VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Skaitmeninių įkalčių surinkimo debesų kompiuterijos sistemose įrankis

Digital Evidence Collection Tool in Cloud Computing Systems

Bakalauro baigiamasis darbas

Atliko:	Audrius Žemys	(parašas)
Darbo vadovas:	doc. dr. Linas Bukauskas	(parašas)
Darbo recenzentas:	lekt. Eduardas Kutka	(parašas)

Vilnius – 2020

Santrauka

Šiame darbe nagrinėtas skaitmeninės ekspertizės proceso taikymas debesų kompiuterijos sistemoms. Pristačius debesų kompiuterijos paslaugų modelius, charakteristiką bei skaitmeninės ekspertizės proceso taikymą tradicinėms IT sistemoms, apžvelgti kylantys iššūkiai, taikant skaitmeninės ekspertizės procesą debesų kompiuterijos sistemoms.

Nustatyta, kad tradicinis skaitmeninės ekspertizės procesas nėra tinkamas debesų kompiuterijos sistemoms. Apibendrinus problemas, pasiūlyti skaitmeninės ekspertizės proceso patobulinimai, jį taikant debesų kompiuterijos sistemoms bei atliktas proceso eksperimentas, naudojant sukurtą įrankio taikymą modeliuojamoje aplinkoje.

Raktiniai žodžiai: debesų kompiuterija, debesis, skaitmeninė ekspertizė, skaitmeninis tyrimas, skaitmeninė ekspertizė debesyje

Summary

In this paper, digital forensics process application to cloud computing systems was researched. After presenting cloud computing service models, characteristics and digital forensics process application to traditional IT systems, emerging challenges for digital forensics process application to cloud computing systems have been reviewed.

It was determined that the traditional digital forensics process is not applicable for cloud computing systems. After reviewing the emerged challenges, digital forensics process improvements have been proposed in order to make it applicable for cloud computing systems. An experiment was performed to evaluate the improvements, using a created tool in a simulated environment.

Keywords: cloud computing, cloud, digital forensics, digital investigation, cloud forensics, digital forensics in cloud

TURINYS

Įvadas

Debesų kompiuterijos atsiradimą galime vadinti viena iš ryškiausių IT revoliucijų. Debesies sistemos suteikė galimybę bet kuriuo metu ir iš bet kurios vietos, naudojantis interneto ryšį turinčias priemones, pasiekti norimas paslaugas. IT įmonės savo brangią ir statišką infrastruktūrą migruoja į debesį dėl aibės gaunamų privalumų — nuo mažesnių kaštų iki neįtikėtinų personalizacijos galimybių. Tokie pasaulio grandai kaip Microsoft, Amazon, Google, IBM bei Oracle aktyviai lenktyniauja dėl lyderės pozicijos pelningoje debesų rinkoje, 2018 metais pasiekusią 182,4 mlrd. dolerių apyvartą. Sparčiai augantis naujų produktų poreikis, itin arši konkurencija ir tobulėjančios technologijos yra vienos iš esminių priežasčių, kodėl debesų kompiuterijos rinkai 2022 metais prognozuojama daugiau nei 330 mlrd. dolerių apyvarta [**Cos19**].

Be debesies siūlomų privalumų esama ir rizikos. IT įmonės, pirkdamos debesų kompiuterijos paslaugas, patiki savo infrastruktūrą debesies teikėjui, tikėdamiesi, kad šis užtikrins jų verslo saugumą. Tuo tarpu, kibernetiniai nusikaltėliai debesį mato kaip itin patrauklų taikinį pelningai atakai. Jeigu įmonės, turinčios nuosavą infrastruktūrą, tėra vienas taikinys, tai debesies paslaugos teikėjas yra keleto taikinių aibė, tad viena rasta spraga atveria kelią į daug aukų.

Jau 2011 metais buvo prognozuojami 27 mlrd. svarų nuostoliai Didžiosios Britanijos ekonomikai [**GSG12**] dėl kibernetinių atakų, nukreiptų į debesį. Įspūdingu tempu auganti debesų kompiuterijos rinka nėra nepažeidžiama, tą liudija sėkmingai įvykdytos atakos prieš Amazon infrastruktūrą ir Google produktą Gmail [**GSG12**].

Būtina ne tik gerinti debesies saugumą, bet ir tinkamai pasiruošti įvykusių nusikaltimų tyrimams. Skaitmeninės teismo ekspertizės sritis debesų kompiuterijoje susiduria su itin opiomis problemomis — nuo kylančių techninių kliūčių iki teisinio reguliavimo painiavos. Toks žymus kibernetinio saugumo srities atsilikimas nuo itin sparčiai augančios ir vertingos debesų kompiuterijos sferos yra kritiškai pavojingas.

Tradiciniai skaitmeninės ekspertizės proceso principai, praktikos bei įrankiai dažniausiai yra orientuoti į statinį tyrimą [**Cos19**]. Tyrėjai negali aklai pasikliauti debesies paslaugos teikėjo pasiruošimu ir patys privalo turėti būtinus įrankius ir žinias apie debesies sistemų architektūrą bei veikimo principus, taip pat tyrėjams būtinas aukšto lygio procesas, kuriuo jie galėtų vadovautis. Be aiškios krypties tyrimas gali būti labai chaotiškas, užtrukti labai ilgai bei, galų gale, neatnešti jokios naudos - įkalčių.

Šiame darbe apžvelgiami populiariausi debesų kompiuterijos sistemų paslaugų tipai bei jų charakteristikos, pristatomas vienas iš pagrindinių skaitmeninės ekspertizės procesų. Apibrėžus šias tyrimo sritis, išanalizuotas jų suderinamumas bei identifikuotos skaitmeninės ekspertizės proceso problemos debesų kompiuterijos sistemose ir tiriama, kokie proceso patobulinimai reikalingi, kad skaitmeninės ekspertizės procesas būtų tinkamas debesų kompiuterijos sistemoms. Pasiūlytų proceso patobulinimų veikimas patikrintas atliekant eksperimentą, naudojant sukurtą įkalčių surinkimo įrankį modeliuojamoje aplinkoje.

Šio **darbo tikslas** — sukurti skaitmeninės teismo ekspertizės metodologiją, kuri būtų tinkama debesies tipo paslaugoms bei suprojektuoti įrankio, skirto skaitmeniniams įkalčiams debesų kompiuterijos sistemose surinkti, prototipą.

Darbe keliami uždaviniai:

1. Atlikti egzistuojančių skaitmeninės teismo ekspertizės metodologijų apžvalgą;
2. Įvertinti jų tinkamumą debesų kompiuterijos sistemoms;
3. Sukurti skaitmeninės teismo ekspertizės metodologiją, tinkamą debesų kompiuterijos sistemoms;
4. Sukurti įrankį, naudingą tyrimui debesies sistemose;
5. Pagrįsti metodologijos bei įrankio veiksmingumą, ją pritaikant eksperimente modeliuojamoje aplinkoje;
6. Įvertinti eksperimento rezultatus bei įvardinti plėtros galimybes;

1. Debesų kompiuterija

1.1. Charakteristika

Debesų kompiuterija suteikia galimybę naudotis kompiuterine ir programine įranga internete ir mokėti tik už tai, kiek teikiama paslauga buvo pasinaudota [**Peč11**]. Ši XXI a. pradžioje sparčiai plisti pradėjusi technologija vis dažniau naudojama ne tik fizinių asmenų, bet ir verslo organizacijų poreikiams. Debesų kompiuterijai apibūdinti naudojamos šios charakteristikos [**MG11a**]:

- **Savarankiškai ir pagal poreikį (on-demand):** Vartotojas gali pagal savo poreikius apsibrėžti reikalingą kompiuterinių ir atminties resursų kiekį savarankiškai bei automatiškai, be paslaugos teikėjo darbuotojo pagalbos.
- **Plačios prieigos tinkle galimybės:** Paslaugos prieinamos naudojantis interneto prieigą turinčiais įrenginiais, pvz.: išmaniaisiais telefonais, planšetiniais kompiuteriais, asmeniniais bei nešiojamaisiais kompiuteriais.
- **Išteklių sutelkimas:** Paslaugos teikėjo ištekliai yra sutelkti teikti paslaugas skirtingiems vartotojams vienu metu, naudojant skirtingus fizinius bei virtualius resursus, dinamiškai priskirtus arba perskirstytus pagal kliento poreikius. Klientas iš esmės neturi jokios kontrolės ar žinių apie tikslią teikiamų resursų lokaciją, tačiau turi galimybę identifikuoti jų vietą aukštu abstrakcijos lygiu (valstybė, valstija, duomenų centras). Išteklių pvz.: duomenų saugyklos, kompiuterinio apdorojimo bei skaičiavimo galimybės, atmintis bei ryšio pralaidumas.
- **Lankstumas:** Kai kuriais atvejais ištekliai gali būti priskirti bei atleisti lanksčiai ir automatiškai, sparčiai juos praplečiant arba sumažinant pagal poreikį. Klientui išteklių prieinamumas ir kiekis gali atrodyti neriboti bei išskiriami bet kokiam laikotarpiui.
- **Paslaugos pamatuojamumas:** Debesies sistemos automatiškai valdo ir optimizuoja išteklius, išmatuojant jų kiekį pagal atitinkamą paslaugos charakteristikos abstrakcijos lygmenį (t.y. apdorojimo galingumas, atminties kiekis bei duomenų saugojimo talpa, aktyvių sistemos naudotojų skaičius). Išteklių naudojimas gali būti valdomas, stebimas, raportuojamas, užtikrinant skaidrumą ir aiškumą tiek teikėjui, tiek klientui. Tai įprastai skirta apskaičiuoti paslaugos kainą, mokant už konkretų išteklių naudojimą.

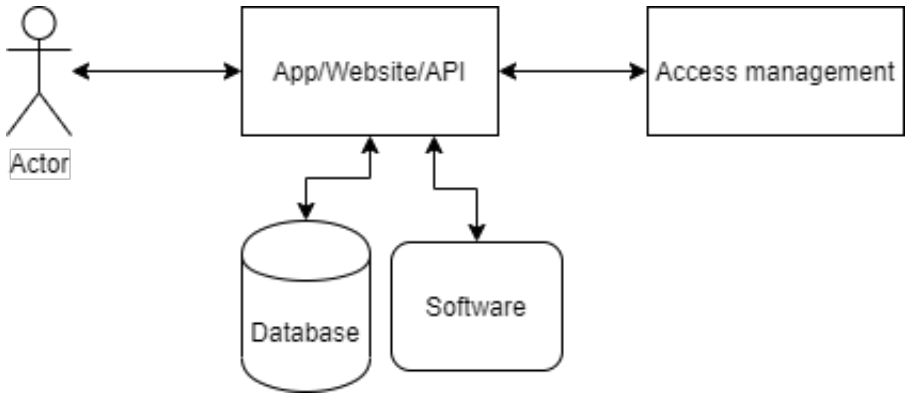
Palyginimui, įprastai IT organizacijos turi nuosavą techninės įrangos infrastruktūrą (arba įsigytą iš išorinių teikėjų), įskaitant duomenų centrus, serverius, tinklo sistemą, administratorius. To privalumai yra visiškai arba dalinai savarankiškas išteklių valdymas, galimybė rinktis techninę įrangą pagal savo norus ir poreikius. Trūkumai: nelankstumas ir sunkiai išmatuojamas išteklių poreikis, dažnas jų trūkumas arba perteklius, būtinybė turėti kompetentingą personalą, prižiūrintį infrastruktūrą, bei fizinio išlaikymo kaštai (duomenų centrų patalpos, elektros energija, aušinimas, tinklo sistemos įranga). Taip pat be galo svarbu pasirūpinti saugumu ir prieinamumu – įvairūs techninės įrangos gedimai įmonei gali itin brangiai kainuoti, sutrikdydami ar visiškai sustabdydami įmonės veiklą. Debesis yra daugelio šių problemų sprendimas.

1.2. Paslaugų tipai

Vienas iš pagrindinių debesų kompiuterijos tikslų yra teikti klientui programinės įrangos paslaugas (angl. Software services), suteikiant galimybę jam nesirūpinti technine įranga (angl. Hardware). Debesies teikiamos paslaugos įprastai skirstomos į tris modelius [**GSG12**][**MG11a**][**GER+12**], kurie pristatomi tolimesniuose poskyriuose, apžvelgiant jų veiklos modelį, kontrolės lygmenis, pateikiant keletą paslaugų pavyzdžių, paanalizuojant keletą saugumo ir privatumo klausimų, privalumus bei trūkumus.

1.2.1. *Software as a service (SaaS)* modelis

Vartotojas naudojasi programinės įrangos aplikacijomis, veikiančioms debesies teikėjo infrastruktūroje. Jos įprastai pasiekiamos naudojant interneto prieigą turinčiais įrenginiais per interneto naršyklę arba programinę vartotojo sąsają. Klientas neturi jokios programinės įrangos infrastruktūros kontrolės, t.y. negali keisti jos veikimo ar operacinės sistemos, tačiau kartais yra galimybė konfigūruoti komponentų prieigą ar teises. Ko gero, žinomiausias *SaaS* modelio aplikacijos pavyzdys yra elektroninis paštas, pvz.: Gmail, Outlook. Šiam modeliui taip pat priklauso duomenų saugykla Dropbox, Google Apps (Google Docs, Google Drive), pokalbių aplikacija Slack, Microsoft Office 365 aplikacijų rinkinys. *SaaS* modelio paslaugas kasdien naudoja bene kiekvienas išmaniojo telefono savininkas. Aplikacijų naudojimo metu sugeneruoti duomenys yra saugomi debesyje, dažnu atveju ir vartotojo įrenginiuose.



1 pav. *SaaS* paslaugos komponentai (autoriaus iliustracija)

SaaS modelio paslaugos, kaip ir visų kitų debesies paslaugų modelių, bene didžiausias privalumas yra kaina, nes vartotojai neturi rūpintis ir investuoti brangią į techninę ir programinę įrangą, *SaaS* atveju - paslaugų naudojimui nebūtina galinga techninė įranga. Kitas svarbus privalumas yra pasiekiamumas - vartotojai paslauga gali naudotis iš skirtingų interneto prieigą turinčių įrenginių. Vartotojui nėra reikalingas konkretus įrenginys, kuriame yra įdiegta programinė įranga ir paslauga pasiekiama tik per jį. Šis bruožas taip pat yra didelis privalumas saugumo atžvilgiu, nes vartotojui nereikia fiziškai nešiotis konkretaus įrenginio, kuris, pavyzdžiui, darbinės kelionės metu gali būti pavogtas [**GER+12**].

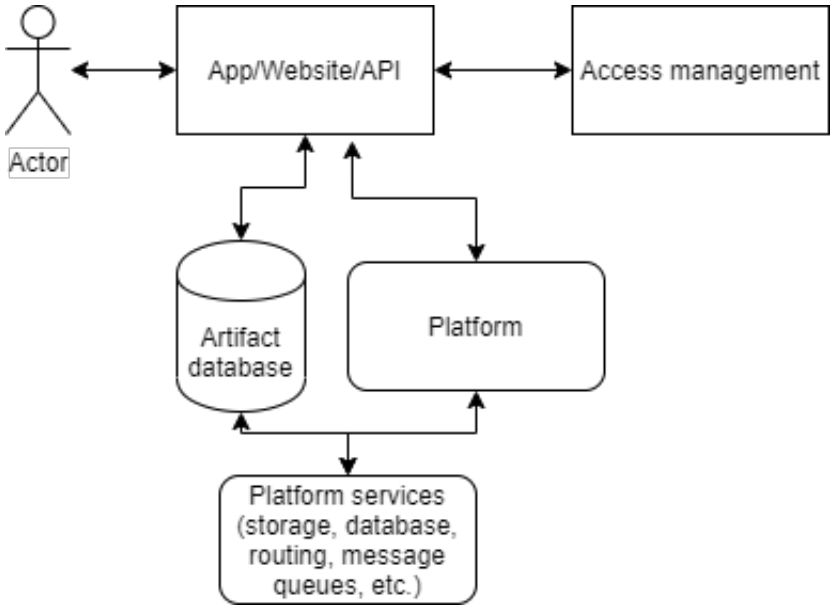
Tačiau kyla nemažai neaiškumų, daugiausiai susijusių su teisiniais ir privatumo klausimais.

Pavyzdžiui, debesies duomenų centrų jurisdikcija, personalo galimybės pasiekti vartotojų duomenis, duomenų kriptografija ir vientisumas, vartotojams dažnai nėra žinoma informacija [**GER+12**].

1.2.2. *Platform as a service (PaaS)* modelis

Suteikia vartotojui platformą (virtualią techninę įrangą) kurti bei saugoti savo ar įsigytas programas, sukurtas naudojant įvairias programavimo technologijas, kalbas, bibliotekas, įrankius, palaikomus debesies teikėjo. Šio modelio paslaugų vartotojai dažniausiai būna programuotojai, kurie suteikiama paslauga naudojasi per taikomųjų programų programavimo sąsają (API). Klientas nevaldo debesies infrastruktūros komponentų, pvz.: tinklo, serverių, operacinės sistemos, tačiau turi galimybę kontroliuoti savo saugomas aplikacijas, kartais ir konfigūracijos nustatymus. *PaaS* modelio žinomiausios paslaugos: Heroku, Google App Engine, Windows Azure (dažniausiai naudojamas kaip *PaaS*), duomenų bazių valdymo sistema (DBVS) Amazon SimpleDB.

Svarbu suprasti, kad *PaaS* modelio paslaugos iš esmės yra sukurtos *IaaS* paslaugų pagrindu [**GER+12**], t.y. suteikiama prieiga prie iš anksto sukonfigūruotų, paskirstytų virtualių aplinkų. Tai klientams suteikia galimybę mažinti administravimo, aplinkos paruošimo darbui bei palaikymo kaštus.



2 pav. *PaaS* paslaugos komponentai (autoriaus iliustracija)

PaaS modelio paslaugos suteikia galimybę programinės įrangos inžinieriams, architektams bei sistemų administratoriams patogiai valdyti savo sistemas vienoje vietoje. Klientų įdiegti produktai įprastai yra konteinerizuojami (virtualizuojami naudojant konteinerių technologijas), taip užtikrinant optimalų veikimą ir lankstumą. Konteinerius galima suprasti kaip tam tikrą abstrakcijos lygmenį, leidžiantį lanksčiai keisti sistemos komponentus.

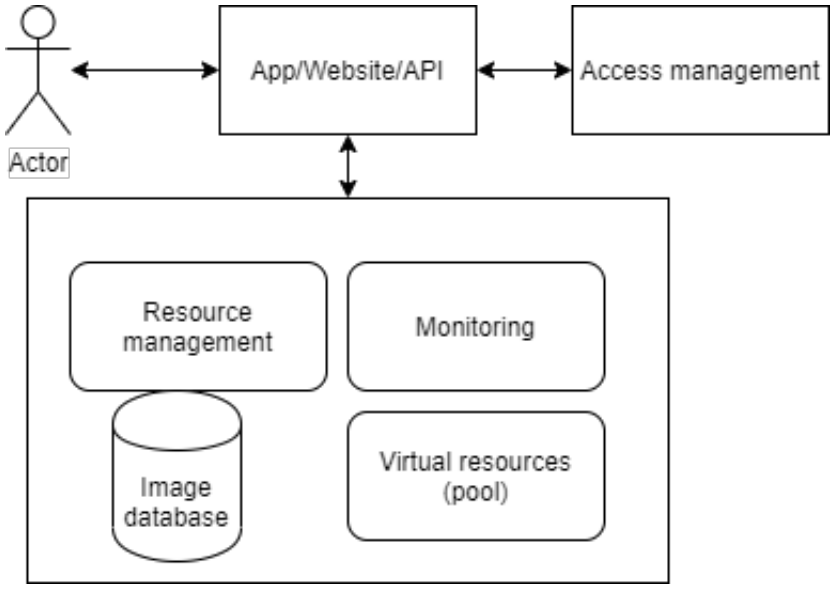
PaaS modelio paslaugų trūkumas yra suderinamumas. Dauguma *PaaS* iš esmės veikia skirtingai, tad migracija nuo vienos paslaugos prie kitos gali būti sudėtinga [**GER+12**], todėl kiekvienas klientas turi pasirinkti sau tinkamiausią paslaugos teikėją, siekiant išvengti perteklinių konfigūracijos ir pasirengimo (arba sistemos pritaikymo) darbų. Kaip ir kito modelio paslaugoms, *PaaS*

paslaugų duomenų saugojimo būdai bei geografinė lokacija yra potenciali teisinė problema. *PaaS* paslaugos neišvengia ir saugumo bei prioritetizavimo iššūkių. Visų pirma, paslaugos teikėjas privalo užtikrinti kiekvieno kliento interesų tenkinimą, tad negali būti jokių išskirtinių privilegijų vienam ar kitam klientui. Be to, klientai neturi turėti jokių prieigos galimybių prie kitų klientų platformų ar produktų [**GER+12**].

1.2.3. *Infrastructure as a service (IaaS)* modelis

Vartotojui suteikiama virtualizuotų kompiuterinių resursų, pvz.: apdorojimo galios, atminties, nuolatinės duomenų saugyklos, skirtos saugoti virtualias mašinas, prieiga. Šiame modelyje klientas turi daugiausiai kontrolės iš visų paminėtų modelių, iš esmės yra suteikiama galimybė turėti savo infrastruktūros valdymą bei ja pilnai naudotis: pasirenkama norima operacinė sistema, diegiamos aplikacijos, duomenų saugyklos dydis. Tačiau tai nereiškia, kad suteikiamas kontrolės lygmuo yra ekvivalentus kaip ir nuosavos infrastruktūros, kur savininkui nėra jokių apribojimų. Pačios glūdinčios debesies infrastruktūros vartotojas valdyti negali, kaip ir daugeliu atvejų negali konfigūruoti tinklų. *IaaS* modelio paslaugos: Amazon Elastic Compute Cloud (EC2), Digital Ocean, Google Compute Engine (GCE).

IaaS modelio paslaugų vartotojai gali naudoti paslaugos teikėjo skiriamus ir valdomus virtualius išteklius ir atlikti anksčiau minėtas veiklas bei mokėti tik už išnaudotus išteklius. Paslaugos teikėjas turi detalią sistemos veiklos stebėjimo sistemą (angl. Monitoring), tad gali itin tiksliai nustatyti kliento sunaudotų išteklių kiekį bei pateikti atitinkamą sąskaitą. Taigi, nors ir vartotojai turi laisvę naudotis fizine bei virtualia teikėjo infrastruktūra, turi galimybę laisvai priimti sprendimus dėl savo virtualių mašinų operacinės sistemos, konfigūracijos bei programinės įrangos, veikiančios jose, tačiau neša atsakomybę dėl optimalaus išteklių valdymo. Tiesa, *IaaS* paslaugų (pvz.: Azure DevOps) teikėjai suteikia galimybę apsibrėžti išlaidų biudžetą ir tokiu būdu klientai gali apsisaugoti nuo netikėtų išlaidų, atsiradusių dėl savo pačių klaidų - prasto išteklių valdymo.

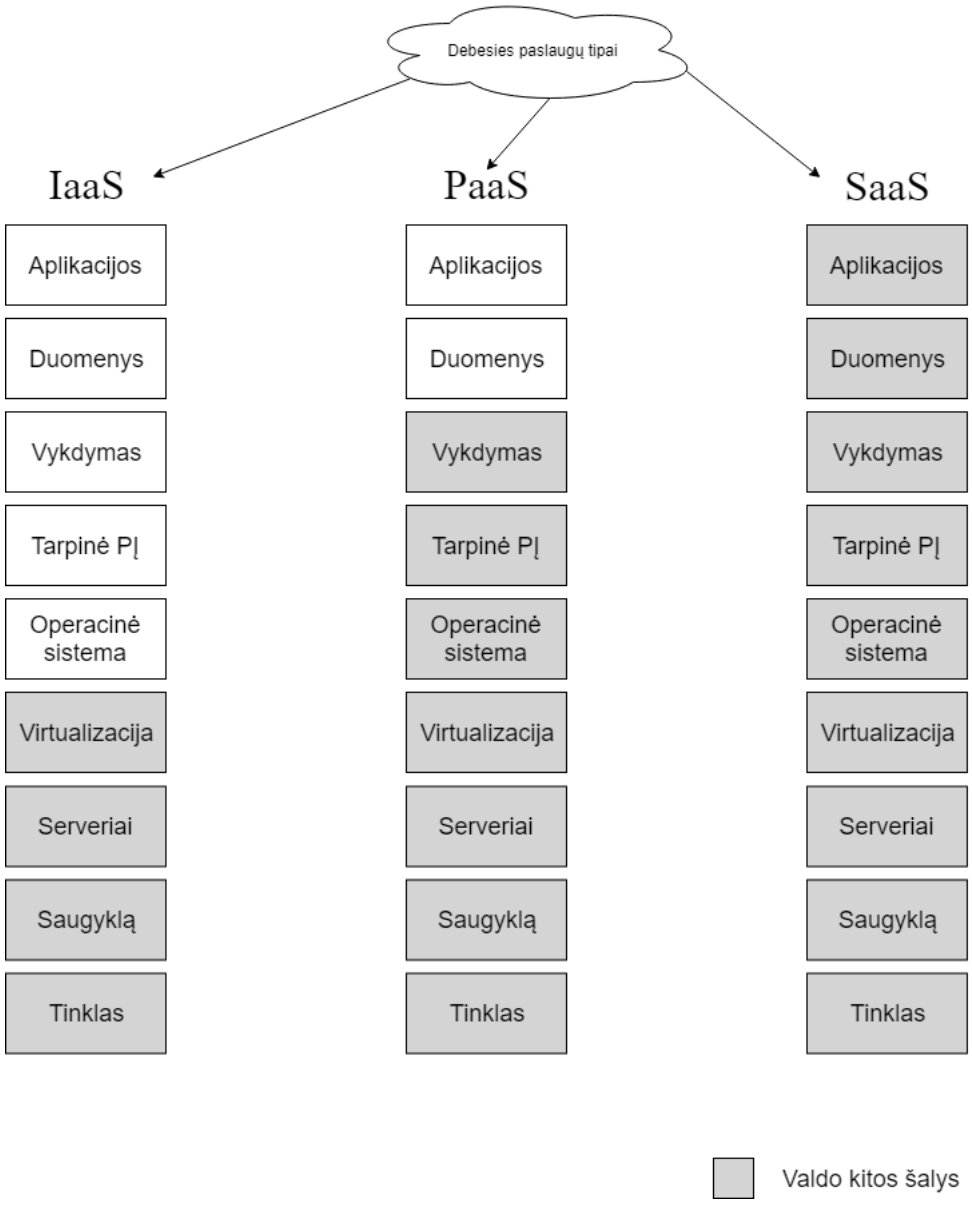


3 pav. *IaaS* paslaugos komponentai (autoriaus iliustracija)

IaaS modelio paslaugų saugumas yra bene svarbiausias punktas debesų kompiuterijos sistemose, nes kito modelio paslaugos yra sukurtos *IaaS* pagrindu [**GER+12**]. Svarbu suprasti, kad, šiuo atveju, viena įmonė saugo kitų kompanijų duomenis ir yra tam tikros jų veiklos vykdytojai. Paslaugos teikėjas neša didžiulę atsakomybę dėl savo paslaugos saugumo, skaidrumo ir privalo ne tik užtikrinti nepageidaujamos prieigos prie virtualių resursų draudimą, apsaugoti sistemas nuo nepageidautinos veiklos (pvz.: hipervizoriaus sluoksnio kompromitavimo, netinkamo prieigos teisių suteikimo, einamuoju momentu veikiančios mašinos duomenų nepageidautino pakeitimo ir t.t.), bet ir užtikrinti detalų ir prasmingą žurnalizavimą, saugų užšifruotą komunikacijos kanalą.

1.3. Išvados

Skirtingi paslaugų tipai suteikia klientui galimybę turėti skirtingos grupės komponentų kontrolę (žr. 4 pav.), kartu ir apibrėžiamos atsakomybių sritys.



4 pav. Šalių valdomos sritys (autoriaus iliustracija)

Remiantis 4 iliustracija, matoma, jog naudojantis bet kurio tipo paslaugomis, už techninę įrangą visuomet atsako kitos šalys (t.y. paslaugos teikėjas), tai verslui suteikia galimybę ženkliai sumažinti kaštus ir susitelkti ties programine įranga. Pabrėžtina, kad debesies paslaugos teikėjas nebūtinai yra infrastruktūros savininkas.

Pastebima, kad visi paslaugų tipai iš esmės yra sukurti *IaaS* pagrindu, tad galime teigti - *IaaS* paslaugų saugumo užtikrinimas yra vienas iš svarbiausių uždavinių.

SaaS ir *PaaS* paslaugų modelyje didžiąją dalį kontrolės turi paslaugos teikėjas, tad įvykus incidentui ir atliekant tyrimą, bendradarbiauti su paslaugos teikėju yra naudinga, nes šis gali pateikti daug naudingos informacijos bei suteikti prieigą prie tam tikrų artefaktų (pvz.: veiklos žurnalų, tam tikrų virtualizacijos lygmenų ir t.t.). Tačiau *IaaS* paslaugų modelyje teikėjas be išankstinio pasiruošimo negali suteikti tiek daug informacijos kiek galėtų kitų modelių atveju - kokią veiklą savo virtualiose mašinose vykdo klientas, ką ten saugo, kas ten veikia, kokie duomenys ten saugomi - nėra aišku. Tad identifikuoti artefaktus kliento virtualiose mašinose yra didelis iššūkis. Įrankių, galinčių automatiškai tirti (t.y. identifikuoti norimus artefaktus) didelius kiekius virtualių mašinų, šiuo metu rasti nepavyksta.

2. Skaitmeninė ekspertizė ir reagavimas į incidentą

Informacinės technologijos ir sistemos kasdien susiduria su įvairiomis grėsmėmis. Fizinis asmuo savo išmaniajame telefone saugo ne tik aibę asmeninių duomenų, nuotraukų bei dokumentų, bet ir naudojasi prieiga prie bankinių sistemų, elektroninės komercijos sistemų, saugo slaptažodžius, skirtus prieigai prie savo asmeninių socialinių tinklų, el. pašto bei kitų sistemų. Organizacijos didelę dalį savo veiklos vykdo naudodamos informacines technologijas: internetinės prekybos platformos, reprezentacinės svetainės, vidinio naudojimo programos, dokumentų archyvai. Žmogui, įmonei ar net valstybei bet koks informacijos nutekėjimas ar jos vagystė, veiklos sutrikdymas ar visiškas įrangos sugadinimas gali sukelti milžinišką finansinę ir moralinę žalą. Bandymas gauti neautorizuotą prieigą prie sistemų, neleistinas jų naudojimas, nelegalūs sistemos pakeitimai bei kiti anksčiau minėti įvykiai bendrai yra vadinami kibernetiniais incidentais. Kibernetinius incidentus sukelia kibernetines atakas vykdantys asmenys ar organizacijos, siekiančios nelegaliai pasipelnyti, pakenkti ar sutrikdyti asmens ar įmonės veiklą, skleisti klaidingą informaciją arba pasisavinti slaptus duomenis.

Nuo nusikaltimų niekas nėra apsaugotas, tačiau tiek fizinėje, tiek kibernetinėje erdvėje egzistuoja prevencinės priemonės, skirtos užkirsti kelią incidentams, o jiems įvykus – surasti atsakingus asmenis ir įrodyti jų kaltę teisme, pateikiant įkalčius. Visas incidento tyrimo, įkalčių surinkimo ir analizės procesas vadinamas skaitmenine teismo ekspertize (angl. Digital Forensics).

2.1. Įkalčių surinkimas

Renkant skaitmeninius įkalčius būtina laikytis proceso, kad surinkta informacija būtų vienareikšmė ir aiški, surinkta visa, kurios reikia tyrimui, nepažeidžiant įstatymų. Remiantis JAV teisingumo departamento pateikiamu gidu [**NSH08**], įkalčiai turi būti:

1. Gauti legaliai. Pirmasis atsakytojas (angl. First responder) privalo turėti teisę rinkti įkalčius – turi būti gautas savininko leidimas ar teismo įsakas. Jis taip pat turi įsitikinti, kad savo darbą atlieka šalies, kurioje turi teisę dirbti, jurisdikcijoje.
2. Vientisi ir nepakitę. Atsakytojas privalo imtis visų būtiniausių priemonių apsaugoti fizinę įkalčių laikmeną. Kai įkalčių išsaugoti fizinėje laikmenoje neįmanoma, būtina įrodyti kopijos teisėtumą. Tam įprastai naudojami sertifikuoti maišos algoritmai (angl. Hash algorithms) bei detali dokumentacija.
3. Dokumentuoti [**Joh17**]. Pareigūnas privalo kiekvieną žingsnį aiškiai dokumentuoti, nurodant:
    - Kokie duomenys buvo surinkti,
    - Kaip surinkti, kokia programinė ar techninė įranga buvo naudojama,
    - Dokumentavimo laikas (įprasta praktika yra naudoti GPS laikrodžius, užtikrinant laiko validumą).

Įkalčių rinkimo sekos (angl. Chain of custody) procesas yra kritiškai svarbus, privalo būti užfiksuotas kiekvieno įkalčio gyvavimo ciklas, nuo įkalčio gavimo iki jo sunaikinimo ar grąžinimo savininkui. Esant sekos trūkiams ar dokumentacijos netikslumams, įkalčiai gali būti pripažinti

negaliojančiais ir neįtraukti į tyrimą [**NSH08**].

2.2. Skaitmeninio tyrimo modelio struktūra

Skaitmeninio tyrimo arba skaitmeninės ekspertizės procesas neapsiriboja vien reagavimu į incidentus bei įkalčių surinkimu. Šis daugeliu atvejų prasideda nuo incidento užfiksavimo momento ir baigiasi ištirtos medžiagos pateikimu teismui ar kitoms institucijoms. Siūlomų skaitmeninės ekspertizės proceso modelių variantų yra daug, tačiau visi jie iš esmės yra panašūs – šioje dalyje panagrinėsime 2001 metais vykusios pirmosios „Skaitmeninės ekspertizės tyrinėjimo konferencijos“ (angl. Digital Forensics Research Workshop) metu pasiūlytą „Skaitmeninio tyrimo schemą“ (angl. Digital Investigative Process), trumpai vadinamą DIP. Ši susideda iš šešių elementų [**NSH08**]:

1. **Identifikacija**. Lokardo mainų principas yra itin populiari diskusijų tema tyrėjų bendruomenėje. Šis teigia, kad du objektai sąveikaudami palieka pėdsakų, kaip, pavyzdžiui, pirštų antspaudai ar pėdų įspaudai. Skaitmeniniame pasaulyje šis principas taip pat galioja – sistemoms kontaktuojant tarpusavyje jų veikla palieka savotiškus „pėdsakus“ [**Sam12**]. Žinomiausias pavyzdys būtų interneto naršyklėse saugomi slapukai, fiksuojantys naršytojo veiklą [**NSH08**]. Taip pat prisijungimas prie sistemų yra žurnalizuojamas, t.y.: užfiksuojamas prisijungimo laikas, prisijungiančiojo IP adresas. Įvairių žurnalų (angl. Logs) peržiūra suteikia pakankamai daug informacijos pastebėti sistemos veiklos sutrikimus pagal tam tikrą laiko tarpą ar konkretaus individo veiklą. Tai yra vienas iš potencialių įkalčių identifikacijos pavyzdžių.
2. **Apsaugojimas**. Identifikavus potencialius įkalčius, būtina imtis apsaugos priemonių, užtikrinant, kad jie nebus pakeisti ar ištrinti. Primityviausias būdas yra izoliuoti sistemą nuo tinklo, kartais ją net visiškai išjungiant. Tačiau toks būdas dažnai yra nepageidautinas, kadangi sistemų išjungimas organizacijoms gali būti tolygus veiklos sustabdymui, o tai gali sukelti didžiulių nuostolių ar pakenkti jų klientams. Įkalčius galima apsaugoti ir sisteminiais nustatymais, pavyzdžiui, uždraudžiant žurnalų failų modifikaciją, sistemą perjungiant į skaitymo režimą ar juos išsaugoti atliekant momentinę sistemos kopiją (angl. Snapshot). Taip pat svarbu stipriai apriboti vartotojų prieigą prie sistemos, kadangi šie, tyčia ar netyčia, gali negrįžtamai pakeisti įkalčius. Su pažeista sistema derėtų elgtis itin atsargiai ir suteikti prieigą tik kvalifikuotiems specialistams.
3. **Surinkimas**. Šiame žingsnyje pirmasis atsakytojas ar kitas kompetentingas agentas pradeda įkalčių surinkimo procesą. Svarbu suprasti, kad duomenų rinkimas turi būti labiau kokybinis nei kiekybinis – dideli pašalinės informacijos kiekiai gali trukdyti tyrimui, vilkinti procesą. Tačiau būtina surinkti visus potencialius įkalčius ir kiek įmanoma daugiau svarbios informacijos, kuri būtų naudinga tyrėjui. Renkant skaitmeninius duomenis agentui būtina pasirūpinti reikiama technine (pavyzdžiui: pakankamos talpos laikmena kopijai, rašymą blokuojančiu įrankiu) ir programine (pvz.: FTK Imager, WinPmem, Wireshark) įranga bei suvokti galimus įkalčių šaltinius, jų principus. Pavyzdžiui, daug vertingos informacijos galima rasti nepastoviuose duomenyse (angl. Volatile data). Šie duomenys gali būti prarasti išjungus sistemą. Galimi šių duomenų šaltiniai yra operatyvioji atmintis (angl. RAM), priešatmintis

(angl. Cache), maršrutizavimo lentelė (angl. Routing table). Rekomenduotina pirmiau rinkti įkalčius iš nepastoviosios atminties, pereinant prie pastoviosios, pavyzdžiui, išorinio kietojo disko duomenų.

   (a) Tinkamas įkalčių valdymas yra kritiškai svarbus. Surinkimo procesas jokiu būdų negali pakeisti ar pažeisti įkalčių, šis taip pat privalo būti išsamiai ir nuosekliai dokumentuojamas.

   (b) Įkalčių rinkimo sekos dokumentas yra taisyklingai vykdyto surinkimo proceso įrodymas. Šiame dokumente fiksuojami įkalčiai, jų eilės numeriai, aprašai, jų charakteristikos (gamintojas, modelis, serijos numeris), laikas, fiksuojama transportavimo informacija, nurodant kelią bei priežastį. Skaitmeninių įkalčių modelius bei gamintojus užfiksuoti yra sudėtingiau, tad be minėtų komponentų reikia užfiksuoti ir įkalčio „antspaudą“, įprastai matomą baitų eilutės formatu, panaudojus maišos algoritmą. Dokumentas gali būti ruošiamas automatiškai, naudojant sertifikuotą programinę įrangą, arba rankiniu būdu, informaciją užrašant rašikliu ant popierinės formos.

4. **Ekspertizė**. Šią ir analizės fazę nebūtinai vykdo tas pats asmuo, kuris pirmasis reagavo į incidentą bei surinko duomenis. Šiame etape agentas turi ištirti gautus įkalčius, detaliai aprašyti naudojamą duomenų išgavimo techniką bei to prasmę, naudotus specifinius įrankius. Pavyzdžiui, esant poreikiui analizuoti tinklo veiklą, ekspertui gali reikėti išgauti SSH prisijungimo duomenis. Svarbu paminėti, kad ir šitame etape yra svarbu išsaugoti įkalčių vientisumą bei apsaugą, nes net ir tokioje vėlyvoje stadijoje sugadinti įkalčiai gali tapti nepanaudojamais ar nepatikimais. Taip pat turi būti pašalinta nereikalinga informacija, kad tyrėjas būtų kuo mažiau trukdomas ir turėtų laisvę tirti.
5. **Analizė**. Kompetentingas tyrėjas šioje stadijoje analizuoja surinktų duomenų visumą, bandydamas išaiškinti incidento kaltininką, atkurti incidento scenarijų.
6. **Prezentacija, pateikimas**. Po analizės gautos išvados turi būti konkrečios, aiškios ir nešališkos. Ekspertas pateikia detalią ataskaitą, kurioje aprašomas kiekvienas atliktas veiksmas, svarbių duomenų užfiksavimas. Ataskaita turi būti nuosekli, tiksli ir be jokių interpretacijų. Šis dokumentas dažniausiai būna didesnio tyrimo dalis ir padeda išaiškinti incidento kilmę. Be to, ekspertas gali būti pakviestas liudyti teisme, pateikti ir pristatyti faktus bei išvadas, vengiant nuomonių ar šališkumo. Svarbu suprasti, kad pristatyti išvadas gali tekti ir techniškai nekompetentingiems asmenims, tad svarbu gebėti informaciją pateikti suprantamu formatu jos neiškraipant.

Konferencijoje, kurioje buvo suformuotas šis modelis, buvo sutarta, kad jį reikia plėtoti ir tobulinti. Ši schema buvo atspirties taškas naujiems tyrimams, jau po metų modelis papildytas dar trimis fazėmis: pasiruošimo, strategijos bei įkalčių grąžinimo [**GSG12**], detaliais fazių aprašais. Šis papildytas DIP modelis suformuoja esmines skaitmeninės ekspertizės gaires. Tačiau bandant pritaikyti šią schemą debesų kompiuterijos sistemomis, kyla itin daug neaiškumų, nuo įkalčių identifikavimo iki jų surinkimo bei analizės. Abstrakčios schemos fazės tyrimui yra tinkamos net tokiose sistemose, bet jų realizacija yra problematiška.

3. Skaitmeninės ekspertizės ir reagavimo į incidentą procesų iššūkiai debesų kompiuterijos sistemose

Šiame skyriuje išnagrinėta kiekvieno DIP modelio etapo problematika debesų kompiuterijos sistemose, remiantis [**MCC+19**], [**RCK+11**], [**AIJ14**].

3.1. Identifikacija

Tiriamųjų objektų, artefaktų, potencialių įkalčių bei apskritai - incidentų - identifikavimas yra viena iš opiausių problemų, kylančių dėl paties debesies veikimo principo. Debesies išteklių fizinė **lokacija** nėra konkreti - paslaugos teikėjas valdo didelį kiekį duomenų ir kompiuterinių išteklių centrų, kurie įprastai yra skirtingose pasaulio vietose. Tam įtaką daro kaštų mažinimas, laiko zonos, žmogiškieji ištekliai ir kiti verslo sprendimai. Įprastu atveju, tyrėjas turi arba fizinę prieigą prie tiriamojo objekto (pvz.: gali gauti konkretų kietąjį diską iš konkretaus serverio, žino konkretų kompiuterį su konkrečia programine įranga), arba nuotolinę prieigą, kuria gali pasinaudoti dėka teisinių įrankių. **Teisėta prieiga** prie tiriamųjų objektų yra būtina, kitaip įkalčiai būtų laikomi negaliojančiais ir pats tyrėjas potencialiai susilauktų rimtų teisinių problemų. Kiekviena valstybė ir/ar sąjunga turi savo teisinius įrankius ir neturi teisės reikalauti duomenų iš kitos jurisdikcijos, negavusi jų leidimų. Vienos valstybės asmens duomenų apsaugai skiria ypatingą dėmesį ir atsakingai riboja prieigą prie jų, kitų valstybių teisiniai organai gali pareikalauti skubiai pateikti vartotojų duomenis prieš tai jų net neįspėjus [**RHG+16**]. Maža to, fizinė lokacija ne tik nėra konkreti, bet neretai ir nežinoma [**RCK+11**], tą gali padaryti tik debesies paslaugos teikėjas - dažnai tik labai abstrakčiai.

Artefaktai debesų kompiuterijos sistemose labai dažnu atveju yra **nevientisi**, t.y. paskirstyti skirtinguose duomenų centruose. Virtualieji ištekliai dažnai yra migruojami bei klonuojami, siekiant užtikrinti nepertraukiamą prieinamumą [**RCK+11**]. Viena iš konkretesnių virtualių artefaktų problemų yra **daugialypės virtualizacijos**, potencialūs įkalčiai gali būti virtualizuoti skirtinguose lygiuose, o tai itin apsunkiną jų identifikaciją.

Be to, kai kurie debesies paslaugų teikėjai saugomus duomenis **užšifruoja**, gauti reikalingus kriptografinius raktus taip pat gali būti ilgas ir sudėtingas procesas.

Tyrimą vykdantys asmenys įprastai bendradarbiauja su sistemų savininkais ir/ar vartotojais (įmonės sistemų administratoriai, programuotojai, fiziniai asmenys), siekiant identifikuoti incidentą ir galimai tyrimui naudingus artefaktus. Debesų kompiuterijos sistemų atveju, tyrėjai turi bendradarbiauti ne tik su paslaugų klientais (kurie gali būti jau minėtų pareigų specialistai ar tiesiog vartotojai), bet ir su paslaugos teikėju. Tyrimo eiga iš esmės **priklauso nuo debesies paslaugos teikėjo**, jo nustatytų taisyklių ir, galų gale, jurisdikcijos bei galimybių. Tenka pasikliauti jų sukurtos infrastruktūros skaidrumu ir duomenų prieigos galimybėmis - iš paslaugos kliento pusės, prieiga prie veiklos žurnalų ar metaduomenų dažniausiai yra stipriai apribota [**RCK+11**]. *SaaS* modelio paslaugų sugeneruotus duomenis apskritai yra sudėtinga gauti dėl menkų kliento kontrolės galimybių, *IaaS* modelio atveju - prieiga prie virtualių mašinų ar saugyklų atvaizdų irgi yra teikėjo rankose. Tiesa, *IaaS* sistemų virtualių mašinų prieigą taip pat turi ir klientas, tad galimybė

tirti pačias virtualias mašinas su kliento sutikimu (arba orderiu) yra reali.

Be jau egzistuojančios tyrimo eigos priklausomybės nuo debesies paslaugos teikėjo, šio **paslaugos gali priklausyti nuo kitų debesų paslaugų teikėjų (priklausomybių grandinė)**. Pavyzdžiui, *SaaS* paslaugos teikėjas savo programinę įrangą gali kurti bei valdyti kito teikėjo *PaaS* suteikiamoje platformoje, kurios pagrindas yra dar kito teikėjo *IaaS* virtualios infrastruktūros, o šis teikėjas fizinius techninius išteklius gali pirkti iš dar kito teikėjo. Toks atvejis puikiai iliustruoja daugelį minėtų sunkumų, t.y. teisines problemas, prieigos nebuvimą, paslaugos neskaidrumą. Galų gale tai itin apsunkina ir vilkina patį tyrimo procesą.

3.2. Apsaugojimas

Teoriškai, sustabdyti tam tikrų sistemų veiklą iki tyrimo pabaigos, naudojant teisines priemones, yra įmanoma net debesų kompiuterijos sistemose. Deja, praktiškai tai įvykdyti yra itin sudėtinga, nes debesies paslaugos teikėjas veiklą vykdo su daugiau nei vienu klientu - kitų veiklos stabdyti pagrindo nėra [**AIJ14**]. Sistemų veiklos stabdymas yra įprasta praktika ne debesų kompiuterijos sistemų atveju, siekiant **užtikrinti įkalčių vientisumą, nepakitimą tyrimo metu bei izoliaciją**. Tai yra itin sudėtinga debesies atveju dėl duomenų bei veiklos paskirstymo prigimties - tam tikrų nusikaltimų atvejais yra praktiškai neįmanoma atgaminti įkalčių priklausomybės seką [**AIJ14**].

Įkalčių nepastovumas (kintamumas) - debesyje ištrinti virtualią mašiną ir nepalikti jos veiklos pėdsakų yra paprasta, o tai gali amžiams sunaikinti potencialius įkalčius. Pavyzdžiui, tam tikros *PaaS* sistemos nuolat atnaujina, perkrauna aplikacijų konteinerius, neišsaugant ankstesnės būsenos. Laiku atlikta sistemos momentinė kopija gali padėti išsaugoti nepastovius duomenis tyrimui svarbiu momentu, tačiau to nepadarius atkurti sistemos veiklą norimu laiko momentu yra neįmanoma - debesies paslaugų teikėjai įprastai nesaugo ištrintų sistemų atvaizdų/kopijų, nes tai paprasčiausiai kainuotų be galo daug išteklių.

Įkalčių rinkimo seka turi būti aiškiai ir nuosekliai dokumentuojama. Dideliu iššūkiu tampa įkalčių gavimo **laiko sinchronizacija dėl paskirstytų duomenų lokacijų** [**RCK+11**]. Jeigu įkalčių surinkimo sekos dokumentas yra nevientisas ir klaidingas - gauti įkalčiai teisme gali būti pripažinti negaliojančiais.

3.3. Surinkimas

Esminis iššūkis, kylantis surinkimo etape yra **įrankių trūkumas bei esamų įrankių nesuderinamumas** [**GSG12**]. Nors ir debesų kompiuterija bei skaitmeninės ekspertizės procesas debesyje yra sąlyginai senos sritys, paprasčiausiai nėra pakankamai sukurtų įrankių, būtinų įkalčių surinkimui [**MCC+19**]. Egzistuoja nemažai sprendimų, leidžiančių surinkti įkalčius iš kompromituotų sistemų, tačiau **tam reikia nuoseklaus išankstinio pasiruošimo ir išteklių**. Tyrėjui gali būti labai sudėtinga surinkti įkalčius iš kompromituotų debesies sistemų dėl įvairių priežasčių: prieigos nebuvimas arba prieigos egzistavimas tik iš kliento pusės, populiarūs įrankiai, naudojami ne debesies sistemų ekspertizės procese, gali būti nenaudingi, o tam tikrų įrankių diegimas į kompromituotą

sistemą jau pakeičia jos būseną ir gauti įkalčiai gali būti pripažinti negaliojančiais.

Dar viena labai opi problema yra **augantis įkalčių kiekis**. Vidutinis vienos skaitmeninio incidento bylos dydis kasmet sparčiai auga [**RCK+11**], [**GSG12**]. Tyrėjui būtina surinkti kiek įmanoma kokybiškus įkalčius, vengiant nereikalingos pašalinės informacijos (triukšmo). Galima įsivaizduoti kiek laiko gali užtrukti dešimties terabaitų sistemos analizė. Taigi, nekokybiškai atliktas surinkimo etapas gali iš esmės nulemti tolimesnę ekspertizės proceso nesėkmę.

Dalis problemų surinkimo etape yra tokios pačios kaip ir ankstesniuose etapuose: **priklausomybė nuo debesies paslaugos teikėjo**, **teisinės problemos**, **duomenų nepastovumas**. Iš priklausomybės nuo debesies paslaugos teikėjo problemos, kyla dar vienas iššūkis - **pasitikėjimas**. Juk realu, kad pats paslaugos teikėjas gali būti įsivėlęs į nusikalstamą veiklą, tad yra suinteresuotas tam tikrų įkalčių neaptikimu. Tam tikri skaitmeninės ekspertizės debesyje sprendimai į tai, kad paslaugos teikėjas galimai yra nepatikimas, neatsižvelgia [**MCC+19**].

3.4. Ekspertizė ir analizė

Šių dviejų etapų problemos iš esmės yra tos pačios, tad galima jas apžvelgti bendrai. Šiose fazėse tyrėjai tiria gautus įkalčius, patikrina jų vientisumą ir prasmingumą bei bando išaiškinti incidento aplinkybes. Čia viena svarbiausių problemų yra panaši kaip ir ankstesniuose etapuose - **įkalčių vientisumo ir patikimumo užtikrinimas**. Gautus įkalčius reikia sulyginti su realiais duomenimis sistemoje, įprastai tai daroma tikrinant antspaudus. Antspaudų gavimas gali būti sudėtingas procesas, nes, vėlgi, **egzistuoja priklausomybė nuo debesies paslaugos teikėjo** bei **pasitikėjimo juo klausimas**. Netgi gavus patikimus įkalčius, juos tirti gali būti labai sudėtinga dėl jų **formatų nevienodumo bei kliūčių juos apjungiant**. Vienas populiariausių pavyzdžių yra skirtingi žurnalų formatai - praktiškai neįmanoma jų suvienodinti, dėl to sunku stebėti veiksmų seką laike. Iš to seka ta pati problema, esanti apsaugojimo etape - **laiko žymų nelygybė**. Dėl skirtingų laiko juostų bei skirtingų sistemų lokalių laikų analizuoti tam tikrą veiklą gali būti labai painus procesas.

3.5. Prezentacija, pateikimas

Pagrindiniai iššūkiai ties paskutine DIP modelio faze iš esmės galioja ir debesų kompiuterijos sistemų tyrimams: **ataskaitos nuoseklumas ir vientisumas**, **įkalčių teisinio pagrindo įrodymas**, **įkalčių surinkimo seka**, **įrodymas, kad įkalčiai surinkti teisėtais būdais** bei **techninių detalių aiškinimas nekompetentingiems asmenims**. Be jau sudėtingų techninių detalių pateikiant įkalčius, svarbu užtikrinti debesies technologijos supratimą. Šiame etape iš esmės patikrinami visi ankstesnių etapų problemų sprendimai.

4. Skaitmeninės ekspertizės ir reagavimo į incidentą proceso patobulinimai, taikant debesų kompiuterijos sistemoms

Nors ir tradicinis DIP modelis yra aukšto lygio abstraktus skaitmeninės ekspertizės procesas, jo siūlomi tyrimo žingsniai debesų kompiuterijos sistemoms kelia daug neaiškumų. Teigiama, kad nėra vieno universalaus proceso, tinkančio kiekvienam tyrimui, siekiant gauti įkalčius [**AIJ14**], tačiau tokio proceso, kuriuo vadovaujantis tyrėjai būtų nukreipti tinkama eiga ir galėtų lengviau išspręsti kylančius neaiškumus debesies aplinkose rasti nepavyko.

Šiame skyriuje pristatomas aukšto lygio skaitmeninės ekspertizės proceso patobulinimo pasiūlymas, parengtas remiantis DIP modeliu.

4.1. Pasiruošimo etapo įtraukimas

DIP modelį būtų galima papildyti pasiruošimo etapu, orientuotu į specialistų apmokymą bei gilesnį egzistuojančių debesų kompiuterijos paslaugų pažinimą.

Kadangi egzistuoja daug skirtingų debesies paslaugų tipų, dar daugiau skirtingų jų implementacijų, paslaugų teikėjai galėtų specialistams padėti iš anksto pasiruošti vykdydami mokymus ir skaidrindami savo sistemos veiklą. Po incidento, tyrėjų komanda neturi galimybės skirti daug laiko paslaugų teikėjo siūlomų paslaugų taisyklių analizei ir naudojamo debesies architektūros nagrinėjimui. Idealiu atveju, patys teikėjai galėtų ruošti personalą, atsakingą už reagavimą į incidentą, kurie bendradarbiautų su tyrėjų komanda ir taip palengvintų bei pagreitintų tyrimo procesą. Deja, tokie specialistai, ko gero, praktiškai neturėtų nuolatinio darbo ir dirbtų kitus darbus budėjimo režime, tad teikėjui būtų palanku tokias paslaugas samdytis iš išorės, vykdant mokymus.

Didžiosios debesies paslaugų kompanijos kaip Amazon, Google ar Microsoft galėtų apmokyti tyrėjų komandas ir pareigūnus iš anksto, vesdami mokymus ir išduodami sertifikatus, kad šie, įvykus incidentui konkretaus teikėjo paslaugoje, būtų susipažinę su privatumo politika, iš anksto įmonės parengtomis tyrimo pagalbinėmis priemonėmis ir apskritai su siūlomomis paslaugomis, taip galėtų greičiau atlikti tyrimo procesą, geriau suvokdami dalykinę sritį. Suprantama, kad tyrėjai niekaip neturės galimybių susipažinti su kiekvieno debesies paslaugos teikėjo produktais, tačiau gilesnis susipažinimas bent jau su populiariausių teikėjų paslaugomis yra naudingas.

4.2. Identifikacijos etapo papildymas

Identifikacijos etapas yra skirtas paties incidento pirminiam nagrinėjimui, siekiant nustatyti galimas tyrimo kryptis bei artefaktus - potencialius įkalčius.

Kiekvieno incidento tyrimo debesų kompiuterijos sistemose atveju būtų galima iškart atsakyti į klausimus:

- Kokio pobūdžio yra incidentas, ar jis įvykdytas iš kliento pusės, ar galimai buvo vidinės veiklos pasekmė?
- Kokio modelio yra paslauga ar paslaugos, kuriose reikalingas tyrimas: *SaaS*, *PaaS* ar *IaaS*?
- Kokie artefaktai būtų naudingi informacijai apie incidentą gauti?

- Kokiu būdu būtų galima artefaktus pasiekti, kokios prieigos yra reikalingos?

Tam tikrais atvejais, tyrimą galima atlikti ir iš kliento pusės, pavyzdžiui tiriant mobilųjį telefoną ar kompiuterį, kuriuos naudojant buvo atlikta nusikalstama veika, tarkim, *SaaS* modelio elektroninio pašto paslaugoje. Tyrėjas galėtų pareikalauti tam tikrų prisijungimų: prie vartotojo profilio ar prie konkrečių virtualių mašinų.

4.3. Apsaugojimo ir surinkimo etapo papildymai, nuolatinis dokumentavimas

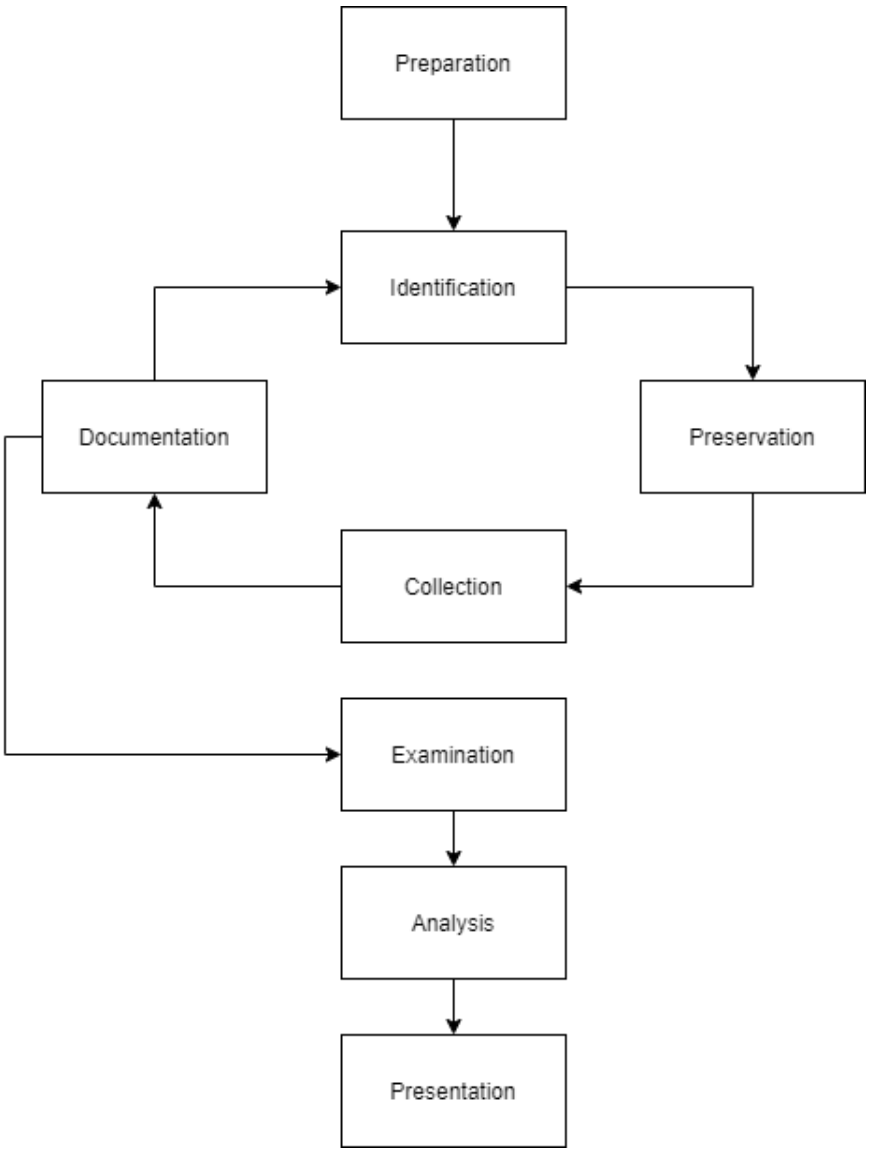
Pavykus identifikuoti artefaktus, tyrėjas savo įgaliojimų ribose galėtų pats atlikti surinkimo ir apsaugojimo darbus - fizinis įrenginių konfiskavimas, su prisijungimais prie kompromituotų paslaugų, tam tikrų virtualių mašinų sustabdymas arba momentinių kopijų surinkimas.

Jeigu yra įtariamas tam tikrų duomenų ištrynimas, tyrėjui gali pavykti juos atkurti atsarginių kopijų pagalba, kurias, galbūt, gali suteikti paslaugos teikėjas. Taip pat tyrėjas gali pareikalauti tam tikro laikotarpio teikėjo žurnalų archyvo prieigos arba pareikalauti teikėjo pateikti konkretaus kliento veiklos žurnalus, taip nepažeidžiant kitų klientų privatumo teisių.

Tyrėjui vykdant surinkimo etapą yra itin svarbu tiksliai aprašyti veiklos žingsnius, tad būtinas nuolatinis dokumentavimas. Tiriant kiekvieną virtualią aplinką, idealiu atveju, naudojami įrankiai turėtų sugeneruoti veiklos raporto lapą automatiškai, taip palengvinant tyrėjo darbą. Jeigu nepavyksta automatiškai dokumentuoti, tyrėjas turėtų tą daryti rankiniu būdu. Svarbu paminėti, kad nors ir veiklos dokumentavimo etapas nėra išreikštas atskirai, bet jis vyksta nuolat kiekviename etape.

4.4. Iteracinis debesies tyrimo modelis

Atsižvelgiant į minėtus iššūkius, atliekant skaitmeninį tyrimą debesų kompiuterijos sistemose, bei į pasiūlytus papildymus, siūloma, kad tyrimo procesas turėtų būti iteratyvus. Kadangi debesies tipo produktų pagrindas yra virtualizacijos technologijos, o itin dažnas reiškinys yra kelių lygių virtualizacijos, reikėtų tirti kiekvieną aptiktą virtualią aplinką atskirai, nes paprastai iš išorės suvokti virtualizacijoje esantį turinį ir iš anksto numatyti visus artefaktus nėra paprasta.



5 pav. Skaitmeninės ekspertizės tyrimo debesų kompiuterijos sistemose proceso modelis (autoriaus iliustracija)

Vadovaujantis pateiktu proceso modeliu [žr. 5 pav.], tyrėjui suteikiama aiškesnės tyrimo eigos suvokimo galimybė. DIP modelyje įkalčių identifikavimas, apsaugojimas ir surinkimas seka vienas po kito, tačiau debesų kompiuterijos sistemose dažnai neįmanoma nustatyti visų tiriamų artefaktų iš anksto, tad vykdant šiuos etapus iteracijomis, sudaroma galimybė turėti aiškesnę dokumentaciją ir įkalčių hierarchiją. Dėl to ekspertizės ir analizės (šiuos etapus nebūtinai vykdo tas pats asmuo) žingsniai yra paprastesni, nes tyrėjui pateikta medžiaga yra aiškesnė ir nuoseklesnė nei viena didelė įkalčių aibė.

Be to, papildomas dokumentavimo žingsnis išskirtinai pabrėžia vieną iš esminių tyrimo žingsnių, būtiną paskutiniajam - pristatymo - etapui. Šį žingsnį būtina išreikšti atskirai, nes iš anksto nežinant, kiek yra virtualizacijos lygių, sunku nuosekliai ir tvarkingai dokumentuoti.

5. Literatūroje siūlomi techniniai problemų sprendimai

Šiame skyriuje aptarsime porą konkrečių literatūroje siūlomų skaitmeninės ekspertizės debesų kompiuterijos sistemose problemų sprendimų, jų privalumus ir trūkumus.

5.1. Artefaktų identifikavimas

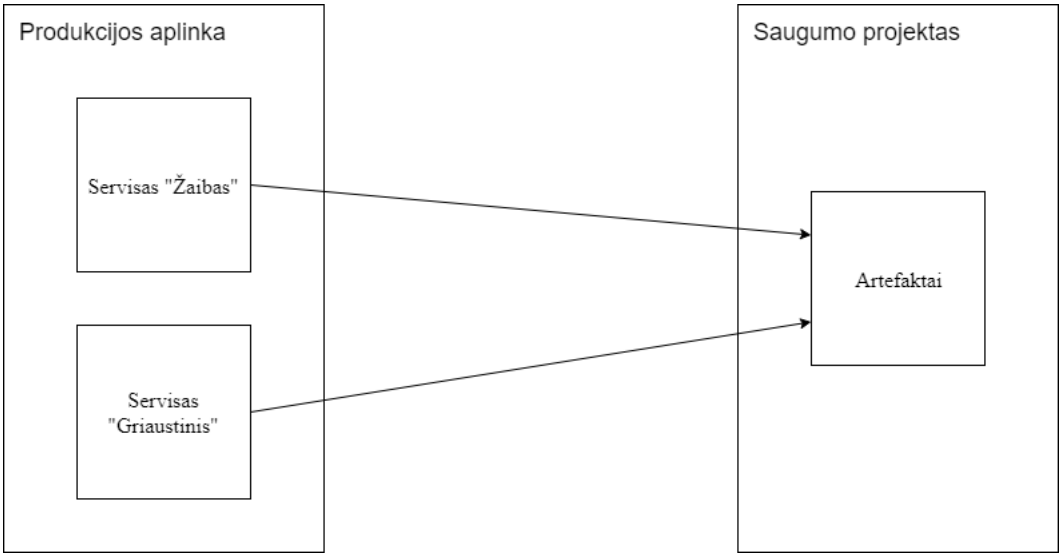
Veiksmų planas yra bene svarbiausias komponentas reagavimui į įvykusį incidentą. Organizuotumas ir pasiruošimas yra kritiniai veiksniai incidento identifikavimo, likvidavimo ir įkalčių rinkimo procese, sumažinantys klaidų ir panikos tikimybę. Svarbu ne tik aiškiai numatyti incidento reagavimo veiksmus, bet ir svarbius surinkti duomenis, artefaktus. Sklandžiam tyrimui yra būtini konkretūs ir esminiai duomenys, optimalus naudingos informacijos rinkinys be šiukšlių (nereikalingos informacijos), galimai trukdysiančių tyrėjui ir ištęsiančių tyrimą. Įkalčių surinkimas ir incidento tyrimas yra tarsi adatos ieškojimas šieno kupetoje, tad svarbu pasiruošti taip, kad šieno kupeta būtų kiek įmanoma mažesnė [**Zuh18**]. Artefaktai, kaupiantys tyrimui būtiną informaciją, yra:

- Žurnalai (angl. Logs) – saugoma sistemos veiklos istorija, įvykiai, klaidos. Tai yra duomenys, kurie nėra išsaugoti atmintyje – fiksuojami tik veiklos pėdsakai. Žurnaluose saugomas milžiniškas kiekis informacijos, tad svarbu ją tvarkingai grupuoti ir turėti galimybę taikyti norimą filtravimą. Keletas svarbių žurnalų grupių:
  - Platformos žurnalai:
    - Administratoriaus veikla. Administratorius turi visas prieigos teises sistemoje. Jokia kita sistemos rolė neturi tiek galios, tad būtent šio profilio veiklą stebėti yra svarbiausia. Vienas pagrindinių įsibrovėlių tikslų – gauti administratoriaus teises, siekiant modifikuoti sistemos konfigūraciją ar paleisti kenksmingą programą;
    - Duomenų prieiga. Duomenų rinkiniai yra vertinga informacija įsibrovėliui, todėl be galo svarbu stebėti kas ir kada pasiekė tam tikrus duomenis;
  - Vartotojo (aplikacijos) generuoti žurnalai;
  - Srauto žurnalai. Įjungiami rankiniu būdu, fiksuoja einamąją veiklą tinkle – užklausas, siunčiamą informaciją, agentus. Tinklo srautas yra „triukšmingas“, tad nuolatinis jo žurnalizavimas sugeneruotų daug nereikalingos informacijos ir brangiai kainuotų. Neįprasta tinklo veikla dažniausiai būna periodinė, pvz.: nuolatinės užklausos iš nepatikimo agento, periodiškai siunčiami neaiškūs paketai.
- Laikmenos, diskai (angl. Disks) – saugomi sistemos failai. Tradicinėse sistemose diskus kaip įkalčius galima pasiimti fiziškai ir atlikti tyrimą. Kitas būdas gauti šį artefaktą – paruošti momentinę sistemos kopiją.
- Einamuoju momentu generuojama informacija – informacija apie šiuo metu vykdomą veiklą sistemoje, siunčiamus ir gaunamus paketus, ryšius tinkle bei prisijungusius vartotojus.

Gavę šiuos artefaktus, tyrėjai turi pakankamai daug medžiagos atlikti savo darbą ir rinkti įkalčius. Bet lieka neišspręstų problemų – kaip paimti šiuos artefaktus, nepažeidžiant įstatymų ar kitų taisyklių, kaip užtikrinti jų validumą ir vientisumą, kur juos saugoti, kokioje aplinkoje atlikti tyrimą?

5.2. Konkretus sprendimas: infrastruktūros projektavimas - apsaugos projektas, saugykla (GRR)

Norint nesunkiai gauti artefaktus ir su jais dirbti, reikia turėti iš anksto parengtą debesies infrastruktūrą, kuri būtų tinkama tyrimui. Vienas iš siūlomų sprendimų yra debesies teikėjui turėti paruoštą atskirą saugumo projektą — saugyklą (žr. 5 pav.)[**Zuh18**][**RHG+16**].



6 pav. Apsaugos projektas (autoriaus iliustracija)

Ši aplinka – izoliuota, turinti griežtai apribotą fizinę ir programinę prieigą. Jos egzistavimo faktas turėtų būti slaptas. Šioje aplinkoje būtų sudiegta reikalinga įranga tirti bei gauti tiriamų sistemų kopijas, žurnalus, turėtų galimybę stebėti einamuoju metu tiriamoje sistemoje vykdomą veiklą bei eksportuoti duomenis tinkama forma pateikimui teisme. Apie galimą veiklą su kiekvienu iš minėtų artefaktų:

- Žurnalams saugoti verta naudoti produktus, leidžiančius laisva forma filtruoti bei grupuoti duomenis. Tokių produktų pavyzdžiai: Google Stackdriver logging, ElasticSearch, Google BigQuery. Visų žurnalų laikyti tyrimo mašinoje neapsimoka, vertėtų išskirti servisą atskirai žurnalų saugojimui bei transportavimui. Duomenų kiekiai žurnaluose yra dideli, efektyvesnis procesas būtų iš žurnalų saugyklos siųsti reikalingus duomenis saugyklos aplinkai, o iš ten pagal poreikį eksportuoti. Pabrėžtina, kad periodinis žurnalų archyvavimas yra įprastas ir reguliarus procesas, tad tyrėjo veikla (šiuo atveju – žurnalinių duomenų filtravimas ir apdorojimas) tiriamojoje aplinkoje nesukels įsibrovėliui įtarimų, kad jo veikla jau yra stebima.
- Laikmenos yra bene esminis artefaktas, nes praktiškai leidžia tirti kompromituotą sistemą „tiesiogiai“. Kadangi fizinės prieigos prie debesies diskų neturime, galime daryti tik jų kopijas. Tokiai veiklai atlikti debesies infrastruktūra turi turėti paruoštą API, kuris leistų pasirinkti, kokių instancijų (serviso, mašinos, t.t.) kopijų norime, gautų susietų diskų sąrašą failo forma grąžintų momentines kopijas, paruoštas naudoti tyrimo aplinkoje. Toks procesas irgi galėtų būti reguliarus atsarginių sistemos kopijų saugojimui, tad atliekama sistemos kopijos darymas neišduotų vykdomo tyrimo.

- Sistemos veikla einamuoju laiku gali atskleisti labai daug tyrimui naudingos informacijos. Ją pasiekus, poreikis skaityti žurnalus sumažėja, paprasčiau rinkti įkalčius sistemos kopijoje. Problema, kad šią informaciją pasiekti yra sudėtinga. Prisijungęs prie pažeistos aplinkos asmuo greičiausiai „padovanos“ savo prisijungimo duomenis įsibrovėliui bei galimai paveiks nusikaltėlio veiksmus – tai gali pakenkti tyrimui. Stebėti sistemos veiklą galima naudojant agentą, pvz.: Google Rapid Response (GRR). GRR suteikia galimybę stebėti aplinkos veiklą (naršyklės istoriją, gauti failo metaduomenis ar gauti nepastoviųjų duomenų kopiją). Šis įrankis taip pat suteikia galimybę ieškoti grėsmių ar spragų sistemoje, tai yra ypač naudinga kai sistema yra nuolat užpuolama ir nepavyksta išsiaiškinti to priežasčių. GRR irgi gali būti sukonfigūruota reguliariam veikimui, pagrįstam užduotimis (angl. Tasks).

Saugyklos arba saugumo projekto turėjimas debesies infrastruktūroje išsprendžia prieigos prie kompromituotos sistemos problemą, suteikia galimybę gauti norimus artefaktus, sukuria patogią aplinką ir sertifikuotus įrankius tirti. Taip pat yra suteikiama galimybė atlikti tyrimą neišsiduodant apie tai įsibrovėliui, tačiau net ir turint kokybišką ir patogią infrastruktūrą, kokybiškam darbui yra būtinas aiškus planas ir kvalifikuotas personalas.

Šis infrastruktūros modelis taip pat leidžia patogiai paruošti įkalčius. Galimybė diskų kopijas, žurnalus ir einamojo momento informaciją išeksportuoti norimu formatu, prieš tai ją apdirbus vienoje vietoje, leidžia patogiai juos pateikti kaip įkalčius teisme. Be to, saugumo projekte vykdoma veikla taip pat galėtų būti žurnalizuojama bei iš tų duomenų automatiškai sugeneruojamas įkalčių rinkimo sekos dokumentas, papildant reikalingomis ataskaitomis ir išvadomis.

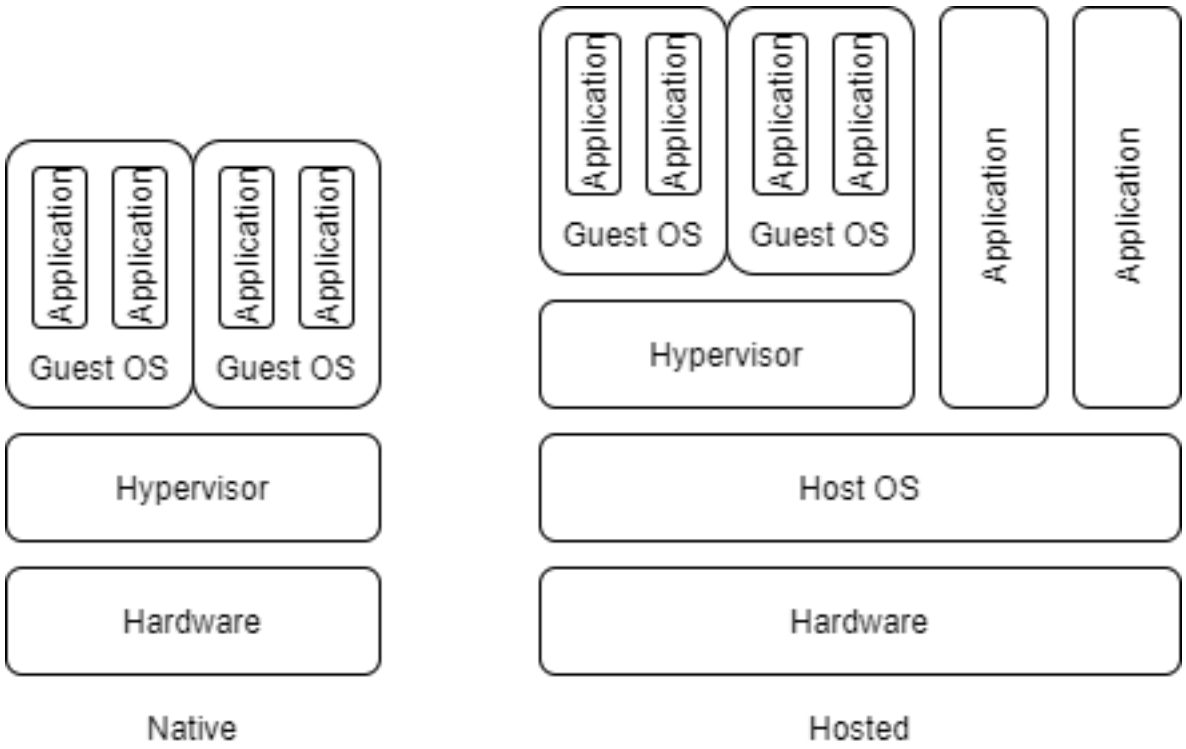
Šio sprendimo esminis trūkumas yra pasiruošimo etapo būtinumas: aplinkos projektavimas, reikalingų įrankių išankstinis diegimas. Tai jau yra papildomos išankstinės išlaidos, kurių reikalaujama iš debesies paslaugos teikėjo. Kiekvienas teikėjas tokį sprendimą gali parengti savaip, tad tyrėjui gali būti sudėtinga naudotis tokiu sprendimu skirtingų teikėjų sistemose.

Be to, agento naudojimas teisme gali būti interpretuotas kaip kenksmingos programinės įrangos panaudojimas skaitmeninės ekspertizės tyrime [**MCC+19**], tad būtina išsiaiškinti visus teisinius klausimus prieš rengiant tokį sprendimą.

5.3. Konkretus sprendimas: virtualių mašinų introspekcija (savistaba)

Kitas gan populiarus sprendimo siūlymas yra virtualių mašinų introspekcija (savistaba), toliau - VMI, suteikianti galimybę mašinas tirti tiek gyvai, tiek statiškai. VMI suteikia galimybę stebėti ir analizuoti stebimosios mašinos programinę įrangą bei tinklo srautą iš hipervizoriaus [**XLW+16**].

Esminė sprendimo idėja yra virtualias mašinas paleisti kitoje, vadinamojoje „šeimininkėje“, mašinoje [**Zil16**] (žr. 7 pav.). Kadangi šeimininkė mašina turi prieigą prie hipervizoriaus, galima gauti labai žemo lygio informaciją.



7 pav. Virtualios mašinos introspekcijos (VMI) vizualizacija (autoriaus iliustracija)

„Šeimininkėje“ mašinoje turi būti įdiegta atitinkama VMI įranga, populiariausia - LibVMI. Įvairūs atminties tyrimo įrankiai (pvz.: Volatility, Helix ISO) atlieka užklausas VMI įrangai, kuri transformuoja užklausas reikiamu formatu stebimai virtualiai mašinai - tarsi užklausa būtų vykdoma iš jos pačios.

Pavyzdžiui, veiksmų seka teikiant užklausą Kernel simboliui gauti, naudojant LibVMI (*http://libvmi.com/docs/gcode-intro.html*):

1. VMI aplikacija pateikia užklausą Kernel simboliui gauti,
2. LibVMI randa Kernel simbolio virtualų adresą,
3. Kernel kategorijų puslapių kataloge randa reikalingo puslapio rodyklę,
4. Rodyklės adresu esantis puslapis grąžinamas LibVMI,
5. LibVMI grąžina duomenis VMI aplikacijai.

Šio sprendimo privalumai [**Zil16**]: nekeičiama stebima mašina, sunkiai pastebimas stebėjimas, galimybė pasiekti labai daug informacijos apie stebimą mašiną, galimybė naudoti egzistuojančius analizės įrankius bei galimybė virtualią mašiną tirti tiek gyvai, tiek statiškai.

Deja, kaip ir anksčiau aprašytame sprendime, šis taip pat reikalauja išankstinio pasiruošimo. Reikia daug konfigūracinių darbų, norint paruošti tokią aplinką, iš anksto sudiegti ir susieti reikalingus įrankius. Taip pat VMI vykdymui reikia daug papildomų išteklių [**Zil16**]. Nėra aišku, kiek toks sprendimas gali paveikti stebimos mašinos greitaveiką [**TRR16**].

VMI turi ir kitų rimtų problemų, kaip, pavyzdžiui, „semantinis tarpas“. Neapdoroti duomenys yra saugomi kitokiu formatu nei atvaizduojami, tad VMI privalo atlikti identiškas duomenų transformacijos operacijas kaip ir operacinė sistema. Iš to seka, kad reikalingos tiriamosios mašinos operacinės sistemos žinios.

Be to, VMI metodas susiduria su kritiškais saugumo klausimais: vartotojų atskyrimas, privatumo klausimai, naudojamų įrankių saugumas. Pati „šeimininkė“ mašina gali būti labai vertingu atakos taikiniu.

6. Skaitmeninių įkalčių debesų kompiuterijos sistemose surinkimo įrankio prototipas, eksperimentas modeliuojamoje aplinkoje

Literatūros apžvalgos metu rasti siūlomi sprendimai atlikti debesų kompiuterijos sistemų ekspertizę dažniausiai reikalauja išankstinio pasiruošimo - infrastruktūros parengimo, įrankių diegimo, konfigūracinių darbų. Taip yra sukuriama atsakomybė debesų kompiuterijos paslaugų teikėjui paruošti tyrimui tinkamą aplinką, kas, deja, nėra finansiškai naudinga ir to nėra reikalaujama.

Šiuo metu jaučiamas didelis trūkumas įrankių, leidžiančių virtualias mašinas tirti nežinant jokios informacijos apie jas bei kitaip nekonfigūruojant aplinkos. Svarbu paminėti, kad įrankių po incidento diegti į tiriamą virtualią mašiną negalima, nes būtų pažeistas įkalčių nekintamumo principas.

Pasirinkta kurti įrankį, skirtą *IaaS* modelio sistemoms, vadovaujantis klasikiniu DIP modeliu, skiriant dėmesį pačiam pirmajam - įkalčių identifikavimo - žingsniui. *IaaS* modelio sistemas tirti nuspręsta dėl didžiausio kontrolės lygio iš visų debesies paslaugų modelių, o identifikacijos žingsnis pasirinktas, nes debesyje identifikuoti įkalčius yra viena iš esminių skaitmeninės ekspertizės debesų kompiuterijos sistemoms problema. Nežinant nieko apie tiriamą virtualią mašiną yra sudėtinga ir kainuoja daug laiko gauti specifinę informaciją apie jos vidų - kas joje saugoma, kokie procesai veikia, ar ta mašina pati virtualizuoja, ką galime pasakyti apie tiriamą virtualią mašiną - tėra keletas klausimų, į kuriuos turi atsakyti tyrėjas.

Šiame skyriuje pristatomas sukurtas skaitmeninių įkalčių surinkimo debesų kompiuterijos sistemose įrankio prototipas, skirtas automatiškai tirti virtualią mašiną, naudojant SSH prisijungimą. Įrankis identifikuoja mašinoje veikiančius Docker konteinerius bei pateikia jų metaduomenis, taip pat pateikia informaciją apie operacinę sistemą bei apie pačią virtualią mašiną.

Svarbu paminėti, jog eksperimento metu galioja prielaidos:

- Tyrėjui legaliai suteikta teisė tirti virtualią mašiną,
- Tyrėjas turi šakninės prieigos SSH mandatus.

Nelegaliai surinkti įkalčiai teisme pripažįstami negaliojančiais, o neturint SSH mandato - nėra galimybės prisijungti prie tiriamos virtualios mašinos.

6.1. Pasirinktos technologijos

Virtualios mašinos virtualizavimui panaudotas Oracle VM VirtualBox produktas, virtualios mašinos operacinė sistema - Ubuntu 20.04 (64-bit). Įrankis sukurtas naudojant Python programavimo kalbą. Technologijų pasirinkimo priežastys:

- VirtualBox pasirinktas dėl to, kad tai yra nemokamas produktas, sukurtas patikimos Oracle korporacijos bei gali virtualizuoti skirtingų operacinių sistemų virtualias mašinas. Produktas yra paprastas naudoti ir suteikia geras rekomendacines gaires,
- Ubuntu 20.04 pasirinktas dėl to, kad Linux operacinių sistemų šeimos virtualios mašinos yra vienos populiariausių dėl savo greičio ir lankstumo. Ši Ubuntu versija šiuo metu yra

rekomenduojama naudoti, dar nenutrauktas jos palaikymas,

- Python programavimo kalba pasirinkta dėl plačios siūlomų įrankių aibės bei paprastumo. Taip pat Python programos nėra priklausomos nuo operacinės sistemos, jų paleidimas yra labai paprastas.

Technologijos nėra nišinės ir yra plačiai naudojamos skaitmeninių tyrimų profesionalų bendruomenėje.

6.2. Modeliuojama aplinka

VirtualBox produktas atsiųstas iš oficialaus internetinio puslapio ir įdiegtas į darbo kompiuterį, iš *https://www.osboxes.org/ubuntu/* atsiųstas Ubuntu 20.04 (64-bit) virtualios mašinos failas. Paleidus virtualią mašiną veikti, įdiegtas SSH serveris ir atvertas SSH portas:

```
$ sudo apt install openssh-server
$ sudo ufw allow ssh
```

Virtualios mašinos SSH prisijungimo IP: localhost, prisijungimo vardas: osboxes, slaptažodis: osboxes.org. Virtualioje mašinoje įdiegtas Docker Engine:

```
$ sudo apt-get update
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg
    | sudo apt-key add -
$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

Įdiegus Docker, iš bendruomenės puslapio įdiegtas HTTP serverio Docker konteineris (*https://hub.docker.com/r/danjellz/http-server*)

```
$ docker run -d -p 8080:8080 -v $(PWD):/public danjellz/http-server
```

Ši modeliuojama aplinka praktiškai atitinka *IaaS* modelio paslaugą - vartotojas turi galimybę paleisti savo norimą virtualią mašiną, joje įdiegti norimus įrankius ir atlikti norimą veiklą.

6.3. Įrankio veikimas ir eksperimentas

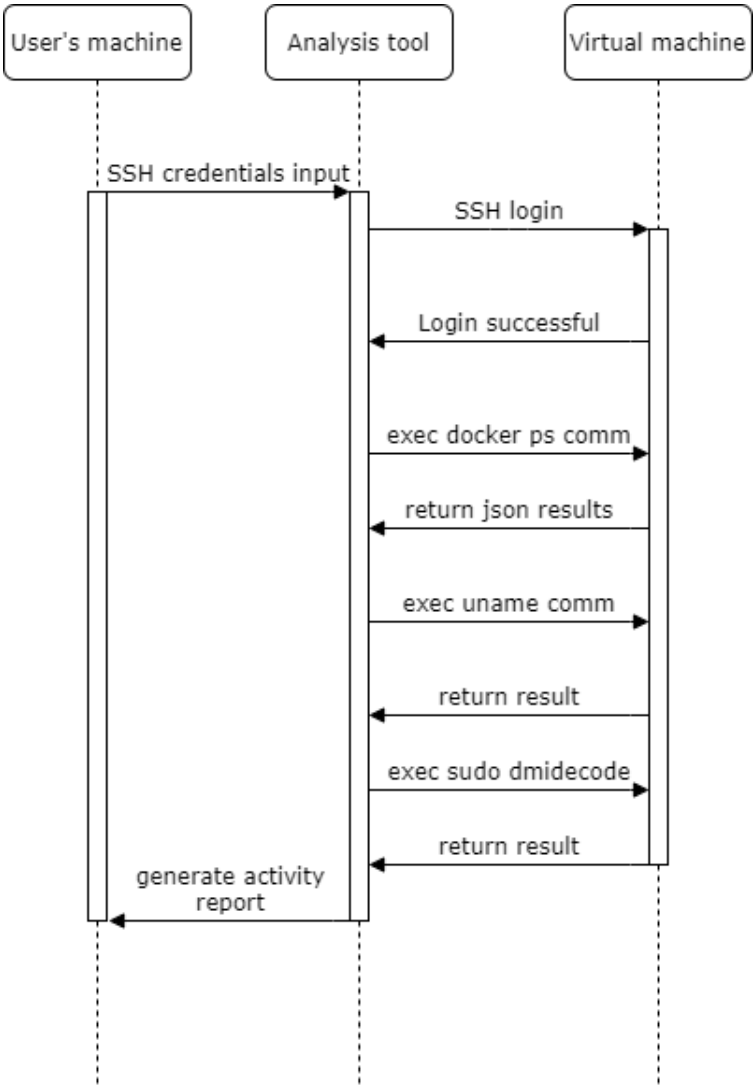
Sukurto įrankio tikslas - be jokio išankstinio pasiruošimo automatiškai atlikti paviršutinišką mašinos analizę - surinkti informaciją apie ją ir apie operacinę sistemą, ištirti, ar šioje mašinoje yra virtualizuojamų Docker konteinerių ir, jei taip, gauti metaduomenis apie juos.

Kadangi skaitmeninio tyrimo procesas reikalauja vykdomo tyrimo ataskaitų, įrankis automatiškai sugeneruoja raportą, kuriame atvaizduojama gauta informacija bei įrankio veiklos žurnalas su UTC laiko žymomis.

Įrankio paketą sudaro:

- Programa *VMRunningDockerAnalysisTool.py*,
- Raporto šablonas *report-template.html*.

Įrankio veiklos seka ir komponentų sąveika pavaizduota 8 pav.



8 pav. Įrankio veiklos seka (autoriaus iliustracija)

Įrankis paleidžiamas per terminalą (demonstracijoje naudojamas Windows terminalas) (žr. 9 pav.):

```
C:\Users\Audrius\Desktop\Bakalaurinis\Tool>VMRunningDockerAnalysisTool.py
You are running Virtual Machine's inspection tool, current UTC date is 2020/05/25
Enter Virtual Machine's IP address: localhost
Enter SSH Username: osboxes
Enter SSH Password:
[2020-05-25 02:30:44.325752]: Started to establish SSH connection to localhost for user osboxes
[2020-05-25 02:30:46.885873]: SSH connection to localhost for user osboxes established successfully
[2020-05-25 02:30:46.885873]: running command: docker ps --no-trunc --format='{json .}'
[2020-05-25 02:30:46.957119]: running command: uname -a
[2020-05-25 02:30:46.963952]: running command: sudo -S dmidecode -t system
[2020-05-25 02:30:47.006898]: Generating report
[2020-05-25 02:30:47.048942]: Report generated, file 20200525023047_Report.html
```

9 pav. Įrankio veikimo demonstracija

Paskutinis įrankio darbo žingsnis - raporto lapo generavimas, kuris išsaugomas paketo aplanke. Sugeneruotas raportas (žr. 1 priedą) turėtų būti panaudotas kaip dalis bendros tyrimo ataskaitos teisme, parodantis dalį tyrimo eigos.

6.4. Eksperimento įvertinimas

Eksperimento metu pavyko automatiškai surinkti informaciją apie tiriamąją virtualią mašiną bei identifikuoti konkrečius šiuo metu veikiančius Docker konteinerius bei gauti jų metaduomenis.

Sukurtas įrankis nors ir yra nedidelis, tačiau gali sutaupyti nemažai laiko tyrėjui, suteikiant naudingos informacijos apie tiriamąjį objektą ir potencialius įkalčius. Suvokiant milžiniškas debesų kompiuterijos sistemų apimtis, šio įrankio pagalba tyrėjas galėtų greitai atlikti grubią pirmo lygio virtualizacijų analizę, taip nesunkiai susiaurindamas tiriamųjų objektų aibę.

Esminis įrankio trūkumas šiuo metu yra jo nelankstumas - kol kas jis nėra pritaikytas analizuoti kitų nei Linux šeimos operacinių sistemų virtualizacijas. Taip pat sugeneruotas raporto lapas galėtų būti sistematiškesnis ir tvarkingiau apdorotų kai kuriuos gautus duomenis, idealu jeigu vienas raporto lapas galėtų tęstis įrankiui tiriant kitas aplinkas,

Įrankio plėtros galimybės:

- Vizualizacijų parengimas. Įrankis galėtų vizualiai pademonstruoti virtualizacijų hierarchiją komponentais ar medžiu;
- Automatinis prisitaikymas prie tiriamos virtualios mašinos operacinės sistemos. Teikiamos užklausos skiriasi skirtingose operacinėse sistemose, tad reikėtų jas atitinkamai pritaikyti;
- Vartotojo sąsajos parengimas;
- Lygiagretus veikimas tiriant keletą virtualių mašinų;
- Kito tipo konteinerių identifikavimas;
- Virtualizacijų failų paieška ir metaduomenų surinkimas;
- Raporto lapo elektroninio pasirašymo implementacija;

Sukurtą įrankį padarius universalesnį, jis būtų tikrai naudingas tyrėjams ir ženkliai sutrumpintų identifikacijos etapo vykdymo laiką.

Rezultatai

Darbe pasiekti **rezultatai**:

1. Apžvelgus literatūrą, apibrėžta debesų kompiuterijos technologija ir išnagrinėti skirtingi paslaugų modeliai, charakteristikos, apibrėžta skaitmeninio tyrimo modelio struktūra bei įkalčių charakteristika.
2. Įvardinti skaitmeninės ekspertizės ir reagavimo į incidentą procesų iššūkiai debesų kompiuterijos sistemose: teisiniai neaiškumai, duomenų nevientisumas bei lokacijos nekonkretumas, priklausomybė nuo debesies paslaugos teikėjo bei debesies paslaugų teikėjų priklausomybių grandinė, įkalčių nepastovumas, milžiniškas duomenų kiekis tyrėjui, laiko sinchronizacijos iššūkiai bei įrankių trūkumas.
3. Atliktus literatūroje siūlomų sprendimų apžvalgą, pateiktas keleto jų kokybinis įvertinimas.
4. Pateikti siūlomi skaitmeninės ekspertizės ir reagavimo į incidentą proceso debesų kompiuterijos sistemose patobulinimai, pasiūlytas tyrimo proceso modelis.
5. Sukurtas skaitmeninių įkalčių surinkimo debesų kompiuterijos sistemose įrankio prototipas, orientuotas į skaitmeninio tyrimo identifikavimo etapą, *IaaS* modelio paslaugų klientinę pusę. Prototipas SSH protokolu gali prisijungti prie Linux šeimos virtualių mašinų ir automatiškai surinkti metaduomenis apie ją bei identifikuoti veikiančius Docker konteinerius.
6. Atliktas prototipo bandymas (eksperimentas) modeliuojamoje aplinkoje, sugeneruotas tyrimo raporto lapas.
7. Atliktas prototipo įvertinimas - įvardinti privalumai, trūkumai, panaudojimo scenarijai, galimi plėtros taškai ir iššūkiai.

Išvados ir tolimesni galimi darbai

Išanalizavus rezultatus gautos **išvados**:

1. Debesų kompiuterijos sistemos šiuo metu nėra tinkamai parengtos skaitmeniniam tyrimui vykdyti:
   (a) Dėl debesies technologijų globalumo, valstybės negali įprasta tvarka reikalauti prieigos prie duomenų, esančių įvairiose užsienio valstybėse - trūksta tarptautinių teisinių sprendimų,
   (b) Būtinas griežtesnis teisinis reguliavimas debesų paslaugos teikėjams, būtina reikalauti iš anksto paruošti medžiagą bei aplinką tyrimams, taip leidžiant tyrėjų komandai pasitikėti teikėjų bendradarbiavimu,
   (c) Be alternatyvių tyrimui paruoštų architektūrinių debesies paslaugų sprendimų vykdyti skaitmeninį tyrimą yra labai sudėtinga.
2. Būtina kurti sprendimus ir įrankius, reikalingus skaitmeninei ekspertizei debesų kompiuterijos sistemose - jaučiamas ženklus jų trūkumas,
3. Universalios tyrimo metodikos sukurti neįmanoma, tačiau būtina aiški skaitmeninio tyrimo vykdymo metodologija, skirta debesų kompiuterijos sistemoms. Vienas populiariausių tyrimo vykdymo modelių - DIP modelis - nėra tinkamas skaitmeninio tyrimo vykdymui debesų kompiuterijos sistemose.

Tolimesni galimi darbai:

1. Atlikti daugiau eksperimentų, taikant pasiūlytus skaitmeninės ekspertizės proceso patobulinimus. Eksperimentus atlikti skirtingiems debesies paslaugos modeliams;
2. Sukurtą įrankį plėsti, kad šis veiktų universaliai - gebėtų tirti skirtingų operacinių sistemų virtualizacijas, kito tipo veikiančius konteinerius, aptiktų virtualizacijų failus;
3. Sukurti pasiūlymus teisinėms problemoms spręsti;

Šaltiniai

[AIJ14] ALMULLA, Sameera; IRAQI, Youssef; JONES, Andrew. A state-of-the-art review of cloud forensics. Journal of Digital Forensics, Security and Law, 2014, 9.4: 2.

[CEC17] Kim-Kwang Raymond Choo, Christian Esposito and Aniello Castiglione. Evidence and Forensics in the Cloud: Challenges and Future Research Directions. *IEEE Cloud Computing*, 4(3), 2017, 14-19 psl.

[CMK+17] Cahyani, Niken Dwi Wahyu and Martini, Ben and Choo, Kim-Kwang Raymond and Al-Azhar, AKBP Muhammad Nuh. Forensic data acquisition from cloud-of-things devices: windows Smartphones as a case study. *Concurrency and Computation: Practice and Experience*, 29(14), 2017, e3855 psl.

[Cos19] Katie Costello, Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.5 Percent in 2019. Prieiga internetu: `https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g`, 2019

[DDC17] Daryabar, Farid and Dehghantanha, Ali and Choo, Kim-Kwang Raymond: Cloud storage forensics. MEGA as a case study. *Australian Journal of Forensic Sciences*, 49(3), 2017, 344-357 psl.

[DS11] DYKSTRA, Josiah; SHERMAN, Alan T. Understanding issues in cloud forensics: two hypothetical case studies. UMBC Computer Science and Electrical Engineering Department, 2011.

[FLM+11] FEHLING, Christoph, et al. A collection of patterns for cloud types, cloud service models, and cloud-based application architectures. University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany, University of Stuttgart, Institute of Architecture of Application Systems, Technical Report Computer Science, 2011, 5: 19.

[GER+12] GIBSON, Joel, et al. Benefits and challenges of three cloud computing service models. In: 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN). IEEE, 2012. p. 198-205.

[GSG12] G. Grispos, T. Storer, W. B. Glisson. Calm Before the Storm: The Challenges of Cloud Computing in Digital Forensics. *International Journal of Digital Crime and Forensics (IJDCF)*, 4(2), 2012, 28-48 psl. Prieiga internetu: `https://arxiv.org/ftp/arxiv/papers/1410/1410.2123.pdf`

[Joh17] Gerard Johansen. *Digital Forensics and Incident Response: A practical guide to deploying digital forensic techniques in response to cyber security incidents*, Packt Publishing Ltd., 2017

[MCC+19] MANRAL, Bharat, et al. A Systematic Survey on Cloud Forensics Challenges, Solutions, and Future Directions. ACM Computing Surveys (CSUR), 2019, 52.6: 1-38.

[MG11] Stephen Mason, Esther George. Digital evidence and 'cloud' computing. *Computer Law & Security Review*, 27(5), 2011, 524-528 psl.

[MG11a] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. Prieiga internetu: `http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf`, 2011

[MSH08] Michael B. Mukasey, Jeffrey L. Sedgwick, David W. Hagy. Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition. Prieiga internetu: `https://www.ncjrs.gov/pdffiles1/nij/219941.pdf`, 2008

[Peč11] Angelė Pečeliūnaitė. Debesų kompiuterija: darbas, bendradarbiavimas ir komunikacija. Ar debesis tenkina studentų ir mokslininkų poreikius? *Informacijos mokslai*, 55, 2011

[PLS15] PICHAN, Ameer; LAZARESCU, Mihai; SOH, Sie Teng. Cloud forensics: Technical challenges, solutions and comparative analysis. Digital investigation, 2015, 13: 38-57.

[RCK+11] RUAN, Keyun, et al. Cloud forensics. In: IFIP International Conference on Digital Forensics. Springer, Berlin, Heidelberg, 2011. p. 35-46.

[RCK+13] RUAN, Keyun, et al. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. Digital Investigation, 2013, 10.1: 34-43.

[RHG+16] Nurul Hidayah Ab Rahman, William Bradley Glisson, Yanjiang Yang, Kim-Kwang Raymond Choo. Forensic-by-design framework for cyber-physical cloud systems. *IEEE Cloud Computing*, 3(1), 2016, 50-59 psl.

[RNN+17] Ab Rahman, Nurul Hidayah and Cahyani, Niken Dwi Wahyu and Choo, Kim-Kwang Raymond. Cloud incident handling and forensic-by-design: cloud storage as a case study. *Concurrency and Computation: Practice and Experience*, 29(14), 2017, e3868 psl.

[RZH+16] RAJASEKARAN, Sundaresan, et al. Scalable cloud security via asynchronous virtual machine introspection. In: 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16). 2016.

[Sam12] John Sammons, *The Basics of Digital Forensics (Second Edition)*, 2015

[SEK19] SPIEKERMANN, Daniel; EGGENDORFER, Tobias; KELLER, Jörg. A Study of Network Forensic Investigation in Docker Environments. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019. p. 1-7.

[THG+10] M. Taylor, J. Haggerty, D. Gresty, R. Hegarty. Digital evidence in cloud computing systems, *Computer Law & Security Report*, 26(3), 2010, 304-308 psl.

[TLC+17] TIEN, Chin-Wei, et al. Memory forensics using virtual machine introspection for Malware analysis. In: 2017 IEEE Conference on Dependable and Secure Computing. IEEE, 2017. p. 518-519.

[TRR16] TAUBMANN, Benjamin; RAKOTONDRAVONY, Noëlle; REISER, Hans P. Cloudphylactor: Harnessing mandatory access control for virtual machine introspection in cloud data centers. In: 2016 IEEE Trustcom/BigDataSE/ISPA. IEEE, 2016. p. 957-964.

[VMC+08] VAQUERO, Luis M., et al. A break in the clouds: towards a cloud definition. 2008.

[Zuh18] Sami Zuhuruddin. *Cloud Forensics 101 (Cloud Next '18)*. Prieiga tinkle: `https://youtu.be/OkjTqlETgMA`, 2018

[WRP18] WAHYUDI, Erfan; RIADI, Imam; PRAYUDI, Yudi. Virtual Machine Forensic Analysis And Recovery Method For Recovery And Analysis Digital Evidence. International Journal of Computer Science and Information Security, 2018, 16.

[XLW+16] XIAO, Jidong, et al. HyperLink: virtual machine introspection and memory forensic analysis without kernel source code. In: 2016 IEEE international conference on autonomic computing (ICAC). IEEE, 2016. p. 127-136.

[ZH13] ZAWOAD, Shams; HASAN, Ragib. Digital forensics in the cloud. ALABAMA UNIV IN BIRMINGHAM, 2013.

[Zil16] Zillner, Tobias. *Memory Forensics using Virtual Machine Introspection for Cloud Computing, Black Hat USA*. Prieiga tinkle: `https://www.blackhat.com/docs/us-16/materials/us-16-Zillner-Memory-Forensics-Using-VMI-For-Cloud-Computing.pdf`, 2016

Priedas nr. 1

Sugeneruotas raporto lapas

DIGITAL FORENSICS INVESTIGATION REPORT NO.

1. REPORT DATE	2. INCIDENT DATE	3. REPORT AUTHOR(S)
2020/05/25		

4. JUSTIFICATION OF INVESTIGATION

5. ACTIONS PERFORMED

- Generated by Virtual Machine's inspection tool:
  - Connection to a suspect VM via SSH,
  - OS information gathering,
  - VM information gathering,
  - Running Docker containers detection.

6. IP OF INVESTIGATED VIRTUAL MACHINE	7. SSH USERNAME	8. SSH PASSWORD
localhost	osboxes	osboxes.org

9. MACHINE INFO

b'# dmidecode 3.2\nGetting SMBIOS data from sysfs.\nSMBIOS 2.5 present.\n\nHandle 0x0001, DMI type 1, 27 bytes\nSystem Information\n\tManufacturer: innotek GmbH\n\tProduct Name: VirtualBox\n\tVersion: 1.2\n\tSerial Number: 0\n\tUUID: 41bc6c69-98bd-43d4-8597-af19bd9dd04f\n\tWake-up Type: Power Switch\n\tSKU Number: Not Specified\n\tFamily: Virtual Machine\n\n'

10. OS INFO

b'Linux osboxes 5.4.0-26-generic #30-Ubuntu SMP Mon Apr 20 16:58:30 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux\n'

11. DOCKER PROCESSES

Command	"http-server"
CreatedAt	2020-05-16 14:23:56 -0400 EDT
ID	92c40f7673f4336b577038481927c6de910bf0351e9f5f282b100edbd6f91929
Image	danjellz/http-server
Labels	maintainer=Dan Jellesma
LocalVolumes	1
Mounts	f435fd1a3d38b47bb2c7d3204fa0b6929305e131e925e6ed7efcaf6067b465d9
Names	gracious_euler
Networks	bridge
Ports	0.0.0.0:8080->8080/tcp
RunningFor	8 days ago
Size	0B
Status	Up 8 days

12. ACTIVITY LOG

1. [2020-05-25 02:30:34.795675]: Service started
2. [2020-05-25 02:30:39.238279]: User entered Virtual Machine's IP address, value: localhost
3. [2020-05-25 02:30:41.872309]: User entered SSH username, value: osboxes
4. [2020-05-25 02:30:44.324777]: User entered SSH password, value: osboxes.org
5. [2020-05-25 02:30:44.325752]: Started to establish SSH connection to localhost for user osboxes
6. [2020-05-25 02:30:46.885873]: SSH connection to localhost for user osboxes established successfully
7. [2020-05-25 02:30:46.885873]: running command: docker ps --no-trunc --format='{json .}'
8. [2020-05-25 02:30:46.956145]: command docker ps --no-trunc --format='{json .}' executed successfully, raw result: b'{"Command":"\\"http-server\\"","CreatedAt":"2020-05-16 14:23:56 -0400 EDT","ID":"92c40f7673f4336b577038481927c6de910bf0351e9f5f282b100edbd6f91929","Image":"danjellz/http-server","Labels":"maintainer=Dan Jellesma","LocalVolumes":"1","Mounts":"f435fd1a3d38b47bb2c7d3204fa0b6929305e131e925e6ed7efcaf6067b465d9","Names":"gracious_euler","Networks":"bridge","Ports":"0.0.0.0:8080-\\u003e8080/tcp","RunningFor":"8 days ago","Size":"0B","Status":"Up 8 days"}\n'
9. [2020-05-25 02:30:46.957119]: running command: uname -a
10. [2020-05-25 02:30:46.963952]: command uname -a executed successfully, raw result: b'Linux osboxes 5.4.0-26-generic #30-Ubuntu SMP Mon Apr 20 16:58:30 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux\n'
11. [2020-05-25 02:30:46.963952]: running command: sudo -S dmidecode -t system
12. [2020-05-25 02:30:47.006898]: command sudo -S dmidecode -t system executed successfully, raw result: b'# dmidecode 3.2\nGetting SMBIOS data from sysfs.\nSMBIOS 2.5 present.\n\nHandle 0x0001, DMI type 1, 27 bytes\nSystem Information\n\tManufacturer: innotek GmbH\n\tProduct Name: VirtualBox\n\tVersion: 1.2\n\tSerial Number: 0\n\tUUID: 41bc6c69-98bd-43d4-8597-af19bd9dd04f\n\tWake-up Type: Power Switch\n\tSKU Number: Not Specified\n\tFamily: Virtual Machine\n\n'
13. [2020-05-25 02:30:47.006898]: Generating report

13. LEADING AUTHOR'S SIGNATURE	14. REVIEWER'S SIGNATURE

Priedas nr. 2

Sukurto įrankio kodas

```
1  import paramiko
2  import time
3  import datetime
4  import os
5  import sys
6  import getpass
7  import json2html
8  import json
9
10
11 def ssh_con(ip, un, pw):
12     global ssh, session
13     action = "[{0}]: Started to establish SSH connection to {1} for user {2}".format(
           ↪ datetime.datetime.utcnow(), ip, un)
14     log.append(action)
15     print(action)
16     ssh = paramiko.SSHClient()
17     ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
18     ssh.connect(ip, username=un, password=pw, timeout=200)
19     transport = ssh.get_transport()
20     session = transport.open_session()
21     session.set_combine_stderr(True)
22     session.get_pty()
23     action = "[{0}]: SSH connection to {1} for user {2} established successfully".format(
           ↪ datetime.datetime.utcnow(), ip, un)
24     log.append(action)
25     print(action)
26
27
28 def ssh_exec_command_and_read(command):
29     stdin, stdout, stderr = ssh.exec_command(command)
30     return stdout.read()
31
32
33 def ssh_exec_sudo_command_and_read(command):
34     stdin, stdout, stderr = ssh.exec_command(command)
35     stdin.write(ssh_credentials['password'] + "\n")
36     stdin.flush()
37     return stdout.read()
38
39
40 def parse_docker_table_to_html(docker_table):
41     infoFromJson = json.loads(docker_table)
42     return json2html.json2html.convert(json = infoFromJson)
43
```

```
44
45  def generate_html_from_log():
46      for index, item in enumerate(log):
47          log[index] = "<li>{0}</li>".format(item)
48      dictionary['{LOGITEM}'] = " ".join(map(str, log))
49
50
51  def generate_report():
52      generate_html_from_log()
53      f = open("report-template.html", "r")
54      template = f.read()
55      f.close()
56      for key, value in dictionary.items():
57          template = template.replace(key, str(value))
58      filename = str("{0}_Report.html".format(datetime.datetime.utcnow().strftime("%Y%m%d%H
            ↪ %M%S")))
59      result = open(filename, "w+")
60      result.write(template)
61      result.close()
62      return filename
63
64
65  global dictionary
66  dictionary = {
67      "{REPORTDATE}": "{REPORTDATE}",
68      "{INCIDENTDATE}": "{INCIDENTDATE}",
69      "{IP}": "{IP}",
70      "{SSHUSERNAME}": "{SSHUSERNAME}",
71      "{SSHPASSWORD}": "{SSHPASSWORD}",
72      "{MACHINEINFO}": "{MACHINEINFO}",
73      "{OSINFO}": "{OSINFO}",
74      "{DOCKERTABLE}": "{DOCKERTABLE}",
75      "{LOGITEM}": "{LOGITEM}"
76      }
77  global ssh_credentials
78  ssh_credentials = {'ip': 'localhost', 'username': 'osboxes', 'password': 'osboxes.org'}
79  global log
80  log = []
81
82  log.append("[{0}]: Service started".format(datetime.datetime.utcnow()))
83
84  dictionary['{REPORTDATE}'] = datetime.datetime.utcnow().strftime('%Y/%m/%d')
85  print("You are running Virtual Machine\'s inspection tool, current UTC date is %s" % str(
        ↪ dictionary['{REPORTDATE}']))
86
87  ssh_credentials['ip'] = str(input('Enter Virtual Machine\'s IP address: '))
88  log.append("[{0}]: User entered Virtual Machine\'s IP address, value: {1}".format(
        ↪ datetime.datetime.utcnow(), ssh_credentials['ip']))
89  ssh_credentials['username'] = str(input('Enter SSH Username: '))
```

```
log.append("[{0}]: User entered SSH username, value: {1}".format(datetime.datetime.utcnow
    ↪ (), ssh_credentials['username']))
ssh_credentials['password'] = getpass.getpass('Enter SSH Password: ')
log.append("[{0}]: User entered SSH password, value: {1}".format(datetime.datetime.utcnow
    ↪ (), ssh_credentials['password']))
ssh_con(ssh_credentials['ip'], ssh_credentials['username'], ssh_credentials['password'])

dictionary['{IP}'] = ssh_credentials['ip']
dictionary['{SSHUSERNAME}'] = ssh_credentials['username']
dictionary['{SSHPASSWORD}'] = ssh_credentials['password']

action = "[{0}]: running command: docker ps --no-trunc --format='{{json .}}'".format(
    ↪ datetime.datetime.utcnow())
print(action)
log.append(action)

docker_table = ssh_exec_command_and_read("docker ps --no-trunc --format='{{json .}}'")
action = "[{0}]: command docker ps --no-trunc --format='{{json .}}' executed successfully
    ↪ , raw result: {1}".format(datetime.datetime.utcnow(), docker_table)
log.append(action)
dictionary['{DOCKERTABLE}'] = parse_docker_table_to_html(docker_table)

action = "[{0}]: running command: uname -a".format(datetime.datetime.utcnow())
print(action)
log.append(action)
dictionary['{OSINFO}'] = ssh_exec_command_and_read("uname -a")
action = "[{0}]: command uname -a executed successfully, raw result: {1}".format(datetime
    ↪ .datetime.utcnow(), dictionary['{OSINFO}'])
log.append(action)

action = "[{0}]: running command: sudo -S dmidecode -t system".format(datetime.datetime.
    ↪ utcnow())
print(action)
log.append(action)
dictionary['{MACHINEINFO}'] = ssh_exec_sudo_command_and_read("sudo -S dmidecode -t system
    ↪ ")
action = "[{0}]: command sudo -S dmidecode -t system executed successfully, raw result:
    ↪ {1}".format(datetime.datetime.utcnow(), dictionary['{MACHINEINFO}'])
log.append(action)

action = "[{0}]: Generating report".format(datetime.datetime.utcnow())
print(action)
log.append(action)
result = generate_report()
print("[{0}]: Report generated, file {1}".format(datetime.datetime.utcnow(), result))
```