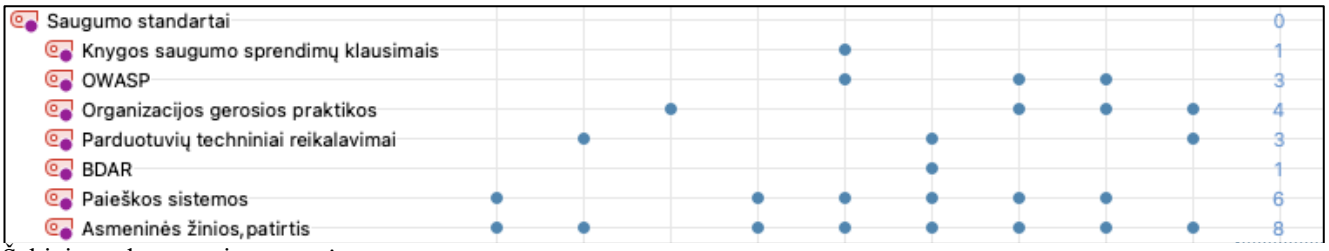


Šaltinis: sudaryta tyrimo autorės

19 pav. Ekspertų atstovaujamų organizacijų/veiklų naudojami mobiliųjų aplikacijų testavimo metodai

Ekspertų buvo klausiama ar jų organizacijose/veikloje yra atliekami su mobiliosios programėlės saugumu susiję rizikos vertinimai. 4 ekspertai atsakė, jog nedaromas (nors du iš jų teigė, jog planuoja pradėti juos daryti), 2, jog daromas, bet be aiškios metodologijos, o 1, jog daromas pagal organizacijos mastu patvirtintą rizikos vertinimo metodologiją. 2 ekspertai negalėjo atsakyti į šį klausimą, nes į programėlės kūrimo procesą įsijungia vėliau, jau po analizės ir specifikacijos fazės.

Informantų taip pat buvo teirautasi, kokiais saugumo standartais (jei tokie egzistuoja), kokia informacija jie remiasi kurdami ir valdydami programėles. Visi ekspertai vienbalsiai sutiko, kad visuotinai priimto saugumo standarto nėra, todėl dauguma (8 informantai iš 9) pasikliauja savo asmeninėmis žiniomis ir patirtimi. Didžioji dauguma ekspertų informacijos taip pat ieško paieškos sistemose (6/9), kiti kliaujasi vidiniais organizacijų gerųjų praktikų dokumentais (4/9), aplikacijų parduotuvių reikalavimais (3/9). 3 ekspertai taip pat minėjo OWASP mobiliųjų pažeidžiamumų sąrašą, 1 teigė informacijos ieškantis knygose, 1 – kad kaip vieną saugumo standartų vertina BDAR.



Šaltinis: sudaryta tyrimo autorės

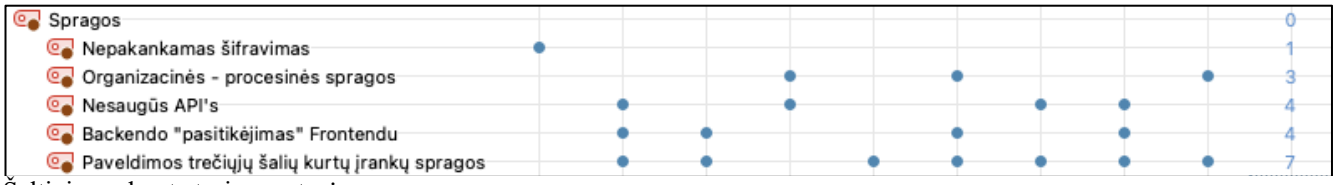
20 pav. Ekspertų įvardinti mobiliųjų aplikacijų saugumo standartai

Paskutiniai du pagrindinių klausimų kategorijoje buvo klausimai apie saugumo spragas. Informantų buvo prašoma įvardinti dažniausiai kylančias saugumo spragas bei teiraujamasi, kaip jų būtų galima išvengti. Kaip dažniausiai iškylančią spragą ekspertai įvardino trečiųjų šalių bibliotekų ar įrankių saugumo spragas, kurias paveldi jų kuriama ar valdoma programėlė (8/9). Kaip dažniausiai pasitaikantį pažeidžiamumą dėl programėlės ar jos atnaujinimo kūrėjo kaltės 4 ekspertai iš 9 įvardino *backendo* pasitikėjimą *frontendu*. Cituojant vieną respondentų:

„*Dažnai nutinka taip, kad develop'indami develop'eriai kažkodėl nusprendžia, kad visa tai, ką išsitraukė iš duombazės aš numesiu atgal į aplikaciją, o aplikacija pati nuspręs, ko jai reikia, ko jai nereikia. Tai*

su šitu sprendimu problema yra tai, kad bet kuris hacker'is ar bet kuris daugiau IT žinių turintis žmogus gali tą traffic'ą tarp mobiliosios aplikacijos ir back end'o stebėti.“

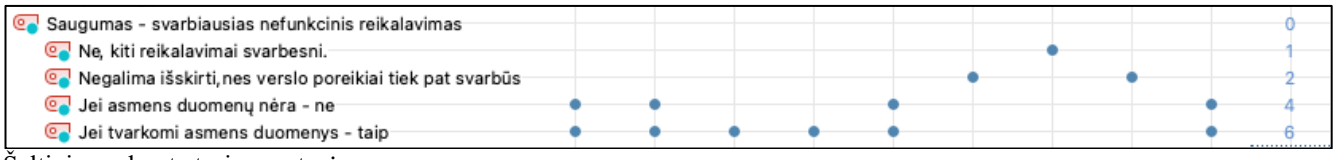
Perfrazuojant, programuotojai kartais iš *backendo* į programėlę siunčia daugiau informacijos, negu programėlei reikia, galvodami, jog programėlė pati atsirinks ir vartotojui parodys tik tuos duomenis, kuriuos jam reikia matyti, tačiau komunikaciją tarp programėlės ir serverio stebintis tarpinis žmogus ar įrenginys gali nuskaityti visą perduodamą informaciją. Su komunikacija susijusius pažeidžiamumus įvardino ir kiti 4 informantai, teigdami, jog daug spragų atsiranda API sąsajose. 3 ekspertai pastebėjo, jog spragos atsiranda dėl organizacinių-procesinių priežasčių. 1 informantas kaip spragų atsiradimo priežastį paminėjo nepakankamą duomenų šifravimą. Susistemintą informaciją apie ekspertų įvardintas spragas iliustruoja 21 paveikslas. Paklausti, kaip šių spragų būtų galima išvengti, ekspertai akcentavo, jog svarbu nuolat atnaujinti trečiųjų šalių bibliotekas, naudoti naujausias įrankių versijas, atidžiai rašyti programėlių kodą bei jas testuoti, operatyviai reaguoti bei taisyti atsiradusias spragas.



Šaltinis: sudaryta tyrimo autorės

21 pav. Ekspertų įvardintos mobiliosiose programėlėse atsirandančios saugumo spragos

Papildomų klausimų dalyje ekspertams buvo pateikiami 3 klausimai, siekiant išsiaiškinti mobiliųjų aplikacijų rinkos pokyčius požiūrio į saugumą klausimais. Pirmiausia informantų buvo klausiama, ar jie sutinka su teiginiu, jog saugumas – svarbiausias nefunkcinis mobiliųjų aplikacijų reikalavimas. 6 ekspertai laikėsi nuomonės, jog jei programėlė turi prieigą prie jautrių duomenų, šis reikalavimas – svarbiausias, tačiau 4 iš jų taip pat pabrėžė, jog jei programėlė prie tokių duomenų neprieina, svarbesni yra tokie reikalavimai kaip jos prisitaikymas prie skirtingų įrenginių, greitaveika ir pan. 2 ekspertai mane, jog verslo poreikiai, t. y., patogumas vartotojui, greitaveika, prisitaikymas yra tiek pat svarbūs kiek programėlės saugumas, todėl vieno nefunkcinio reikalavimo nebūtų galima išskirti kaip svarbiausio. 1 ekspertas laikėsi nuomonės, jog patogumas vartotojui yra svarbiau, nei saugumas.

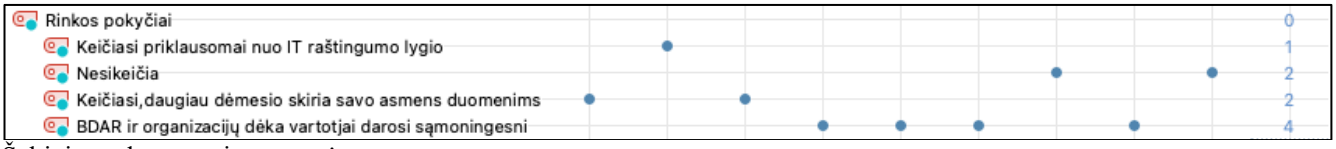


Šaltinis: sudaryta tyrimo autorės

22 pav. Ekspertų nuomonė apie saugumą kaip apie svarbiausią nefunkcinį reikalavimą

Informantų taip pat buvo klausiama, ar, jų nuomone, programėlių vartotojai daugiau dėmesio skiria programėlių saugumui. Dauguma ekspertų, 7 iš 9 apklaustųjų, mano, kad vartotojų sąmoningumas saugumo klausimu keičiasi: 4 šią situaciją vertina kaip BDAR ir organizacijos vykdomos asmens

duomenų apsaugos politikos bei su tuo susijusio švietimo pasekmes; 2 – kaip natūralų tobulėjimo procesą; 1 ekspertas linkęs įžvelgti ryšį tarp vartotojų sąmoningumo ir IT raštingumo. Tuo tarpu 2 ekspertai laikosi nuomonės, jog vartotojai kol kas neskiria daugiau dėmesio savo asmens duomenims, taigi šiuo klausimu rinka nesikeičia.



Šaltinis: sudaryta tyrimo autorės

23 pav. Ekspertų nuomonė apie vartotojų sąmoningumas saugumo klausimais

Galiausiai ekspertų buvo klausiama, ar jie mano, jog mobiliąsias programėles kuriančių ir/ar valdančių organizacijų požiūris į programėlių saugumą yra daugiau techninis (apimantis techninius sprendimus), ar daugiau strateginis (apimantis požiūrį ir resursų paskirstymą) klausimas. Visi 9 respondentai atsakė, jog programėlių saugumas jų atstovaujamose organizacijose – strateginis klausimas, tačiau keli pastebėjo, jog šis požiūris dar nėra plačiai paplitęs. Informantų citatos:

„Manau, strateginis ir dar manau, kad mano įmonė – puikus pavyzdys, jau yra pasitaikę variantų, kai į mūsų projektą žmonės perėjo dėl to, kad kito projekto sprendimai buvo mažiau saugūs.“

„Saugumas yra mums visiškai strateginis, nes mūsų galimybė vykdyti veiklą yra susijusi su tuo, koks yra mūsų patikimumas.“

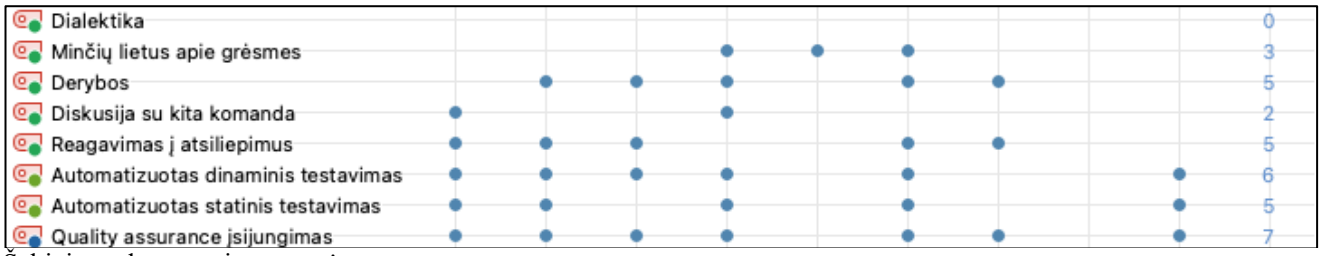
„Jei paimsime ilgalaikę perspektyvą, tai tikrai yra strateginis, nes niekas nenori, kad išlįstų jų vartotojų duomenys, kažkokios spragos, ir jų įmonės vardas būtų suterštas. Bet manau reikia stiprios kompanijos, kad tie, kurie kuria patį funkcionalumą, […] galėtų atsiremti į savo profesionalumą ir užtikrinti, kad programėlė naudoja geriausias praktikas visiems saugumo dalykams.“

„Jeigu tai nėra strateginis klausimas, jam nėra skiriama pakankamai dėmesio, jei jam nėra skiriama pakankamai dėmesio, pradeda lįsti kvailiausios klaidos […]. Šioje vietoje tikrai labai padeda GDPR‘as ir nauji data leakeg‘ai, nes visi susimąstė, ar yra saugūs.“

„[Saugumas] turi būt strategijos dalis ir tas turi būt įvardinta strategijoj. Ir kai tai yra įvardinta strategijoj, tada organizacijai bus žymiai aiškiau. Ir, manau, vadovybei bus žymiai aiškiau, ko jie gali tikėtis iš to.“

„Aš manau, kad saugumas – strateginis klausimas. Šiai dienai, aš manau, bent jau iš to, ką aš pastebiu, […] tai tiktai dideli enterprise'ai, kurie sutartyse įsideda ten punktus apie didžiules, kosmines baudas, skiria [pakankamą] dėmesį saugumui. Visos kitos aplikacijos, kurios yra kuriamos kontoroje, kurioje programėlės yra „kepamos", […] ten viskas daroma daug paprasčiau, daug greičiau ir saugumas... Aš tikiu, kad jie apie saugumą galvoja, bet galbūt jie neturi tiek laiko ir neturi tiek įdirbio, ir neturi tiek iš vidaus žmonių, kurie galėtų pakankamai daug security klausimų padengti."

Atliekant tyrimo rezultatų analizę buvo pastebėta, jog daugelis informantų atstovaujamų organizacijų taiko bent po kelis dialektikos žingsnius, todėl tikslinga tyrimo duomenis išnagrinėti ir dialektikos kontekste. Minčių lietaus metodologiją siekiant identifikuoti galimus įsilaužėlius taiko 3 informantų atstovaujamos organizacijos. Derybos tarp programėlių saugumą užtikrinančių bei verslo poreikius užtikrinti siekiančių suinteresuotųjų šalių (darbuotojų, klientų), vyksta 5 organizacijose iš 9. Kitos komandos įžvalgas pasitelkia 2 organizacijų darbuotojai. Kokybės užtikrinimo arba testuotojų roles ar komandas turi 7 organizacijos iš 9. Tiek statinis, tiek dinaminis automatizuoto testavimo metodas taikomas 5 organizacijose, tik automatizuotas dinaminis – 1 organizacijoje. 5 organizacijų atstovai teigė, jog skiria daug dėmesio programėlės veikimo analizei bei vartotojų atsiliepimams ir programėlę taiso ar tobulina atsižvelgdami į juos. Taigi, turint omenyje, jog tiek automatizuotas dinaminis, tiek statinis testavimo metodai priskiriami vienam dialektikos žingsniui, darytina išvada, jog 2 organizacijos iš 9 taiko 5 dialektikos žingsnius, 3 organizacijos – 4 žingsnius.



Šaltinis: sudaryta tyrimo autorės

24 pav. Dialektikos žingsniai ekspertų atstovaujamose organizacijose

Kelios dialektikos žingsnių taikymą organizacijose iliustruojančios informantų citatos:

„Bet tie patys reikalavimai, jie yra tiek keliami projekto pakeitimo vizijai, sakykim, tokiam visam konceptualiam lygy, tiek tada mes, kartu su architektais, sėdim ir galvojam – nes jie, aišku, atsineša, kaip patogiausia tai padaryti, nebūtinai saugiausia – tai mes sakom, ne, šito negalima daryti. Jie sako, mums reikia tą daryti greičiau, nu ok, mes sakom, šitą galbūt galima kažkaip tai pakoreguot, bet šitie yra neliečiami dalykai ir jų negalima kažkaip kitaip daryti."

„[…] davėme kitai komandai ištestuoti, tai dar pagal jų siūlymus pataisėme, atlikome korekcijas."

„Kad klientas suprastų, kad dalis, ko jis nori ne visada yra įmanoma, arba galima padaryti geriau, negu jisai tikisi, bet tam reikia sukalibruoti lūkesčius.“

„[...] ar reikia mums tų duomenų, kodėl mums būtent to duomens reikia, ką reikia padaryti, kad nekaupiant tam tikrų duomenų nesikeistų funkcionalumas.“

„Komandų daug, bet visos daro tą patį appsą, tai reikia visiems sinchronizuotis. Yra atskiros gildijos, vienijančios tų pačių funkcijų, bet skirtingų komandų narius, kuriose aptariama, pavyzdžiui, kaip skirtingom platformom pritaikyti geriausiai programėlę. Tokiu būdu yra komandos ir yra kitas pjūvis – gildijos, organizacijos prasme. O komandos – tam tikro funkcionalumo klausimams.“

Pažymėtina, jog organizacijas, kuriose taikomi tokie žingsniai kaip minčių lietaus metodas programėlės ar jos naujinio analizės ar specifikacijos etape, derybos su verslo tikslus ginančiomis suinteresuotomis šalimis, diskusijos su kita komanda ir kt., atstovaujantys ekspertai šiems žingsniams skyrė didelę svarbą, įvardino kaip reikšmingą organizacinės kultūros dalį bei vieną pagrindinių aspektų, padedančių palaikyti aukštą programėlių saugumą ir kokybę apskritai.

Ekspertų sąmoningai nebuvo klausiama kokiam mobiliosios aplikacijos kūrimo modeliui artimiausas jų procesas, vengiant kelių žodžių atsakymo (pavyzdžiui, proceso modelio pavadinimo) ir siekiant gauti platesnį, kontekstualesnį proceso apibūdinimą. Tačiau iš informantų atsakymų galima daryti tvirtas išvadas, jog 7 iš 9 ekspertų atstovaujamos organizacijos veikia „Agile“ principu: informantai minėjo prioritetų eiles, kassavaitinius ar kasdienius susitikimus, iteracijas (kai kurie įvardino ir ciklišku sprintus – tai itin aiškiai išryškėjo klausimuose apie programėlių atnaujinimų leidimą), kai kurie įvardino „Agile“ modelį, pagal kurį dirba. Kelios išvadą pagrindžiančios ekspertų citatos:

„Visi funkcionalumai yra sudedami į prioritetinę eilę ir tada pagal tą eilę ir vyksta programavimas“

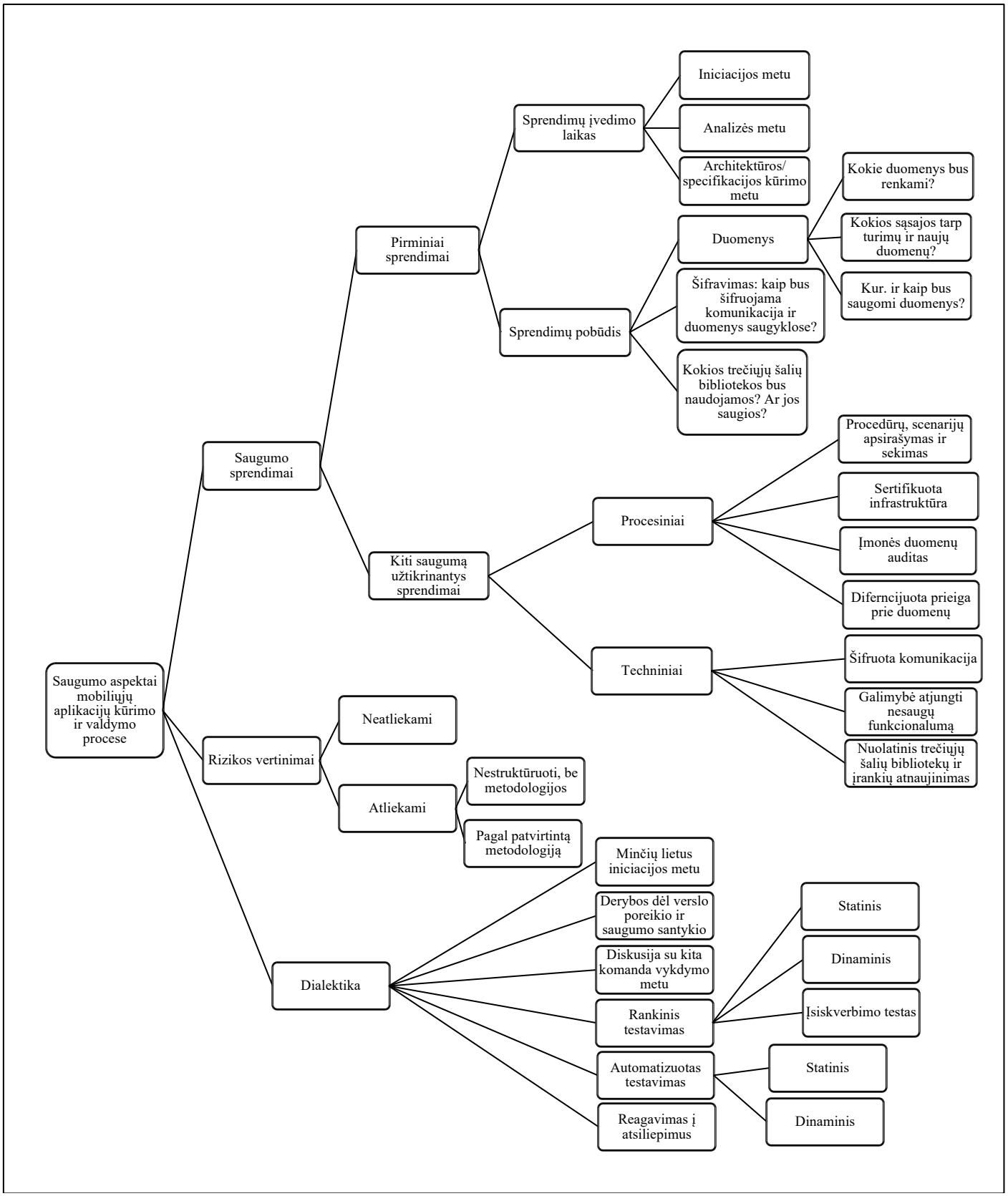
„[...] mūsų release cycle’as, tas programėlių naujinimo ciklas iš esmės yra dvi savaitės. Tai kas dvi savaites mes turim tam tikrą planavimą, į kurį mes įtraukiame, visų pirma - naujus feature’us, naujus funkcionalumus ir lygiagrečiai (su) tuo pačiu įtraukiam tam tikrus taisymus, bug fix’us arba tobulinimus, pakeitimus, kurie yra tam tikri, minimalūs.“

[...] skirtingų funkcijų žmonės priklauso savo gildijoms. Jie turi kassavaitinius susitikimus, kuriuose diskutuoja apie tai, kaip daro patį appsą.

Likusių trijų ekspertų nupasakotos procesų charakteristikos nebuvo tokios ryškios, todėl vienareikšmiškų išvadų apie organizacijoje/veikloje taikomą mobiliosiųjų aplikacijų kūrimo ir palaikymo modelį daryti negalima.

Tyrimo išvados

- Visi informantai laikosi nuomonės, jog programėlės saugumas yra jų atstovaujamų organizacijų strateginis klausimas, tačiau tik 3 ekspertų atstovaujančiose organizacijose saugumo sprendimai buvo įvedami anksčiau nei architektūros/specifikacijos fazėje.
- Programėlių kūrėjai ir valdytojai rimtai žiūri į programėlių testavimą, jam skiria daug dėmesio;
- Didžioji dalis programėlių atnaujinimų daroma dėl naujo funkcionalumo įvedimo ar esamo funkcionalumo pagerinimo, ne dėl saugumo spragų lopymo ar klaidų taisymo. Darytina išvada, jog programėlių saugumo spragos identifikuojamos ir pašalinamos dar iki programėlės paleidimo, daug dėmesio skiriama jų prevencijai: duomenų šifravimui, nuodugniam programėlės testavimui, trečiųjų šalių įrankių naudojimui tam, kad būtų pasiektas aukščiausias galimas saugumo lygis ir pan.;
- Mobiliųjų aplikacijų pasiekiamų, tvarkomų ar valdomų duomenų rizikos vertinimai didžiojoje dalyje organizacijų nėra atliekami (tiksliau yra atliekami 33 proc. tyrime atstovaujamų organizacijų. Jos visos priklauso), tačiau keli informantai išreiškė siekį ateityje tokį vertinimą įtraukti į programėlių kūrimo ir valdymo procesą, mato šio proceso reikalingumą.
- Nors tik vos daugiau nei pusėje ekspertų atstovaujamų organizacijų taikomi 4 ir daugiau dialektikos žingsnių, pastebėtina, jog tyrime dalyvavę informantai aiškiai supranta šių žingsnių naudą ir juos priskiria prie strateginę reikšmę organizacijos sėkmei turinčių aspektų;
- Daugumoje (67 proc.) ekspertų atstovautų organizacijų vadovaujamasi „Agile“ ar iš jos išvestomis metodologijomis . Likusių organizacijų/veiklos procesai neturėjo pakankamo kiekio kažkuriai metodologijai priskirtinų charakteristikų, kad būtų galima daryti išvadas apie jose taikomą metodologiją;
- 7 iš 9 informantų mano, jog mobiliųjų aplikacijų vartotojai tampa sąmoningesni savo asmens duomenų klausimais;
- Tyrimo metu išryškėję mobiliųjų programėlių saugumo aspektai pavaizduoti 14 paveiksle:



Šaltinis: sudaryta tyrimo autorės

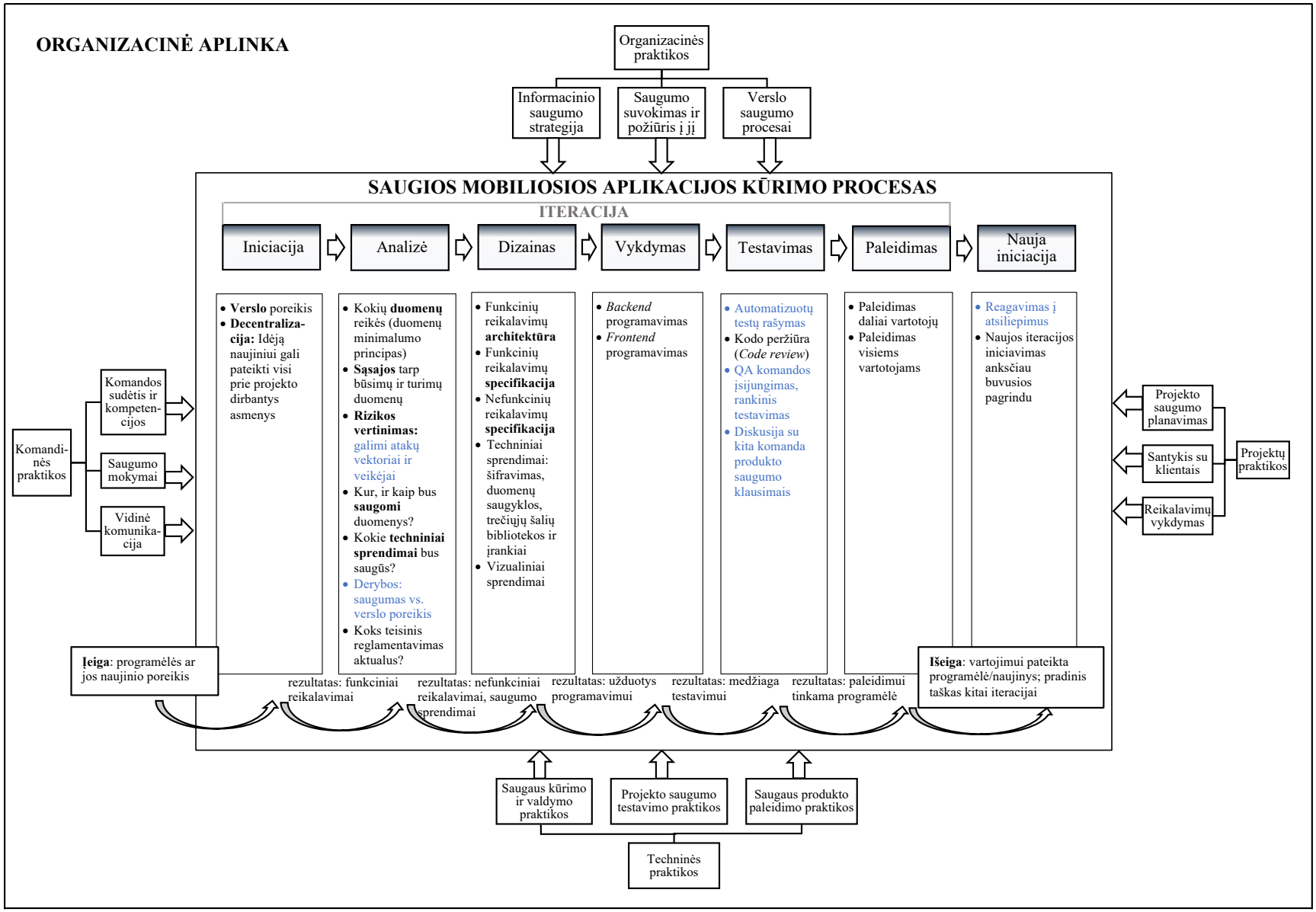
25 pav. Tyrimo dalyvių išskirtų saugumo aspektų ontologija

3. Į SAUGUMĄ ORIENTUOTO MOBILIŲJŲ APLIKACIJŲ KŪRIMO IR VALDYMO MODELIO KŪRIMAS

3.1. Modelio kūrimo metodologija

Tyrimo metu buvo išryškinti praktiniai programėlių kūrimo ir valdymo aspektai. Jungiant juos su teorinėje dalyje atskleista mobiliųjų aplikacijų saugumo problematika ir jos sprendimo būdais bei remiantis jau sukurtais ir įsitvirtinusiais mobiliųjų aplikacijų kūrimo ir valdymo modeliais, bus kuriamas į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis. Kadangi mobiliųjų programėlių kūrimas – procesas, kuriamas modelis taip pat turi būti procesinis, arba, pagal Tatjanos Siderskienės modelių kvalifikaciją – **kokybinio proceso aprašas** (Gulevičiūtė, 2014). Procesiniai aprašai gali būti aiškinantieji/aprašantiejii (*descriptive*), arba patariantieji/rekomendacinio pobūdžio (*perscriptive*) (Rossi, 2002). Tyrimo tikslas – pateikti unifikuotą saugių mobiliųjų aplikacijų kūrimo ir valdymo modelį, potencialiai padėsiantį išspręsti visuotinai pripažįstamų standartų ir saugumo reikalavimų nebuvimo (Weir ir kt., 2017; Weir ir kt., 2020; Jha, Mahmoud, 2019) bei su tuo susijusias skirtingų programėlių saugumo interpretacijų bei vertinimo problemas, todėl **tikslinga kurti patariančiojo/rekomendacinio tipo modelį.**

Siekiant kuo paprastesnio modelio pritaikomumo, tikslinga remtis „Agile“ metodologijia, kuri šiuo metu vyrauja programėlių kūrimo ir valdymo procesuose (Edison ir kt., 2018; Pandey ir kt., 2019; Flora ir kt., 2014). Taip pat pažymėtina, jog pagal modeliavimo metodologiją, **procesinis modelis turi turėti įeigą ir išeigą** (Aytulun, Guneri, 2008; Gulevičiūtė, 2014), todėl į modelį rerikalinga įtraukti ir šiuos elementus. Sukurtas modelis remiasi „Agile“ metodologija, Newton ir kt. „Kritinių saugumo faktorių“ metodologija bei Weir ir kt. Dialektikos metodologija. Modelis pateikiamas 26 paveiksle.



Šaltinis: sudaryta tyrimo autorės

26 pav. Į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis

3.2. Modelio verifikacija

Verifikuojant modelį svarbu atsakyti į du klausimus:

1. **Ar modelis – reikalingas, patogus naudoti, pritaikomas?**
2. **Ar atliktas darbas pasiekia iškeltus tikslus?** (Hicks ir kt., 2015)

Nors mobiliųjų aplikacijų saugumas tampa vis aktualesniu tiek dėl asmens duomenų apsaugos reglamentavimo, tiek dėl augančio vartotojų sąmoningumo, **programėlių kūrimo ir valdymo modelio, akcentuojančio aplikacijų saugumo aspektus, iki šiol nebuvo sukurta**. Kuriamas modelis parimtas „Agile“ metodologija, kuri tiek moksliniuose šaltiniuose, tiek šio darbo apimtyje atlikto tyrimo rezultatuose pripažįstama kaip vyraujanti kuriant ir valdant mobiliąsias programėles, modelio pritaikomumas nebūtų sudėtingas. „Agile“ metodologijai būdingas ciklišKumas, proceso vykdymas iteracijoms. Šis principas atsiskleidžia ir pateiktame modelyje (nors modelis dėl aiškumo pavaizduotas linijiškas, į taivertėtų žiūrėti kaip į savotišką išklotinę). Svarbus ir dialektikos įvedimas – ji savotiškai įformalizuoja, dokumentuoja neformalų skirtingų kompetencijų specialistų bendravimą ir bendradarbiavimą vardan bendro tikslo ir padeda siekti geresnių rezultatų mobiliųjų programėlių saugumo srityje. Modelyje dialektikos žingsniai, paremti C. Weir ir kt. 2017 m. tyrimu, išskirti mėlyna spalva. Taip pat pažymėtina, jog vadovaujantis sukurtu modeliu, saugumo sprendimų turi būti pradedama ieškoti jau analizės fazėje, o suformuluoti saugumo sprendimai tampa analizės fazės rezultatu.

Siekiant patvirtinti, jog modelis atitinka reikalingumo, patogumo ir pritaikomumo reikalavimus, svarbu aptarti visas modelio dalis (Hicks ir kt., 2015).

Organizacinė aplinka. Pažymėtina, jog mobiliųjų programėlių kūrimo procesas neegzistuoja vakuume, taigi ir produkto saugumas priklauso nuo jos kūrėjų kompetencijų, žinių, turimų technologijų bei kitų svarbių aspektų. Todėl į saugumą orientuotas aplikacijų kūrimo ir valdymo modelis yra tiesiogiai veikiamas organizacinių, komandinių, projektų ir techninių saugumo praktikų, programėlės kuriamos organizaciniame kontekste. Kitaip tariant, koncentruojantis į saugios programėlės kūrimą, bet neturint informacinio saugumo strategijos ir ją lydinčių dokumentų, gali nutikti taip, kad, pavyzdžiui, dėl organizacijos naudojamo serverio spragų programėlės *backende* atsiras paveldimas pažeidžiamumas ir programėlė taps nebesaugia. Analogiška situacija gali iškilti neturint pakankamas kompetencijas įgijusių darbuotojų ar neatliekant kitų kontekstiniame modelio sluoksnyje išskirtų praktikų. Todėl be saugumo aspektų išryškinimo programėlės kūrimo procese, organizacijoms yra lygiai tiek pat svarbu sukurti į informacinį bei kibernetinį saugumą orientuotą organizacinę kultūrą. Organizacinės kultūros saugumo aspektai modelyje paremti N. Newton ir kt. 2019 m. išskirtais kritiniais sėkmės faktoriais (KSF). Jie aprašyti 1.6.6 poskyryje, 39 tyrimo puslapyje. KSF atskleidžia kibernetinio ir informacinio saugumo

strategijų ir praktikų pagrindinius elementus, į kuriuos organizacijos turi atkreipti dėmesį, norėdamos užsitikrinti tinkamą savo kibernetinio ir informacinio saugumo lygį.

Saugios mobiliosios aplikacijos kūrimo ir valdymo procesas. Modelio procesinio aprašo dalis suskirsto saugios mobiliosios aplikacijos kūrimą ir valdymą į skirtingas fazes, turinčias įeigas ir siekiamus rezultatais bei kiekvienai fazei aktualius saugumo klausimus. Modelis taip pat turi pagrindinę **įeigą:** programėlės ar jos naujinio poreikį, ir **išeigą**: vartojimui pateikta programėlę ar jos naujinį. Pažymėtina, jog kadangi kūrimo ir valdymo modelis – ciklišas, t.y., vykdomas **iteracijomis**, išeiga automatiškai tampa kito ciklo ar iteracijos užduotimi, pradiniu tašku naujo įeigos poreikio formulavime. Modelis buvo kurtas „Agile“ metodologijos pagrindu, todėl jo skirtingos fazės gali persidengti.

Programėlės kūrimo ciklas arba iteracija susideda iš 6 fazių (septintąja faze prasideda nauja iteracija, ji modelyje pavaizduota tam, kad būtų lengviau suprantamas modelio ciklišKumas bei akcentuojamas nepertraukiamas tobulinimas ankstesnių rezultatų pagrindu): iniciacijos, analizės, dizaino, vykdymo, testavimo ir paleidimo. Kiekvienoje fazėje išryškinami mobiliosios aplikacijos saugumą užtikrinantys žingsniai, kiekviena fazė baigiama pasiekus užsibrėžtą tikslą, tam tikrą rezultatą. Pasiektas rezultatas rodo, jog proceso fazė buvo sėkminga. Tuo tarpu rezultato nepasiekus grįžtama į ankstesnę fazę ir/arba liekama toje pačioje. Pavyzdžiui, jei testuojant programėlę kažkurioje jos dalyje buvo rasta klaidų, grįžtama į programavimo fazę klaidų taisyti, o kitų dalių testavimas gali vykti toliau. Toliau apžvelgiami mobiliosios aplikacijos kūrimo ir valdymo modelio etapai:

1. **Iniciacija.** Šioje fazėje proceso įeiga – programėlės ar jos naujinio poreikis – paverčiama verslo poreikiu: apibrėžiama, kokią programėlę ar jos funkcionalumą tikslinga kurti, atsižvelgiama į verslo tikslus, kuriuos programėlė ar naujinys padėtų pasiekti. Pažymėtina, jog siekiant maksimalaus rezultato, iniciacijos procesas turi būti decentralizuotas, nes skirtingus programėlės elementus dažnai vysto skirtingų lygmenų, specializacijų, kompetencijų darbuotojai, taigi decentralizacija padeda užtikrinti visapusiškesnį programėlės tobulinimą. Fazės rezultatas – funkciniai reikalavimai: aprašymas, kokias pagrindines funkcijas programėlė ar naujinys turi atlikti.
2. **Analizė.** Modelyje ši fazė laikytina esmine, nes joje pradedami analizuoti programėlės ar naujinio saugumo aspektai. Šiame proceso žingsnyje priimami šie sprendimai:
    - Kokie (nauji) duomenys bus naudojami mobiliojoje aplikacijoje;
    - Įvertinamos sąsajos tarp jau turimų ir naujai planuojamų įgyti duomenų, įvertinama, ar nauji duomenys nekels grėsmės jau turimų duomenų saugumui, ar susiejus turimus ir naujus duomenis nebus atskleista jautri informacija apie vartotoją ir pan.;
    - Atliktas kuriamos programėlės ar naujinio rizikos vertinimas. Jis gali būti atliekamas tiek pagal patvirtintą metodologiją, tiek skirtingų kompetencijų darbuotojų minčių lietaus sesijos pavidalu;

- Apsprendžiama, kur ir kaip turi būti saugojami nauji duomenys;
- Atkreipiamas dėmesys į teisinį reglamentavimą: ar naujiems duomenims pagal BDAR ir analogiškus reglamentus netaikomos griežtesnės valdymo ir tvarkymo taisyklės;
- Taip pat šiame žingsnyje įvyksta diskusija tarp verslo poreikiu ir programėlės saugumu suinteresuotų pusių (tai apima tiek tai pačiai organizacijai priklausančių asmenų ar jų grupių, tiek kliento-paslaugos teikėjo santykį), siekiant atrasti tinkamiausią sprendimą ir išlaikyti balansą tarp programėlės saugumo ir konkurencingumo rinkoje.

Ankstyvas saugumo sprendimų įvedimas užtikrina, kad programėlėje saugumui skiriama pakankamai dėmesio, taikomas saugumo *by design* (pagal dizainą, pagal nutylėjimą, nuo pat pradžių) principas, kurio turi būti laikomasi siekiant kurti saugias mobiliąsias aplikacijas. Šios fazės rezultatas – derybų metu tarpusavyje suderinti nefunkciniai reikalavimai (saugumo, UX/UI, greitaveikos ir pan.) bei priimti techniniai saugumo sprendimai (dėl duomenų saugojimo vietos ir pan.)

3. **Dizainas.** Šioje fazėje funkciniai ir nefunkciniai reikalavimai paverčiami konkrečiais techniniais, vizualiniais sprendimais. Dizaino žingsnis apima:
    - Funkcinių reikalavimų architektūros kūrimą: generuojama loginė programėlės ar naujinio funkcijų veikimo schema;
    - Funkcinių reikalavimų specifikacijos kūrimą: aprašoma, kaip turi veikti funkcionalumas;
    - Nefunkcinių reikalavimų specifikacijos kūrimą: aprašoma kokiais požymiais turi pasižymėti nefunkciniai programėlės komponentai, kaip jie turi veikti;
    - Konkrečių techninių sprendimų priėmimą: kokie šifravimo sprendimai bus naudojami, kokios konkrečios duomenų saugyklos, trečiųjų šalių bibliotekos ir kokie įrankiai bus naudojami funkcinių ir nefunkcinių reikalavimų įgyvendinimui ir veikimo užtikrinimui;
    - Vizualinių programėlės ar jos naujinių sprendimų kūrimą.

    Dizaino fazės rezultatas – detalizuota, išsami užduotis programavimui.
4. **Vykdymas.** Šioje fazėje atliekamas programėlės ar jos naujinio programavimas: kuriamas arba koreguojamas (naujinių atveju – jei reikalinga) *backendas*, vėliau atliekamas vartotojo sąsajos, *frontendo* programavimas. Šio žingsnio rezultatas – suprogramuota ir testavimui paruošta programėlė.
5. **Testavimas.** Šioje fazėje skirtingais testavimo metodais tikrinamas programėlės funkcionalumas ir jos saugumas:
    - Automatizuotų testų rašymas, skirtas automatizuotu būdu identifikuoti klaidas programėlės kode, testuoti funkcionalumo veikimo korektiškumą ir pan.;

- Kodo peržiūra (Code review), metodas, kai vienas programuotojas peržiūri kito programuotojo sukurtą kodą;
- QA komandos įsijungimas: rankinis testavimas, įsiskverbimo testai, vykdomi dedikuoto asmens ar komandos;
- Diskusija su kita komanda produkto saugumo klausimais, siekiant kuo objektyviau įvertinti programėlės saugumo sprendimus ir patobulinti juos pagal kolegų pastabas. Diskusija turi būti vykdoma tarp programėlę ar naujinį kuriančios ir prie jos tiesiogiai neprisidedančios komandos, tačiau abi komandos turi turėti panašias kompetencijas.

Pažymėtina, jog testavimo fazę rekomenduotina pradėti dar vykdymo fazėje, lygiagrečiai programėlės funkcijų kūrimui rašant automatinius testus, darant kodo peržiūras ir pan. Fazės rezultatas – paleidimui tinkama programėlė.

6. **Paleidimas.** Programėlės paleidimas gali būti vidinis, kai programėlės progresas pristatomas organizacijos viduje, arba išorinis, kai programėlė patalpinama mobiliųjų aplikacijų parduotuvėje. Pastaruoju atveju priimtina paleidimą vykdyti dviem etapais: pradžioje prieigą prie naujos aplikacijos ar jos naujinio suteikti nedidelei grupei, stebėti programėlės ar naujinio veikimą ir jei programėlė veikia korektiškai – padaryti prieinama visiems. Šios fazės ir visos iteracijos rezultatas sutampa su modelio išeiga, t. y., vartojimui pateikta programėlė ar jos naujinys.

7./1. Naujos iteracijos iniciacija. Kaip jau buvo minėta anksčiau, vienos iteracijos išeiga tampa kitos iteracijos atspirties tašku, nauja iteracija inicijuojama buvusios iteracijos pagrindu. Siekiant kurti konkurencingas ir saugias mobiliąsias aplikacijas svarbu stebėti jos veikimo korektiškumą, reaguoti į vartotojų atsiliepimus.

Modelis tikslingai yra gana abstraktus, jame nenagrinėjami techniniai sprendimai, nes 1) mobiliąsias programėles kuriančios įmonės yra skirtingos ir tikėtina, jog tapačiam rezultatui pasiekti jos pasitelkia skirtingus techninius sprendimus; 2) dėl sparčios technologinės raidos sprendimai, kurie yra aktualūs šiandien, gali būti nebetaikytini po kelerių metų ar net kelių mėnesių. Šiuo atveju abstraktumas gali būti prilygintinas lankstumui, modelį pritaikyti gali kiekviena saugias mobiliąsias aplikacijas kurti norinti organizacija.

Pažymėtina, jog modelis skirtas organizacijoms, kurios savo turimų duomenų saugumą laiko strateginiu ištekliumi, turtu, taigi, ne tik toms, kurios kuria programėles klientams ar vysto savo programėlę kaip pagrindinį verslo vektorių, bet ir organizacijoms, kurių programėlės – tik vienas jų verslo kanalų. Organizacijos, savo kuriamos ar valdomos programėlės procese pasitelkiančios programuotojus ar kitas funkcijas atliekančius žmones ar įmones iš šalies, norėdamos užtikrinti kuriamos aplikacijos saugumą, privalo matyti proceso visumą, nes jos tampa asmens duomenų,

suvedamų į jų programėles, valdytojais. Tačiau verta pastebėti, jog visapusiškas modelio įgyvendinimas gali pareikalauti ir organizacinių pokyčių, o su jais ir daugiau laiko.

IŠVADOS IR REKOMENDACIJOS

1. Asmens duomenų apsaugą reglamentuojantys dokumentai skatina programėlių kūrėjus daugiau dėmesio skirti mobiliųjų aplikacijų saugumui. Nepaisant to, programuotojams dažnai trūksta žinių, galinčių mobiliųjų programėlių saugumą užtikrinti. Iš dalies šią situaciją sąlygoja tai, kad nėra sukurta mobiliųjų aplikacijų saugumo standarto. Tam tikrus reikalavimus taiko mobiliųjų aplikacijų parduotuvės, yra atskirų elementų (pavyzdžiui, API) techniniai reikalavimai, tačiau daugiausiai saugumo reikalavimai pateikiami rekomendacine forma, o kūrėjai ir valdytojai sprendimų ieško arba organizacijos viduje sukurtose geriųjų praktikų aprašuose, arba paieškos sistemose bei knygose.
2. Mobiliųjų aplikacijų saugumas – strateginė organizacijos vertybė. Tik į saugumą žiūrint kaip į strateginę reikšmę turintį klausimą, jam skiriama pakankama svarba ir ištekliai. Kitu atveju dėl programėlės funkcionalumo dažnai daroma nuolaidų aplikacijos saugumo sąskaita. Pastebėtina, jog mobiliųjų aplikacijų rizikos vertinimai sietini su organizacine branda ir nors jų svarbą organizacijos supranta, kol kas jie daugiau atliekami brandžiose, dažnai didelėse organizacijose. Taip pat verta pažymėti, kad dialektikos žingsnius taikančios organizacijos labai palankiai vertina jos atnešamas naudas. Nors 4 ir daugiau dialektikos žingsnių taikė tik 3 organizacijos iš 9 tyrime atstovautų, glaudų bendradarbiavimą tarp skirtingų kompetencijų specialistų skirtingose mobiliosios programėlės fazėse šias organizacijas atstovavę ekspertai įvardino kaip vieną iš organizacijos kuriamo produkto sėkmės faktorių. Ekspertinio interviu metu taip pat išryškėjo, jog saugių mobiliųjų aplikacijų kūrimas atneša ne tik iššūkių, bet taip pat ir galimybių: programėlių kūrėjai teigia, jog jų klientai juos renkasi ne tik dėl siūlomų paslaugų, bet ir dėl mobiliosios aplikacijos ir ją palaikančios sistemos saugumo.
3. Pasiūlytas į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis susideda iš dviejų sluoksnių: organizacinės aplinkos bei procesinio aprašo. Organizacinės aplinkos dalis apima organizacinę kultūrą ir strateginius bei procesinius informacinio ir kibernetinio saugumo aspektus ir sukuria kontekstą programėlės kūrimo procesui. Procesinė dalis suskirsto saugios mobiliosios aplikacijos kūrimą ir valdymą į skirtingas fazes, turinčias įeigas ir siekiamus rezultatais, bei kiekvienai fazei aktualius saugumo klausimus. Saugumo sprendimai pradedami įvedinėti antrojoje, analizės fazėje, siekiant užtikrinti visa apimantį požiūrį į saugumą. Toks požiūris būtinas saugių mobiliųjų aplikacijų kūrime, siekiant vieno pagrindinių kibernetinio ir informacinio saugumo principo – saugumo *by design*. Modelį pritaikyti gali bet kuri organizacija, kurianti ar valdanti mobiliąją aplikaciją.

Rekomendacijos

1. Siekiant išvengti programavimo klaidų tikslingiausia taikyti klaidų prevenciją: testavimą, rizikos vertinimą ir dialektiką. Rekomenduojama taikyti tiek statinį (programėlės kodo) tiek dinaminį (jau veikiančios mobiliosios aplikacijos) testavimą, naudoti ir automatizuoto, ir rankinio testavimo metodus. Siekiant įsivertinti programėlės saugumo sprendimų poreikį ir identifikuoti, kurie jos saugumo aspektai organizacijai potencialiai atneštų daugiausia žalos, svarbu į mobiliosios aplikacijos kūrimo ir valdymo procesą įtraukti rizikos valdymą. Taip pat patartina taikyti dialektikos žingsnius – nuolatinį dialogą tarp skirtingų kompetencijų ir specializacijų darbuotojų organizacijos viduje bei tarp organizacijos ir jos programėlės vartotojų.
2. Į saugumą orientuotą mobiliųjų aplikacijų kūrimo ir valdymo modelį rekomenduojama taikyti visoms organizacijoms, kurioms svarbus jų kuriamų ar valdomų mobiliųjų aplikacijų saugumas. Pažymėtina, jog modelis akcentuoja tiek programėlės kūrimo ir valdymo procesą, tiek organizacinį kontekstą, todėl organizacijoms, siekiančioms visapusiško programėlių saugumo, patartina kreipti pakankamai dėmesio kibernetinio ir informacinio saugumo strategijų ir praktikų diegimui.

LITERATŪRA

Mokslo šaltiniai

1. Abdullah, H. M., & Zeki, A. M. (2014). Frontend and backend web technologies in social networking sites: Facebook as an example. In *2014 3rd international conference on advanced computer science applications and technologies* (pp. 85-89). IEEE. Prieiga per internetą: https://www.researchgate.net/publication/282743092_Frontend_and_Backend_Web_Technologies_in_Social_Networking_Sites_Facebook_as_an_Example.
2. Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M. L., & Stransky, C. (2016). You get where you're looking for: The impact of information sources on code security. In *2016 IEEE Symposium on Security and Privacy (SP)* (pp. 289-305). IEEE. Prieiga per internetą: http://users.umiacs.umd.edu/~mmazurek/papers/sp2016-impact-information-resources.pdf.
3. Almasri, A. K. (2016). A Proposed Hybrid Agile Framework Model for Mobile Applications Development. *International Journal of Software Engineering & Applications (IJSEA)*, *7*(2). Prieiga per internetą: https://www.researchgate.net/publication/300087784_A_Proposed_Hybrid_Agile_Framework_Model_for_Mobile_Applications_Development.
4. Amalfitano, D., Fasolino, A. R., Tramontana, P., & Robbins, B. (2013). Testing android mobile applications: Challenges, strategies, and approaches. In *Advances in Computers* (Nr. 89, pp. 1-52). Elsevier. Prieiga per Science Direct: https://doi.org/10.1016/B978-0-12-408094-2.00001-1.
5. Balebako, R., Marsh, A., Lin, J., Hong, J. I., & Cranor, L. F. (2014). The privacy and security behaviors of smartphone app developers. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.661.4221&rep=rep1&type=pdf.
6. Baležentis A., Žalimaitė M. (2011) Ekspertinių vertinimų taikymas inovacijų plėtros veiksnių analizėje: Lietuvos inovatyvių įmonių vertinimas. Nr. 3 (27), P. 23-31.Prieiga per internetą: http://mts.asu.lt/mtsrbid/article/viewFile/269/298.
7. Barrera, D., Clark, J., McCarney, D., & Van Oorschot, P. C. (2012). Understanding and improving app installation security mechanisms through empirical analysis of android. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices* (pp. 81-92). Prieiga per internetą: http://people.scs.carleton.ca/~paulv/papers/spsm12-barrera.pdf.
8. Binsaleh, M., & Hassan, S. (2011). Systems development methodology for mobile commerce applications: Agile vs. traditional. *International Journal of Online Marketing (IJOM)*, *1*(4), 33-47. Prieiga per internetą: https://www.researchgate.net/publication/220553749_Systems_Development_Methodology_for_Mobile_Commerce_Applications.
9. Campbell, M. (2014). Phone invaders. *New Scientist*, *223*(2977), 32–35. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.1016/S0262-4079(14)61354-3.
10. Clark, J., & Van Oorschot, P. C. (2013). SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *2013 IEEE Symposium on Security and Privacy* (pp. 511-525). IEEE. Prieiga per internetą: https://oaklandsok.github.io/papers/clark2013.pdf.
11. Cohen, D., Lindvall, M., & Costa, P. (2003). Agile software development. *DACS SOAR Report*, *11*, 2003. Prieiga per internetą: http://users.jyu.fi/~mieijala/kandimateriaali/Agile%20software%20development.pdf.

12. de Lima Fontão, A., dos Santos, R. P., & Dias-Neto, A. C. (2015, July). Mobile software ecosystem (mseco): a systematic mapping study. In *2015 IEEE 39th Annual Computer Software and Applications Conference* (Vol. 2, pp. 653-658). IEEE. Prieiga per internetą: https://www.researchgate.net/publication/280014361_Mobile_Software_Ecosystem_MSECO_A_Systematic_Mapping_Study.
13. Dima, A. M., & Maassen, M. A. (2018). From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. *Journal of International Studies*, *11*(2), 315-326. Prieiga per internetą: https://www.researchgate.net/publication/326338469_From_Waterfall_to_Agile_software_Development_models_in_the_IT_sector_2006_to_2018_Impacts_on_company_management.
14. Edison, H., Jabangwe, R., & Duc, A. N. (2018). Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software*, *145*, 98-111. Prieiga per internetą: https://www.researchgate.net/publication/326896132_Software_Engineering_Process_Models_For_Mobile_App_Development_A_Systematic_Literature_Review.
15. Egele, M., Brumley, D., Fratantonio, Y., & Kruegel, C. (2013). An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 73-84). Prieiga per internetą: https://www.cs.umd.edu/class/fall2017/cmsc818O/papers/crypto-misuse-android.pdf.
16. Eling, N., Krasnova, H., Widjaja, T., & Buxmann, P. (2013). Will you accept an app? Empirical investigation of the decisional calculus behind the adoption of applications on Facebook. Prieiga per internetą: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.666.5678&rep=rep1&type=pdf.
17. Fritz, W., Sohn, S., & Seegebarth, B. (2017). Broadening the Perspective on Mobile Marketing: An Introduction. *Psychology & Marketing*, *34*(2), 113–118. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.1002/mar.20978.
18. Goddard, M. (2017). The EU General Data Protection Regulation (GDPR): European regulation that has a global impact. *International Journal of Market Research*, *59*(6), 703-705. Prieiga per Sage: https://journals.sagepub.com/doi/abs/10.2501/IJMR-2017-050?journalCode=mrea.
19. Hansson, S. O., & Aven, T. (2014). Is risk analysis scientific?. *Risk analysis*, *34*(7), 1173-1183. Prieiga per internetą: https://www.researchgate.net/publication/263055065_Is_Risk_Analysis_Scientific.
20. Hicks, J. L., Uchida, T. K., Seth, A., Rajagopal, A., & Delp, S. L. (2015). Is my model good enough? Best practices for verification and validation of musculoskeletal models and simulations of movement. *Journal of biomechanical engineering*, *137*(2). Prieiga per internetą: https://www.researchgate.net/publication/269187913_Is_My_Model_Good_Enough_Best_Practices_for_Verification_and_Validation_of_Musculoskeletal_Models_and_Simulations_of_Movement.
21. Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120-127. Prieiga per internetą: http://sunnyday.mit.edu/16.355/highsmith-agile.pdf.
22. Hubbard, J., Weimer, K., & Chen, Y. (2014). A study of SSL proxy attacks on Android and iOS mobile applications. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)* (pp. 86-91). IEEE. Prieiga per internetą:

https://www.researchgate.net/publication/269291151_A_study_of_SSL_Proxy_attacks_on_Android_and_iOS_mobile_applications.

23. Yusop, N., Kamalrudin, M., Yusof, M. M., & Sidek, S. (2016). Meeting Real Challenges in Eliciting Security Attributes for Mobile Application Development. *Journal of Korean Society for Internet Information*, *17*(5), 25–32. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.7472/jksii.2016.17.5.25.
24. Jelassi, T., & Enders, A. (2005). *Strategies for e-business: creating value through electronic and mobile commerce: concepts and cases*. Pearson Education.
25. Jha, N., & Mahmoud, A. (2019). Mining non-functional requirements from app store reviews. *Empirical Software Engineering*, *24*(6), 3659-3695. Prieiga per internetą: https://www.researchgate.net/publication/333662582_Mining_non-functional_requirements_from_App_store_reviews.
26. Kellner, A., Horlboge, M., Rieck, K., & Wressnegger, C. (2019). False sense of security: A study on the effectivity of jailbreak detection in banking apps. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 1-14). IEEE. Prieiga per internetą: https://www.sec.cs.tu-bs.de/pubs/2019-eurosp.pdf.
27. Libby, R., & Blashfield, R. K. (1978). Performance of a composite as a function of the number of judges. *Organizational Behavior and Human Performance*, *21*(2), 121-129.
28. Momen, N., Hatamian, M., & Fritsch, L. (2019). Did App privacy improve after the GDPR?. *IEEE Security & Privacy*, *17*(6), 10-20. Prieiga per internetą: https://www.researchgate.net/publication/336995204_Did_App_Privacy_Improve_After_the_GDPR.
29. Nigam, P. V., & Amirtharaj, P. (2018). Role of Mobile Applications Quality and Smartphones Attributes in Online Shopping: An Insight. *Jaipuria International Journal of Management Research, 4*(1), 28-35. Prieiga per internetą: https://www.researchgate.net/publication/325992310_Role_of_Mobile_Applications_Quality_and_Smartphones_Attributes_in_Online_ShoppingAn_Insight.
30. Norris, D. F., & Reddick, C. G. (2013). Local E-Government in the United States: Transformation or Incremental Change? *Public Administration Review*, *73*(1), 165–175. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.1111/j.1540-6210.2012.02647.x
31. Nunan, D., & Yenicioglu, B. (2013). Informed, uninformed and participative consent in social media research. *International Journal of Market Research*, *55*(6), 791–808. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.2501/IJMR-2013-067.
32. Pandey, M., Litoriya, R., & Pandey, P. (2019). Novel approach for mobile based app development incorporating MAAF. *Wireless Personal Communications*, *107*(4), 1687-1708. Prieiga per internetą: https://www.researchgate.net/publication/332302216_Novel_Approach_for_Mobile_Based_App_Development_Incorporating_MAAF.
33. Rakestraw, T. L., Eunni, R. V., & Kasuganti, R. R. (2013). The mobile apps industry: A case study. *Journal of Business Cases and Applications*, *9*, 1. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/dow-load?doi=10.1.1.645.3921&rep=rep1&type=pdf.
34. Ravitch, T., Creswick, E. R., Tomb, A., Foltzer, A., Elliott, T., & Casburn, L. (2014). Multi-app security analysis with fuse: Statically detecting android app collusion. In *Proceedings of the 4th Program Protection and Reverse Engineering Workshop* (pp. 1-10). Prieiga per internetą: http://galois.com/wp-content/uploads/2015/03/FUSE-pprew14.pdf.

35. Rupnik, R. & Krisper, M. (2009). Mobile Applications: A New Application Model in Information Systems. Prieiga per internetą: https://www.researchgate.net/publication/242665689_MOBILE_APPLICATIONS_A_NEW_APPLICATION_MODEL_IN_INFORMATION_SYSTEMS.
36. Salini, P., & Kanmani, S. (2016). Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems. *International Journal of Information Security*, *15*(3), 319–334. Prieiga per Ebscohost: https://doi-org.skaitykla.mruni.eu/10.1007/s10207-015-0305-x.
37. Schairer, C. E., Rubanovich, C. K., & Bloss, C. S. (2018). How could commercial terms of use and privacy policies undermine informed consent in the age of mobile health?. *AMA journal of ethics*, *20*(9), 864-872. Prieiga per internetą: https://www.researchgate.net/publication/327832902_How_Could_Commercial_Terms_of_Use_and_Privacy_Policies_Undermine_Informed_Consent_in_the_Age_of_Mobile_Health.
38. Statlnionytė, L. (2013). Reikalavimų programinei įrangai valdymas. Magistro baigiamasis darbas. Prieiga per internetą: https://vb.vgtu.lt/object/elaba:1753814/index.html.
39. Šuminas, A., & Aleksandravičius, A. (2013). Mobiliosios programėlės rinkiminėje komunikacijoje. *Parliamentary Studies*, (15), 39-65. Prieiga per internetą: https://journals.lnb.lt/parliamentary-studies/article/view/234.
40. Taneja, S., & Goel, A. (2015). A Mobile App Architecture for Student Information System. *Int. J. Web Appl.*, *7*(2), 56-63. Prieiga per internetą: http://www.dline.info/ijwa/fulltext/v7n2/v7n2_2.pdf.
41. Treacy, C., & McCaffery, F. (2016). Data security overview for medical mobile apps assuring the confidentiality, integrity and availability of data in transmission. *International Journal on Advances in Security*, *9*(3 & 4), 146-157. Prieiga per internetą: https://eprints.dkit.ie/555/.
42. Truong, N. B., Sun, K., Lee, G. M., & Guo, Y. (2019). Gdpr-compliant personal data management: A blockchain-based solution. *IEEE Transactions on Information Forensics and Security*, *15*, 1746-1761. Prieiga per internetą: https://www.researchgate.net/publication/336367690_GDPR-Compliant_Personal_Data_Management_A_Blockchain-based_Solution.
43. Utz, C., Degeling, M., Fahl, S., Schaub, F., & Holz, T. (2019). (Un)informed consent: Studying gdpr consent notices in the field. In *Proceedings of the 2019 acm sigsac conference on computer and communications security* (pp. 973-990). Prieiga per internetą: https://doi.org/10.1145/3319535.3354212.
44. Van Casteren, W. (2017). The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. *Research Gate*, 1-6. Prieiga per internetą: https://www.researchgate.net/publication/313768860_The_Waterfall_Model_and_the_Agile_Methodologies_A_comparison_by_project_characteristics_-_short.
45. Wang JA, Xia M, Zhang F. Metrics for information security vulnerabilities. *Proceedings of Intellect base International Consortium* 2009;(1)284-294. Prieiga per internetą: http://www.intellectbase.org/e_publications/jagr/JAGR_Volume_1_Issue_1.pdf#page=55.
46. Weir, C., Awais, R., Noble, J. I'd Like to Have an Argument, Please: Using Dialectic for Effective App Security. *EuroUSEC '17*. Internet Society, 2017. Prieiga per internetą: http://dx.doi.org/10.14722/eurousec.2017.23002
47. Weir, C., Hermann, B., & Fahl, S. (2020). From needs to actions to secure apps? the effect of requirements and developer practices on app security. In *29th {USENIX} Security Symposium*

({USENIX} Security 20) (pp. 289-305). Prieiga per internetą: https://www.usenix.org/system/files/sec20fall_weir_prepub.pdf.

48. Welch, C. (2014). Another bogus Android app briefly tops Google Play charts. *The Verge.* Prieiga per internetą: https://www.theverge.com/2014/4/8/5594196/another-bogus-android-app-briefly-tops-google-play-charts.
49. Woods, D., Agrafiotis, I., Nurse, J. R., & Creese, S. (2017). Mapping the coverage of security controls in cyber insurance proposal forms. *Journal of Internet Services and Applications*, *8*(1), 1-13. https://doi.org/10.1186/s13174-017-0059-y.
50. Wottrich, V. M., Reijmersdal, E. A., & Smit, E. G. (2019). App Users Unwittingly in the Spotlight: A Model of Privacy Protection in Mobile Apps. *Journal of Consumer Affairs*, *53*(3), 1056–1083. Prieiga per internetą: https://doi-org.skaitykla.mruni.eu/10.1111/joca.12218.
51. Zeidler, C., Kittl, C., & Petrovic, O. (2008). An integrated product development process for mobile software. *International Journal of Mobile Communications*, *6*(3), 345-356. Prieiga per interrnetą: https://www.researchgate.net/publication/220474759_An_integrated_product_development_process_for_mobile_software.
52. Žydžiūnaitė, Vilma; & Sabaliauskas, Stanislav. *Kokybiniai tyrimai: Principai ir metodai*. Leidykla VAGA, 2017.

Teisės aktai

1. 2016 m. balandžio 27 d. Europos Parlamento ir Tarybos reglamentas (ES) 2016/679 dėl fizinių asmenų apsaugos tvarkant asmens duomenis ir dėl laisvo tokių duomenų judėjimo ir kuriuo panaikinama Direktyva 95/46/EB (Bendrasis duomenų apsaugos reglamentas). Prieiga per internetą: https://eur-lex.europa.eu/legal-content/LT/TXT/?uri=celex%3A32016R0679.
2. 2016 m. liepos 6 d. Europos Parlamento ir Tarybos direktyva (ES) 2016/1148 dėl priemonių aukštam bendram tinklų ir informacinių sistemų saugumo lygiui visoje Sąjungoje užtikrinti. Prieiga per internetą: https://eur-lex.europa.eu/legal-content/LT/TXT/?uri=CELEX%3A32016L1148.

Kiti šaltiniai

1. Appinventive. Explained: Mobile App Architecture – The Basis of App Ecosystem (2020). Prieiga per internetą: https://appinventiv.com/blog/mobile-app-architecture-explained/ [žiūrėta 2021 m. vasarį].
2. App Store Review Guidlines. Prieiga per internetą: https://developer.apple.com/app-store/review/guidelines/ [žiūrėta 2021 m. vasarį].
3. Cambridge Dictionary. Mobiliosios aplikacijos apibrėžimas. Prieiga per internetą: https://dictionary.cambridge.org/dictionary/english/mobile-application.
4. Carielli, S., DeMartine, A., Bongarzone, M., Dostie, P. (2020). The State Of Application Security, 2020. *Forrester*. Prieiga per internetą: https://reprints2.forrester.com/#/assets/2/982/RES159057/report [žiūrėta 2021 m. vasarį].
5. CIS 20 kibernetinio saugumo taisyklių (The 20 CIS Controls & Resources). Prieiga per internetą: https://www.cisecurity.org/controls/cis-controls-list/ [žiūrėta 2021 m. vasarį].
6. Doffman, Z. (2019). Google Confirms Play Store Security Threat: Here's The Fix—But Does It Make You Safer? *Forbes*. Prieiga per internetą:

https://www.forbes.com/sites/zakdoffman/2019/11/10/google-confirms-play-store-security-threat-heres-the-fixbut-does-it-make-you-safer/?sh=6ca633e4337f [žiūrėta 2021 m. vasarį].

7. Dogtiev, A. (2021). App Stores List (2020). *Business of Apps*. Prieiga per internetą: https://www.businessofapps.com/guide/app-stores-list/ [žiūrėta 2021 m. vasarį].
8. Google Play Developer Policy Center. Privacy, Security, and Deception. Prieiga per internetą: https://play.google.com/about/privacy-security-deception/user-data/ [žiūrėta 2021 m. vasarį].
9. May Ying Tee, Marting Zhang (2018) Hidden App Malware Found on Google Play. *Symantec Enterprise Blogs / Threat Intelligence.* Prieiga per internetą: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/hidden-app-malware-google-play?om_ext_cid=biz_social_NAM_linkedin_Asset%20Type%20%20-%20Blog,Campaign%20-%20Intelligent%20Mobile [žiūrėta 2021 m. vasarį].
10. NowSecure 2019 IDC MarketScape for Mobile Application Security Testing (MAST). Analyst report. Prieiga per internetą: https://www.nowsecure.com/resource/2019-idc-marketscape-for-mobile-application-security-testing-mast/. [žiūrėta 2021 m. vasarį].
11. OWASP Mobiliųjų aplikacijų ir įrenginių pažeidžiamumų Top 10. Prieiga per internetą: https://owasp.org/www-project-mobile-top-10/ [žiūrėta 2021 m. vasarį].
12. OWASP rizikų vertinimo metodologija. Prieiga per internetą: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology [žiūrėta 2021 m. vasarį].
13. Positive technologies. Vulnerabilities and threats in mobile applications, 2019. Prieiga per internetą: https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/ [žiūrėta 2021 m. vasarį].
14. Statista tyrimų skyrius. Mobiliųjų programėlių skaičius parduotuvėse 2020 m. IV ketvirtį. Prieiga per Statista svetainę: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ [žiūrėta 2021 m. vasarį].
15. Veracode (2021) State of Software Security volume 11. Prieiga per internetą: https://www.veracode.com/state-of-software-security-report. [žiūrėta 2021 m. vasarį].

Markauskienė, G. O. (2021) *Saugių mobiliųjų aplikacijų kūrimo ir valdymo ypatumai: galimybės ir iššūkiai* (magistro baigiamasis darbas). Vilnius: Mykolo Romerio universitetas

ANOTACIJA

Magistro baigiamajame darbe išanalizuoti saugių mobiliųjų aplikacijų kūrimo ir valdymo iššūkiai, jų sprendimo būdai, pasiūlytas unifikuotas į saugumą orientuotas mobiliųjų programėlių kūrimo ir valdymo modelis. Pirmojoje darbo dalyje išnagrinėta mobiliųjų aplikacijų specifika, mobiliųjų programėlių saugumui taikomi reikalavimai, dažniausiai iškylantys iššūkiai, jų priežastys ir sprendimo būdai. Vėliau išanalizuoti mobiliųjų programėlių kūrimo ir valdymo modeliai. Antroje darbo dalyje siekiant įvertinti mobiliųjų aplikacijų problematiką, su kuria susiduria programėlių kūrėjai ir valdytojai, buvo atliktas tyrimas. Struktūruoto interviu metodu buvo apklausti 9 ekspertai. Trečiojoje darbo dalyje pasitelkus ekspertų žinias ir pirmoje darbo dalyje išskirtus svarbiausius programėlių saugumo aspektus bei remiantis modelių analizės išvadomis buvo sukonstruotas ir pasiūlytas modelis, padėsiantis organizacijoms susifokusuoti į programėlių saugumą pradinėse jų kūrimo fazėse.

Raktiniai žodžiai: mobilioji aplikacija, saugi mobilioji aplikacija, mobilioji programėlė, kibernetinis saugumas, mobiliųjų aplikacijų saugumas, mobiliųjų aplikacijų kūrimo modelis.

Markauskienė, G. O. (2021). *Particularities of Developing and Managing Secure Mobile Applications: Opportunities and Challenges* (master thesis). Vilnius: Mykolas Romeris University

ANNOTATION

This master thesis analyses challenges of developing and managing secure mobile applications, solutions to those challenges. Thesis also suggest a unified and security-oriented model for creating and managing mobile applications. The first part of the thesis examines the specifics of mobile applications, studies requirements for the security of mobile applications, review the most common challenges in creating and developing secure mobile applications, the causes and the ways these challenges can be addressed. Models for creating and managing mobile applications are analysed. In order to assess the security challenges that arise in everyday life of mobile apps' developers and managers, a study was conducted in the second part of the thesis. 9 experts were interviewed using the structured interview method. The third part of the thesis includes the creation of a model, that would help organizations to create and manage secure mobile applications, to focus on different security aspects throughout the creation and management cycle. The model was created combining expert knowledge from the study and the most important security features, that are highlighted in the first part of the thesis.

Keywords: mobile application, mobile app, secure mobile application, security, cybersecurity, security-oriented model, creating and managing mobile apps.

Markauskienė, G. O. (2021) *Saugių mobiliųjų aplikacijų kūrimo ir valdymo ypatumai: galimybės ir iššūkiai* (magistro baigiamasis darbas). Vilnius: Mykolo Romerio universitetas

SANTRAUKA

Per pastarąjį dešimtmetį itin ištobulinti ir išpopuliarėję mobilieji įrenginiai pakeitė pasaulį. Iš vien susisiekimui skirto aparato mobilusis įrenginys virto į daugiafunkcinį ir vis daugiau žmogaus gyvenimo sričių apimantį daiktą, be kurio dažnas žmogus neįsivaizduotų savo dienos. Mobiliosios aplikacijos taip pat tampa vis svarbesniu pardavimų ir informaciniu kanalu. Tačiau vis dažniau pastebima mobiliųjų aplikacijų saugumo problema. Pastebima, jog didžioji dalis mobiliųjų programėlių nėra visiškai saugios. Mokslininkų vertinimu ši situacija susiklostė dėl skirtingo ir subjektyvaus mobiliųjų aplikacijų saugumo vertinimo.

Siekiant išsiaiškinti šios problemos priežastis darbe apžvelgiama mokslinė literatūra. Mokslinės literatūros analizės, normatyvinė analizės ir turinio analizės metodais siekiama išnagrinėti teorinius saugių mobiliųjų aplikacijų kūrimo ir valdymo aspektus. Normatyvinės analizės metodu nagrinėjami teisiniai reikalavimai saugioms mobiliosioms aplikacijoms. Turinio analizė pasitelkta nagrinėjant mokslininkų sukurtus mobiliųjų aplikacijų kūrimo ir valdymo modelius. Struktūruoto interviu metodas pasitelktas tyrimui, kurio tikslas – įvertinti praktinius saugių mobiliųjų programėlių kūrimo ir valdymo aspektus, jų problematiką. Pastebėta, kad mobiliosioms programėlėms taikomi saugumo reikalavimai – daugiau rekomendacinio pobūdžio, mažai detalizuoti, todėl didžioji dalis atsakomybės dėl mobiliųjų aplikacijų saugumo tenka mobiliąsias aplikacijas kuriantiems programuotojams. Darbe atskleista, kad programuotojams dažnai trūksta žinių apie programėlių saugumo užtikrinimą.

Siekiant įvertinti saugių mobiliųjų aplikacijų kūrėjams ir valdytojams kylančius mobiliųjų aplikacijų kūrimo ir valdymo iššūkius bei kaip jie sprendžiami praktikoje, buvo atliekamas tyrimas. Struktūruoto interviu metodu buvo apklausiami 9 ekspertai, turintys nemažesnę nei 5 metų patirtį mobiliųjų programėlių kūrime. Tyrimo metu paaiškėjo, jog mobiliųjų aplikacijų saugumas – strateginis programėlių kūrėjų ir valdytojų klausimas, apimantis ne tik programėlės techninius sprendimus, bet ir organizacijos kultūrą, požiūrį į klientus, organizacijos kibernetinio saugumo lygį ir pan. Taip pat pastebėta, jog saugi mobilioji aplikacija – ne tik iššūkis, bet ir galimybė, nes vartotojai tampa vis sąmoningesni ir renkasi tas programėles, kurie atitinka jų lūkesčius saugumui.

Trečiojoje tyrimo dalyje buvo sukurtas unifikuotas į saugumą orientuotas mobiliųjų aplikacijų kūrimo ir valdymo modelis. Siekiant kuo patogesnio modelio panaudojimo, moksliniame darbe sukurtas modelis buvo kuriamas pagal „Agile“ metodologijos principus. Programėlių saugumui skiriamas dėmesys pradinėse kūrimo fazėse, todėl juo besivadovaujančios organizacijos galėtų lengviau sukurti programėlę, saugią *by design*.

Markauskienė, G. O. (2021). *Particularities of Developing and Managing Secure Mobile Applications: Opportunities and Challenges* (Master‘s thesis). Vilnius: Mykolas Romeris University

SUMMARY

Over the past decade mobile devices have changed the world. From a mere tool for communication, the mobile device has evolved into a multifunctional instrument that blend into more and more areas of human life, and without which many people would not be able to imagine their day. Mobile applications are also becoming an increasingly important channel for sales and information. However, the security problems of mobile applications are becoming increasingly evident. It is noticeable that most mobile apps are not completely secure. According to the researchers, this situation was caused by a different and subjective assessment of the security of mobile applications.

In order to find out the causes of this problem, the scientific literature is reviewed in the work. The methods of scientific literature analysis, normative analysis and content analysis aim to examine the theoretical aspects of the development and management of secure mobile applications. The method of normative analysis examines the legal requirements for secure mobile applications. Content analysis was used to analyze the models of mobile application development and management developed by researchers. The method of structured interviews was used for research, the aim of which is to assess the practical aspects of developing and managing secure mobile apps, their problems. It has been observed that the security requirements for mobile apps are more of a recommendatory nature, with little detail, so most of the responsibility for the security of mobile applications lies with the developers of mobile applications. The paper reveals that programmers often lack knowledge about how to ensure the security of mobile apps.

A study was conducted to assess the challenges for developers and managers of secure mobile applications and how they are addressed in practice. 9 experts with at least 5 years of experience in mobile app development were interviewed using the structured interview method. The research revealed that the security of mobile applications is a strategic issue for app developers and managers, covering not only the technical solutions of the app, but also the organization's culture, attitude towards customers, the level of cyber security of the organization, etc. It is important to note that a secure mobile application is not only a challenge but also an opportunity as users become more aware and choose those applications that meet their security expectations.

In the third part of the study, a unified security-oriented model for the development and management of mobile applications was developed. In order for the model to be used as conveniently as

possible, it was developed in accordance with the principles of the Agile methodology. This module focuses security decisions in the early stages of development, making it easier for organizations that follow it to create a mobile application that is secure by design.

1 PRIEDAS. STRUKTŪRUOTO INTERVIU KLAUSIMYNAS

Sveiki,

esu Mykolo Romerio universiteto Kibernetinio saugumo valdymo magistro programos antrojo kurso studentė. Šiuo metu atlieku kokybinį tyrimą „Saugių mobiliųjų aplikacijų kūrimo ir valdymo iššūkiai“. Šio struktūruoto interviu metu siekiu suprasti, su kokiais programėlių saugumo iššūkiais mobiliųjų aplikacijų kūrėjai susiduria skirtingose kūrimo fazėse.

Mobilioji aplikacija, arba mobilioji programėlė, darbe nagrinėjama kaip programinės įrangos paketas, sukurtas mobiliajam įrenginiui ir teikiantis specifinę paslaugą. Šis terminas apima visus tris mobiliosios aplikacijos architektūros lygmenis – atvaizdavimo (*presentation*), verslo logikos (*business*), bei duomenų (*data*) – bei sąsajų tarp jų. Darbe fokusuojamasi į vietines (*native*) ir hibridinės aplikacijas.

Šio tyrimo metu surinkti duomenys bus pateikti apibendrinta forma, jūsų pateikti atsakymai nebus viešai skelbiami. Šis interviu įrašinėjamas kompiuteriu ir telefone esančiu diktofonu, kad būtų išvengta galimo duomenų praradimo. Įrašai ir juose užfiksuota informacija bus naudojami tik mokslinio darbo tikslais, pateikus darbą įrašai bus negrįžtamai ištrinti.

1. Kiek laiko dirbate mobiliųjų aplikacijų kūrimo srityje?

2. Ar programėles kuriate vienas, kaip laisvai kuriamas darbuotojas (freelancer’is), ar įmonėje, komandoje kartu su kitais programuotojais, projektų vadovais, dizaineriais ir kt.?

**

[3 ir 4 jei į 2 klausimą atsakė “komandoje”]

3. Kokios jūsų pareigos ir funkcijos jūsų darbovietėje?

4. Kokios kitos pareigybės/funkcijos įsteigtos jūsų darbovietėje?

***

5. Kokia apimtimi prisidedate prie programėlių kūrimo (pvz., dokumentacijos rengimas, *front-end* programavimas, *back-end* programavimas).

6. Ar kuriate/valdote daug programėlių, ar kuriate ir palaikote vieną?

7. Trumpai nusakykite mobiliosios programėlės kūrimo ir valdymo procesą savo įmonėje/veikloje.

8. Kuriame programėlės kūrimo ir valdymo proceso etape paprastai pradedate įkorporuoti programėlės saugumo sprendimus? Kokio tipo sprendimai tai paprastai būna?

9. Ar atsižvelgiate į tai, kokie vartotojų asmens duomenys apdorojami jūsų programėlėse? Ar tai atsispindi sprendimuose?

10. Kokių sprendimų imatės, kad jūsų programėlės būtų saugios?

11. Kas kiek laiko paprastai išleidžiate programėlių naujinius (jei kuriate naujinius daugiau nei vienai programėlei, atsakykite remdamiesi ta, kuriai naujinius išleidžiate dažniausiai)?

12. Dėl kokių priežasčių dažniausiai atnaujinate programėles?

13. Koks jūsų programėlių testavimo procesas? Galbūt pasitelkiate automatizuotus testavimo įrankius (SAST, DAST, IAST)? Ar testuojate jau paleistas programėles?

14. Ar atliekate programėlės ir atskirų jos dalių/apdorojamų duomenų rizikos vertinimus? Jei taip, trumpai papasakokite apie vertinimo procesą: kokiame programėlės kūrimo etape jį atliekate, ką paprastai vertinate, kokią reikšmę šis vertinimas turi programėlės kūrimo procese.

15. Ar egzistuoja kažkokie programėlių saugumo standartai? Jei ne, kokia informacija remiatės kurdami ar valdydami programėles?

16. Kokios saugumo spragos, jūsų nuomone, dažniausiai atsiranda? Apžvelkite tiek dėl programuotojo kaltės, tiek dėl paveldėtų pažeidžiamumų galinčias atsirasti spragas.

17. Kaip šią situaciją būtų galima pagerinti?

18. Programėlės saugumas – vienas iš nefunkcinių reikalavimų, daugelio ekspertų įvardinamas kaip svarbiausias. Ar sutinkate?

19. Ar rinka keičiasi? Ar vartotojai tampa sąmoningesni savo asmens duomenų klausimu?

20. Ar manote, kad programėlės saugumas – strateginis (apimantis organizacinę kultūrą, išteklių planavimą ir pan.), ar tehcninis (apimantis konkrečius techninius saugumo sprendimus) programėlę(-es) kuriančios ir/ar valdančios įmonės klausimas?