Kauno technologijos universitetas

Informatikos fakultetas

Kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas

Baigiamasis magistro projektas

Mažvydas Aukščionis

Projekto autorius

Prof. dr. Algimantas Venčkauskas

Vadovas

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas

Baigiamasis magistro projektas

Informacijos ir informacinių technologijų sauga (621E10003)

Mažvydas Aukščionis

Projekto autorius

Prof. dr. Algimantas Venčkauskas

Vadovas

Doc. Audronė Janavičiūtė

Recenzentė

Kaunas, 2020

Kauno technologijos universitetas

Informatikos fakultetas

Mažvydas Aukščionis

Kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Mažvydo Aukščionio, baigiamasis projektas tema „Kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

(vardą ir pavardę įrašyti ranka)	(parašas)

Aukščionis, Mažvydas. Kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas. Magistro studijų baigiamasis projektas / vadovas prof. dr. Algimantas Venčkauskas; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Technologijos mokslai, Informatikos inžinerija.

Reikšminiai žodžiai: Vientisumas, atstatymas, užtikrinimas.

Kaunas, 2020. 65 p.

Santrauka

Šiais laikais vis daugiau informacijos yra talpina skaitmeninėje erdvėje ir yra svarbu, kad patalpinta informacija nebūtų suklastotą ar sugadintą. Kompiuterines sistemos vientisumas garantuoja, kad kompiuterinės sistemos posisteme atliks užduotis taip kaip buvo suprojektavęs jos kūrėjas ir, kad pradiniai duomenys, saugojimo ar perdavimo metu, nebus neleistinai pakeisti. Vientisumo problema yra formuluojama iš prieigos valdymo politikos ir iš mechanizmu, kurie duoda posisteme su izoliacija reikalinga apsaugą nuo informacijos sugadinimo.

Sistemos vientisumas gali būti pažeistas asmenų su kenkėjišku tikslu, pavyzdžiui bandant sunaikinti duomenis ar pakeisti juos klaidinga informacija. Virusai gali pakeisti sisteminius failus ar programas, taip įsitvirtinantys sistemoje, kurioje gali atlikti kenkėjiškus veiksmus. Taip pat aparatinės įrangos vientisumas gali būti pažeistas, pavyzdžiui, kai kurie aparatinės įrangos komponentai gali būti pakeisti kitais, prastesniais arba diskai gali būti išimti įdėti kiti nei, kad nurodyti specifikacijoje. Taip pat vientisumas gali būti pažeistas, kai yra neleistinai įdiegiama papildoma aparatinė įranga. Neleistinai įdiegta įranga gali prieiti prie sistemos resursų ir iš jos ištraukti jautria informacija kaip slaptažodžius, privačiuosius raktus, asmenį identifikuojančius duomenis. Dažniausiai tokie įrenginiai gali dirbti atskirai nuo pagrindinės sistemos ir apsaugos priemones, kaip antivirusinės negali aptikti kenkėjiško kodo, kuris yra patalpintas ir vykdomas kitame procesoriuje. Ypač dabar kai debesų paslaugos kyla aukštyn, dažniausiai neturime prieigos prie aparatines įrangos arba net nežinome, kur yra mūsų tarnybinė stotis.

Magistrinio darbo objektas yra kompiuterinės sistemos vientisumo užtikrinimo modelio sudarymas ir tyrimas.

Šio darbo struktūra:

- Pirmoje darbo dalyje yra aptariama kompiuterinės sistemos vientisumo užtikrinimo problema, kompiuterinės sistemos vientisumo užtikrinimo metodų analizė bei kompiuterinės sistemos vientisumo užtikrinimo produktų analizė;
- Antroje darbo dalyje pateikiama kompiuterinės sistemos vientisumo užtikrinimo modelis, sistemos vizija, architektūra, veikimo principas;
- Trečioje darbo dalyje yra pateikiamas sistemos prototipas, eksperimento tyrimas ir jo rezultatai;
- Darbo pabaigoje yra pateiktos išvados.

Aukščionis, Mažvydas. Development and Research of Model for Computer System Integrity Assurance. Master's Final Degree Project / supervisor prof. dr. Algimantas Venčkauskas; Informatics Faculty, Kaunas University of Technology.

Study field and area (study field group): Technology Sciences, Informatics Engineering.

Keywords: Integrity, Recovery, Assurance.

Kaunas, 2020. 65.

Summary

Nowadays more and more information is being stored in the digital world and it is important that the stored information is not tampered or compromised. Computer system integrity guarantee that the computer subsystem will perform as the creator of the system intended and that the data is not altered when it is in transit or stored on disk. The integrity problem is a formulation of access control policies and mechanisms that provide a system with the isolation necessary for protection from corruption.

System integrity can be compromised by evil actors trying to destroy or manipulate data on the system. Also, malware and viruses can alter system files and programs become persistent in the system and allow people with bad intent to do harmful activities. Hardware integrity can also be compromised by changing component of the system replacing them with inferior ones or by ones that are not to the specification. Also adding unauthorized hardware to the system would qualify as compromised integrity as the unauthorized hardware would have access to the system resources and would be able to extract sensitive information from it, like passwords, private keys and other personally identifiable information. Usually those kinds of hardware can work independent from the host system and security measure like anti-virus programs cannot detect them. Especially now when cloud services are growing rapidly, most of the time we do not know where our system is located, and we do not have access to the hardware.

The objective of this work is to develop and research a model for computer system integrity assurance.

The structure of this work:

- The first part of the work discusses the problem of computer system integrity assurance, the analysis of computer system integrity assurance methods and the analysis of computer system integrity assurance products;
- The second part of the work presents the computer system integrity assurance model, system vision, architecture, operating principle;
- The third part of the work presents a system prototype, experimental study and its results;
- Conclusions are presented at the end of the work.

Turinys

Lentelių sąrašas **7**
Paveikslų sąrašas **8**
Santrumpų ir terminų sąrašas **10**
Įvadas **11**
1. Kompiuterinės sistemos vientisumo užtikrinimo metodų analizė **12**
1.1. Kompiuterinės sistemos vientisumo problema 12
1.2. Teoriniai metodai vientisumui užtikrinti 14
1.2.1. Kontrolinės sumos 14
1.2.2. Duomenų vientisumo užtikrinimas naudojant skaitmeninį parašą 14
1.2.3. Duomenų vientisumo patikrinimas naudojant laiko žymas 14
1.2.4. Duomenų vientisumas užtikrinimo naudojant *Blockchain* technologijas 15
1.3. Kompiuterinės sistemos vientisumo užtikrinimo produktai 15
1.3.1. Tripwire 15
1.3.2. ICAR (angl. Integrity Checking and Recovery) 16
1.3.3. OSck 20
1.3.4. SecCore 22
1.3.5. HyperCheck 25
1.3.6. Elasticsearch Beats 27
1.4. Analizės išvados 28
2. Kompiuterinės sistemos vientisumo užtikrinimo modelis **30**
2.1. Koncepcinis kompiuterinės sistemos vientisumo užtikrinimo modelis 30
2.2. Išvados 44
3. Kompiuterinės sistemos vientisumo užtikrinimo prototipas ir tyrimas **45**
3.1. Kompiuterinės sistemos vientisumo užtikrinimo prototipas 45
3.2. Tyrime naudojama aparatinė ir programinė įranga 50
3.3. Kompiuterinės sistemos vientisumo užtikrinimo sistemos kokybinis tyrimas 52
3.4. Kompiuterinės sistemos vientisumo užtikrinimo sistemos tikrinimo trukmės ir apkrovos tyrimas 52
3.5. Kompiuterinės sistemos vientisumo užtikrinimo sistemos lygiagrečių tikrinimų tyrimas 56
3.6. Kompiuterinės sistemos vientisumo užtikrinimo sistemos palyginimas su esamomis sistemomis 61
3.7. Išvados 61
4. Išvados **63**
Literatūra **64**

Lentelių sąrašas

2.1 lentelė Panaudos atvejai 33
3.1 lentelė Tyrime naudota aparatinė įranga 51
3.2 lentelė Tyrimui naudojamos politikos 52
3.3 lentelė Stebimos sistemos procesoriaus apkrova kai naudojama politika „Tik aparatinė įranga“ 54
3.4 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Tik aparatinė įranga“ 54
3.5 lentelė Stebimos sistemos kietų tinklo apkrova, kai naudojama politika „Tik aparatinė įranga“54
3.6 lentelė Stebimos sistemos procesoriaus apkrova, kai naudojama politika „Programos ir konfigūracija“ 55
3.7 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Programos ir konfigūracija“ 55
3.8 lentelė Stebimos sistemos kietų tinklo apkrova, kai naudojama politika „Programos ir konfigūracija“ 55
3.9 lentelė Stebimos sistemos procesoriaus apkrova, kai naudojama politika „Visa sistema“ 55
3.10 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Visa sistema“ 55
3.11 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Visa sistema“ 55
3.12 lentelė Funkcionalo palyginimas 61

Paveikslų sąrašas

1.1 pav. Kompiuterinė sistema ... 12
1.2 pav. Aukšto lygio operacinė modelio diagrama ... 15
1.3 pav. ICAR sistemos sluoksniai ... 17
1.4 pav. Failų sistemos apsaugos mechanizmas ... 18
1.5 pav. Pagalbiniai algoritmai ... 19
1.6 pav. Žingsniai apsaugos duomenų bazės atnaujinimo procesui ... 20
1.7 pav. *OSck* architektūrinis modelis ... 21
1.8 pav. SecCore komponentų diagrama ... 23
1.9 pav. Hierarchinio tikrinimo schema ... 24
1.10 pav. *Hypercheck* architektūrinė schema ... 26
1.11 pav. Elasticsearch *Beats* architektūra ... 27
2.1 pav. Aukšto lygio kompiuterinės sistemos vientisumo užtikrinimo modelio schema ... 30
2.2 pav. Koncepcinis kompiuterinės sistemos vientisumo užtikrinimo modelis. ... 31
2.3 pav. Siūloma sistemos architektūra ... 31
2.4 pav. Vientisumo užtikrinimo sistemos panaudos atvejų diagrama ... 33
2.5 pav. Panaudos atvejo „Sukurti talpyklas“ veiklos diagrama ... 36
2.6 pav. Panaudos atvejo „Sudiegti reikiama programinę įrangą stebimame serveryje“ veiklos diagrama ... 37
2.7 pav. Panaudos atvejo „Vientisumo duomenų tikrinimas“ veiklos diagrama ... 38
2.8 pav. Panaudos atvejo „Paleisti stebėtoja“ veiklos diagrama ... 39
2.9 pav. Panaudos atvejo „Aparatinės įrangos duomenų surinkimas“ veiklos diagrama ... 40
2.10 pav. Panaudos atvejo „Vientisumo duomenų atnaujinimas“ veiklos diagrama ... 41
2.11 pav. Vientisumo duomenų tikrinimo ir atstatymo sekų diagrama ... 43
2.12 pav. Sistemos diegimo diagrama ... 44
3.1 pav. Prototipo schema ... 46
3.2 pav. Stebimos Sistemos paruošimas ... 47
3.3 pav. Ataskaitos iškarpa ... 48
3.4 pav. Sugadintas stebimas failas ... 49
3.5 pav. Atliekamas tikrinimas su automatiniu atstatymu ... 49
3.6 pav. Atstatytas failas ... 50
3.7 pav. Pirmojo eksperimento schema ... 52
3.8 pav. Vientisumo tikrinimo trukmė nuo tikrinimo pradžios iki pranešimo pristatymo administratoriams ... 53
3.9 pav. Aparatinės įrangos ir vientisumo duomenų surinkimo trukmė ... 54
3.10 pav. Antrojo eksperimento schema ... 56
3.11 pav. Vidutinės tikrinimo įvykdymo trukmė esant skirtingoms politikoms ... 57
3.12 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Tik aparatinės įrangos“ politiką ... 57
3.13 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Tik aparatinės įrangos“ ... 58
3.14 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Programos ir konfigūracija“ politiką ... 58
3.15 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Programos ir konfigūracija“ ... 59

3.16 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Visa sistema“ politiką 60
3.17 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Visa sistema“60

Santrumpų ir terminų sąrašas

XOR	Suma moduliu du (*angl. Exclusive OR*)
HMAC	Maišos funkcijos pagrindu žinutės autentifikacijos kodas (*angl. Hash-based Message Authentication Code*)
DBVS	Duomenų bazių valdymo sistema
SHA	Saugus maišos algoritmas (*angl. Secure Hash Algorithm*)
PCI	Periferinė komponentų sąsaja (*angl. Peripheral Component Interconnect*)
PKI	Viešojo rakto infrastruktūra (*angl. Public key infrastructure*)
TPM	Patikimos platformos modulis (*angl. Trusted Platform Module*)
I/O	Įvestis ir išvestis (*angl. Input and Output*)
DMA	Tiesioginė atminties prieiga (*angl. Direct Memory Access*)
CPU	Centrinis procesorius (*angl. Central Processing Unit*)
SMM	Sistemos valdymo režimas (*angl. System Management Mode*)
BIOS	Bazinė įvesties/išvesties sistema (*angl. Basic Input/output System*)
DOS	Paslaugos sutrikdymas (*angl. Denial of Service*)
RAM	Operatyvioji atmintis (*angl. Random-access Memory*)
SMRAM	Sistemos valdymo RAM (*angl. System Management RAM*)
SMI	Sistemos valdymo pertrauktis (*angl. System Management Interrupts*)
TCB	Patikima kompiuterinė bazė (*angl. Trusted Computing Base*)
ICMP	Interneto kontrolės žinučių protokolas (*angl. Internet Control Message Protocol*)
TCP	Perdavimo kontrolės protokolas (*angl. Transmission control protocol*)
APIC	Programuojamas pertraukčių valdiklis (*angl. Advanced Programmable Interrupt Controller*)
HTTP	Tai užklausimo - atsakymo protokolas, jungiantis klientą ir serverį (*angl. Hypertext Transfer Protocol*)
SSL	Saugiųjų jungčių lygmuo (*angl. Secure Sockets Layer*)
TLS	Transporto sluoksnio saugumas (*angl. Transport Layer Security*)
NIC	Tinklo sąsajos valdiklis (*angl. Network interface Controller*)
CLI	Komandinės eilutės sąsaja (*angl. Command Line Interface*)

Įvadas

Šiais laikais vis daugiau informacijos yra talpina skaitmeninėje erdvėje [1] ir yra svarbu, kad patalpinta informacija nebūtų suklastotą ar sugadintą. Kompiuterines sistemos vientisumas garantuoja, kad kompiuterinės sistemos posisteme atliks užduotis taip, kaip buvo suprojektavęs jos kūrėjas ir, kad pradiniai duomenys, saugojimo ar perdavimo metu, nebus neleistinai pakeisti. Vientisumo problema yra formuluojama iš prieigos valdymo politikos ir iš mechanizmu, kurie duoda posisteme su izoliacija reikalinga apsaugą nuo informacijos sugadinimo. [2]

Kompiuterinės sistemos vientisumo pažeidimus gali sukelti aparatinės ir programinės klaidos, asmenų kenkėjiški tikslai bei vartotojų klaidos. [3]

Sistemos vientisumas gali būti pažeistas asmenų su kenkėjišku tikslu, pavyzdžiui bandant sunaikinti duomenis ar pakeisti juos klaidinga informacija. Virusai gali pakeisti sisteminius failus ar programas, taip įsitvirtinantys sistemoje, kurioje gali atlikti kenkėjiškus veiksmus. Taip pat aparatinės įrangos vientisumas gali būti pažeistas, pavyzdžiui, kai kurie aparatinės įrangos komponentai gali būti pakeisti kitais, prastesniais arba diskai gali būti išimti įdėti kiti nei, kad nurodyti specifikacijoje. Taip pat vientisumas gali būti pažeistas kai yra neleistinai įdiegiama papildoma įranga [4] [5]. Neleistinai įdiegta įranga gali prieiti prie sistemos resursų ir iš jos ištraukti jautria informacija kaip slaptažodžius, privačiuosius raktus, asmenį identifikuojančius duomenis. Dažniausiai tokie įrenginiai gali dirbti atskirai nuo pagrindinės sistemos ir apsaugos priemones, kaip antivirusinės negali aptikti kenkėjiško kodo, kuris yra patalpintas ir vykdomas kitame procesoriuje. Ypač dabar kai debesų paslaugos kyla aukštyn, dažniausiai neturime prieigos prie aparatines įrangos arba net nežinome, kur yra mūsų tarnybinė stotis.

Taip pat sistemos vartotojai esantys programos lygyje, gali pažeisti sistemos vientisumą, netyčia ištrinant sistemini failą, ar duomenų failą kaip DBVS duomenų bazės failą, tai privestu prie tam tikros sistemos neveikimo.

Magistrinio darbo tikslas yra sukurti modelį su kuriuo būtų galima užtikrinti aparatinės įrangos bei programinės įrangos vientisumą.

Darbo tikslui pasiekti yra iškelti šie uždaviniai:

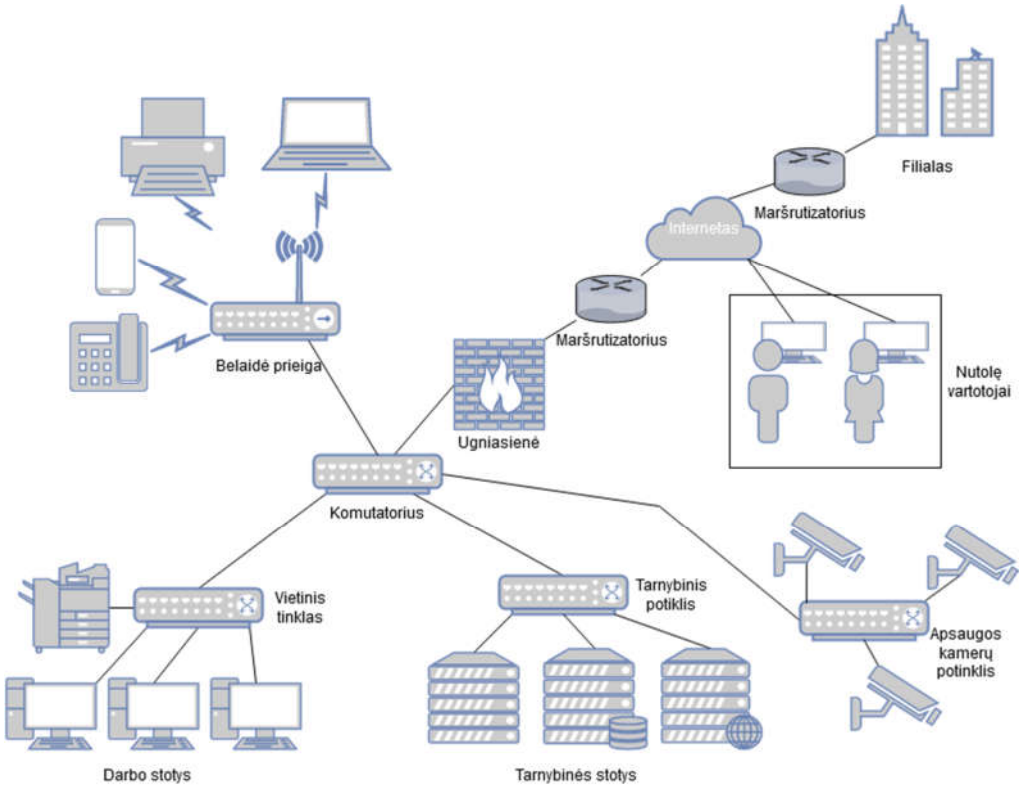
- Išanalizuoti kompiuterinės sistemos vientisumo užtikrinimo metodus ir priemones;
- Sukurti kompiuterinės sistemos vientisumo užtikrinimo modelį;
- Remiantis sudarytu modeliu sukurti sistemos prototipą;
- Naudojantis prototipu atlikti sistemos greitaveikos tyrimą;
- Palyginti sukurtą sistemą su dabar egzistuojančiais produktais.

1. Kompiuterinės sistemos vientisumo užtikrinimo metodų analizė

1.1. Kompiuterinės sistemos vientisumo problema

Kompiuterinė sistema tai yra komponentų, veikiančių kartu siekiant vieno ar daugiau bendrų tikslų, rinkinys 1.1 pav. Kompiuterizuota informacijos sistema gali būti laikoma šešių pagrindinių komponentų sistema:

1. Kompiuterio aparatinė įranga;
2. Kompiuterių programinė įranga, tai yra programos, kurios nurodo kompiuteriui kaip apdoroti duomenis ir dokumentus;
3. Duomenys, kuriuos turi apdoroti aparatinė ir programinė įranga;
4. Kompiuterių personalas, kuris ruošia duomenis kompiuteriniam įvedimui, rašo kompiuterio programas, stebi kompiuterio operacijas ir platina išvestį;
5. Procedūros, reglamentuojančios daiktų veiklą ir tvarkymą;
6. Galutiniai vartotojai, paprastai netechniniai asmenys, kurie naudoja kompiuterio išteklius vykdydami savo kasdienes užduotis.



1.1 pav. Kompiuterinė sistema

Pagal saugumo tyrėjus kompiuterinės sistemos saugumas susideda iš trijų dalių [6]:

- Konfidencialumo;
- Vientisumo;
- Prieinamumo.

Konfidencialumas tai skirtas užtikrinti kad prie duomenų galės prieiti tik tie asmenys, kurie turi tam skirtus leidimus.

Prieinamumas skirtas užtikrinti, kad informacija bus pasiekiama tiems asmenims, kurie turi autorizacija.

Vientisumas turi užtikrinti, kad duomenis būtų galima keisti tik autorizuotais metodais. Neleistini metodai tai yra:

- Neautorizuoto vartotojo duomenų pakeitimas, pavyzdžiui, kai programišius įsilaužia į sistemą ir pakeičia įrašus duomenų bazėje;
- Kai autorizuotas asmuo atlieka neautorizuotus duomenų pakeitimus;
- Kai duomenys yra pakeičiami netinkamais metodais, pavyzdžiui, elektros šuoliai, kurie sugadina duomenis duomenų bazėje;
- Vartotojų neatidumas.

Aparatinės įrangos vientisumas gali būti pažeistas, pavyzdžiui, kai kurie aparatinės įrangos komponentai gali būti pakeisti kitais, prastesniais arba diskai gali būti išimti įdėti kiti nei, kad nurodyti specifikacijoje kurie gali iškraipyti juose kaupiamus duomenis. Taip pat vientisumas gali būti pažeistas kai yra neleistinai įdiegiama papildoma įranga [4] [5]. Neleistinai įdiegta įranga gali prieiti prie sistemos resursų ir iš jos ištraukti jautria informacija kaip slaptažodžius, privačiuosius raktus, asmenį identifikuojančius duomenis. Dažniausiai tokie įrenginiai gali dirbti atskirai nuo pagrindinės sistemos ir apsaugos priemones, kaip antivirusinės negali aptikti kenkėjiško kodo, kuris yra patalpintas ir vykdomas kitame procesoriuje. Ypač dabar kai debesų paslaugos kyla aukštyn, dažniausiai neturime prieigos prie aparatines įrangos arba net nežinome, kur yra mūsų tarnybinė stotis.

Virusai gali pakeisti sisteminius failus ar programas, taip įsitvirtinantys sistemoje, kurioje gali atlikti kenkėjiškus veiksmus ir pažeisti duomenų konfidencialumą ir prieinamumą.

Taip pat sistemos vartotojai esantys programos lygyje, gali pažeisti sistemos vientisumą, netyčia ištrinant sistemini failą, ar duomenų failą kaip DBVS duomenų bazės failą, tai privestu prie tam tikros posistemės neveikimo.

Dėl šių priežasčių kompiuterinėse sistemose reikia užtikrinti jos vientisumą, kad būtų galima pasitikėti duomenimis bei įranga esančioje sistemoje. Jei duomenys yra pakeisti ar sugadinti naudojant neautorizuotus metodus šie duomenys gali prarasti savo vertę bei gali stipriai pakenkti įmonės darbui.

Užtikrinti kompiuterinės sistemos vientisumą tai nėra paprasta užduotis, reikia nustatyti, koks yra dabartinis duomenų bei įrangos vientisumas. Surinktus duomenis saugiai talpinti, kad surinktų duomenų vientisumas nebūtų pažeistas. Nuolatos stebėti ir audituoti sistemas dėl neautorizuoti duomenų pakeitimų ir turėti atsarginė duomenų kopiją bei atsargines aparatinės įrangos dalis, kurios leistų atkurti sistemos vientisumą.

Yra reikalingi metodai, kurie leistu patikrinti sistemos vientisumą, leistų saugiai saugoti surinktus duomenis, informuoti atitinkamus asmenys įvykus sistemos vientisumo pažeidimui bei atstatyti sistemos vientisumą į originalia būseną, prieš vientisumo pažeidimo įvykį.

1.2. Teoriniai metodai vientisumui užtikrinti

1.2.1. Kontrolinės sumos

Kontrolinių sumų skaičiavimas yra gerai žinomas metodas norint atliekant vientisumo tikrinimą. Kontrolinės sumos gali būti suskaičiuojamos duomenų, kuriuos yra diske ir laikomos nuolat. Duomenų vientisumas gali būti patikrinamas palyginant turimą sumą su naujai suskaičiuota reikšme, kai yra skaitoma informacija. Kontrolinės sumos dažniausiai yra suskaičiuojamos naudojant kriptografines maišos funkcijas. Kriptografinės maišos funkcijos suveda skirtingu dydžių eilutes į trumpesnę fiksuoto ilgio rezultatą. Šios funkcijos yra padarytos taip, kad butų atsparos kolizijai, tai reiškia, kad surasti dvi skirtingas eilutes, kurios duotu tokį pat rezultatą yra labai sunku. Funkcijos tokios kaip MD5 ir SHA1 turi tokias ypatybes kaip atsitiktinumas. MAC yra tokia maišos funkcija, kurio sugeneruotas rezultatas yra kriptografiškai apsaugotas. Ji veikia naudojant pagrindinę maišos funkciją su kuria suskaičiuoja pranešimo reikšmę ir raktą, tokiu būdu nustatoma neleistina kontrolinės sumos verčių keitimas. Tai yra vienas iš pagrindinių būdų užtikrinančių, kad duomenys nėra pažeisti keliaujant per nepatikimus kanalus, kaip per internetą. Taip pat gali būti naudojant ir duomenų talpyklų kontekste, kad būtų užtikrinamas vientisumas atliekant duomenų nuskaitymus. [3]

1.2.2. Duomenų vientisumo užtikrinimas naudojant skaitmeninį parašą

Skaitmeninis parašas yra viena iš kriptografijos idėjų. Pagrindinė skaitmeninio parašo nauda yra patikrinti siunčiamo dokumento autentiškumą. Skaitmeninio parašo principas yra toks, kad siunčiamą dokumentą turi pasirašyti siuntėjas, o gavėjas gali patikrinti parašą, kad patikrintų pateikto dokumento autentiškumą. Jos funkcija yra patvirtinti atsiųstus duomenis. Skaitmeninio parašo metodas naudoja maišos algoritmą, kad gautų unikalų simbolių derinį, vadinamą pranešimos santrauką. Tokiu būdu siuntėjas yra atsakingas už dokumento turinį ir gavėjas gali patikrinti dokumentų tikrumą. [7]

Unikalumas yra toks, kad jei kelionės viduryje duomenis yra modifikuojami ir naikinami, net jei ir tik vienas simbolis, tada gavėjo pranešimos santrauka skiriasi nuo tų, kurie buvo siunčiami iš pradžių. Kitas unikalumas yra toks, kad pranešimo santrauka negali būti grąžinta į pradinę formą, kokia ji buvo prieš parašą, todėl ji vadinama vienpusė maiša. [8]

1.2.3. Duomenų vientisumo patikrinimas naudojant laiko žymas

Laiko žymos mechanizmas tai yra procesas skirtas saugiai sekti dokumento sukūrimo ir keitimo laiką naudojant viešojo rakto infrastruktūros skaitmeninį parašą ir teisingą laiką. Laiko žymos žetoną išduoda patikima trečioji šalis, kuri veikia kaip laiko žymėjimo institucija. Žetonas yra naudojamas kaip įrodymas apie tam tikrų duomenų buvimą prieš svarbius įvykius, tokius kaip sutartys, finansinės operacijos, teisiniai įrodymai ar medicininiai dokumentai, be galimybės savininkui suteikti atgalines laiko žymes. Kuriant failą yra suskaičiuojama failo maišos funkcijos reikšmė ir naudojant ją yra gaunamas laiko žymės žetonas iš laiko žymėjimo institucijos. Meta duomenys kaip laikas, failo kelias, laiko žymės žetonas yra patalpinami į duomenų bazę. Failo vientisumo patikrinimui yra išsiunčiamas failo kelias ir laiko žymės žetonas patikrinimui kur yra suskaičiuojama nauja maišos funkcijos reikšmė ir palyginama su ta kuri yra laiko žymės žetone. Jei reikšmės sutampa failo vientisumas nepažeistas, jei nesutampa, failo vientisumas pažeistas. [9]

1.2.4. **Duomenų vientisumas užtikrinimo naudojant *Blockchain* technologijas**

Blockchain tai yra laiko sužymėta blogų grandinė, kuria prižiūri kiekvienas tinkle dalyvaujantis mazgas. Blokas tai yra konteineris kuris kaupia transakcijas. Blokai yra kriptografiškai sujungti, kiekvienas blokas yra pasirašytas skaitmeniniu parašu ir sujungtas su ankstesniu bloku, nes jame yra įtrauktas ankstesnio bloko maišos vertė. Nauji blokai gali būti pridėti tik prie grandinės pabaigos, taigi *Blockchain* suteikia nekintamą duomenų saugykla, esamų transakcijų negalima atnaujinti ar ištrinti. Įvykusiomis transakcijomis galima pasitikėti, nepasitikėdami jokiomis trečios šalies institucijomis. Kriptografija ir skaitmeniniai parašai yra naudojami tapatumui ir autentiškumui įrodyti bei prieigos prie *blockchain* skaitymo ir rašymo kontrolei gauti. [10]

Duomenų vientisumui užtikrinti galima sukurti transakcijas kuriose yra:

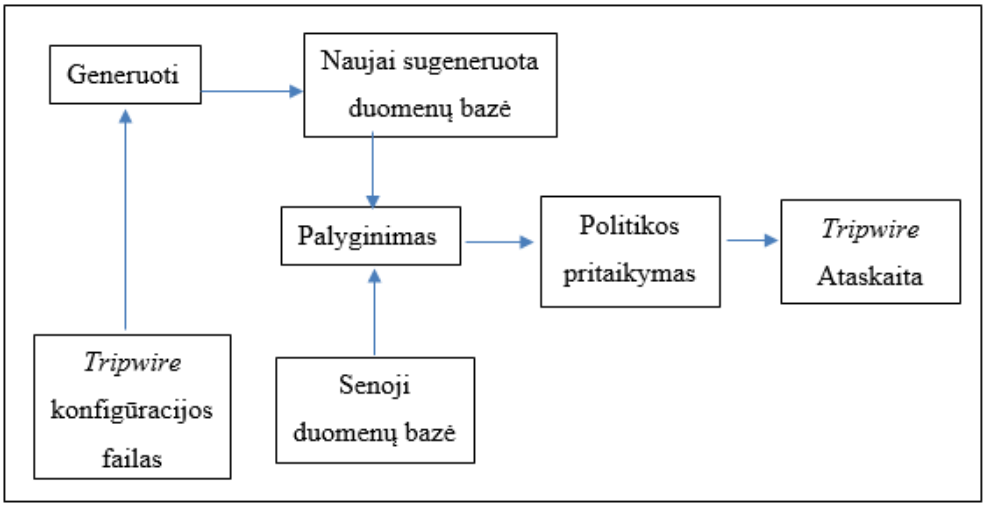
- Transakcijos tipas;
- Dabartinė failo maišos reikšmė;
- Prieš tai buvusios failo maišos reikšmė;
- Laiko žymė;
- Kelias iki failo;
- Skaitmeninis parašas.

Kai norima patikrinti vientisumą blokų grandinėja galima susirasti bloką, kuris turi naujausia failo maišos reikšmė ir palyginti ją su naujai paskaičiuotą. [11]

1.3. Kompiuterinės sistemos vientisumo užtikrinimo produktai

1.3.1. **Tripwire**

Failų sistemos vientisumui užtikrinimui galima naudoti įrankį *Tripwire* [12] [13]. Šis įrankis patikrina dabartinę failų sistemą su žinoma bazinę būseną ir perspėja, ar buvo padaryti, kokie pokyčiai failų sistemoje. Bazinė būsena bei tikrinimo elgsena yra valdoma politikos failo, kuriame yra aprašoma, kokie failai ir katalogai yra stebimai ir kokius parametrus stebėti, pavyzdžiui failo maiša, failo teises ar savininką.



1.2 pav. Aukšto lygio operacinė modelio diagrama

Yra sugeneruojami raktai su kuriais yra pasirašomi politikos, konfigūracijos, ataskaitos ir duomenų bazės failai, kad būtų žinomas autentiškumas. Konfigūracijos faile yra aprašoma, kur bus siunčiamos atskaitos, kur yra duomenų bazės failai, kur yra sugeneruoti rakai bei, kur yra politikos failas. Politikos faile yra aprašoma, kokius failus ir katalogus bei, kokius atributus reikia stebėti:

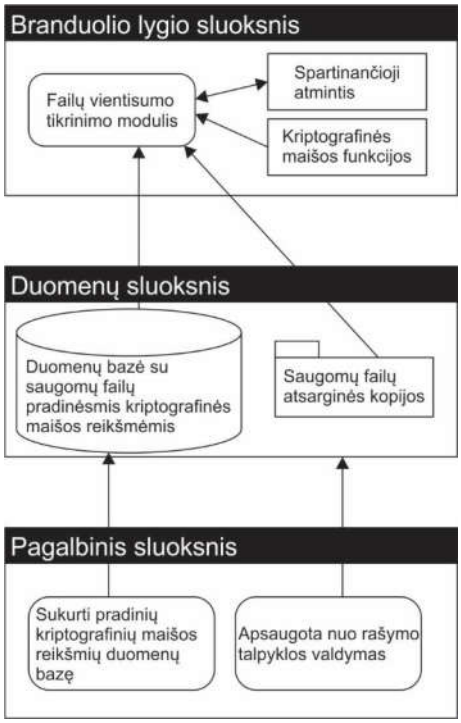
- Failo maiša;
- Failo, katalogo teises;
- Failo, katalogo atidarymo laikas;
- Failo tipas;
- Failo dydis;
- Failo, katalogo savininkas;
- Failo modifikacijos laikas.

Duomenų bazėje yra talpinama dabartinė stebimų failų būsena ir pagal 1.2 pav. Aukšto lygio operacinė modelio diagrama yra atliekamas tikrinimas. Sugeneruotus raktus, politikos, konfigūracijos bei duomenų bazes failus rekomenduojama perkelti į tik skaitymui laikmena, kad šie nebūtų neautorizuotai modifikuoti, nes pavyzdžiui duomenų bazės failas yra nešifruotas ir pasauliui skaitomas. Jei aptinkamas vientisumo pažeidimas yra generuojama ataskaita su pažeidimai ir ši atskaita yra siunčiama sistemos administratoriui patikrinimui.

1.3.2. **ICAR (angl. Integrity Checking and Recovery)**

Operacinės sistemos bei failų sistemos vientisumui užtikrinti ir jei reikia atstatymui galima naudoti įrašymams apsaugota duomenų talpykla. [14] [15] Sistema pavadinta *ICAR* (*angl. integrity checking and recovery*). Ši sistema susideda iš 3 sluoksniu 1.3 pav.: branduolio lygio sluoksnis, duomenų sluoksnis, pagalbinių priemonių sluoksnis. Branduolio lygio sluoksnis patikrina failų sistemos vientisumą. Duomenų sluoksnis turi saugomų failų atsargines kopijas bei duomenų bazę, kurioje yra failų kriptografinės maišos reikšmės. Pagalbinių priemonių sluoksnis suteikia priemones vartotojams ir administratoriams lengvai valgyti sistemą.

7. **Branduolio lygio sluoksnis:** kai *ICAR* sistema yra įkraunama į *Linux* branduolį kaip branduolio plėtinys, ji tampa branduolio dalimi. Užsikrovimo metu plėtinys yra įkraunamas į kompiuterio darbine atmintį. Jis suskaičiuoja failų kriptografines maišos funkcijos reikšmes, kurios bus patalpintos duomenų bazėje duomenų sluoksnyje. Naudojamas algoritmas yra Saugus Maišos algoritmas (*angl. Secure Hash Algorithm*) vienas iš jų yra SHA-3 arba SHA-256. Šis sluoksnis taip pat turi spartinsiančiąja atmintį (*angl. cache*), kurioje yra laikomi vientisumo tikrinimų rezultatai, kurie jau ankščiau buvo atlikti, tai leidžia padidinti skaičiavimų greitį.
8. **Duomenų sluoksnis:** šiame sluoksnyje yra laikomi suskaičiuoti failų kriptografinės maišos funkcijos reikšmės, kurios buvo suskaičiuotos branduolio lygio sluoksnyje. Taip pat šiame lygyje yra laikomos atsarginės kopijos apsaugotų failų, kurios yra naudojamos atstatymui, kai yra aptinkamas failų integralumo pažeidimas. Tiek duomenų bazė tiek atsarginės kopijos yra laikomos rašymams apsaugotuose arba tik skaitymui duomenų talpyklose. Papildomas saugos mechanizmas naudojamas *ICAR* sistemoje yra apsaugoti montavimo taškai kurie rodo į duomenų talpykla, kad nebūtų pateikta ar suklastota duomenų bazė.
9. **Pagalbinių priemonių sluoksnis:** šis sluoksnis leidžia vartotojui bendrauti su *ICAR* sistema. Tai yra tokiems vartotojams kaip sistemų administratoriams, kurie gauna pranešimus apie įsibrovimus ir kada atstatymas yra baigtas.



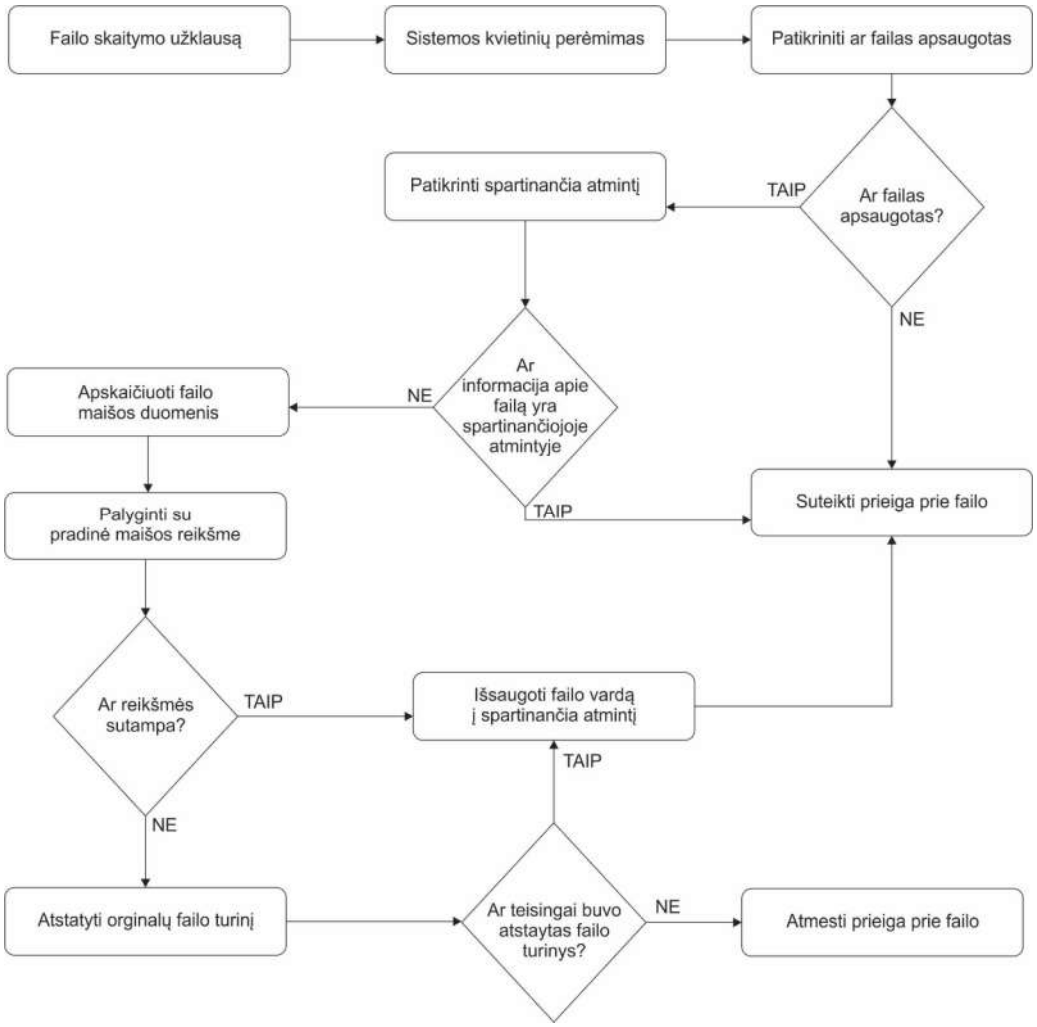
1.3 pav. ICAR sistemos sluoksniai

Pagrindinis algoritmas 1.4 pav. vientisumui tikrinti naudojantis *ICAR* yra atliekamas failo skaitymo arba vykdymo bandymo metu. Tikrinimas yra atliekamas kiekvieną kartą, kai failas yra skaitomas. Pirmiausia yra nuskaitoma duomenų bazė, kad patikrinti ar failas, kurį bandoma atidaryti, yra prie vienas iš apsaugotų failų. Jei nėra informacijos apie šį failą duomenų bazėje, tai, manome, kad šio failo vientisumas nėra saugomas ir prieiga yra suteikiama failo reikalaujančiam procesui. Jei duomenų bazėje yra informacija apie failą tai reiškia, kad failo vientisumas yra saugomas ir *ICAR* patikrina ar failo turinys buvo keistas nuo sistemos paleidimo.

Patikrinimas reikalauja suskaičiuoti kriptografinės maišos funkcijos reikšme dabartiniam failui ir palyginti ji su pradine reikšme, kuri yra patalpinta duomenų bazėje. Tačiau nepaisant to, sistemoje yra naudoja pakankamai efektyvi branduolyje įmontuoja kriptografines maišos funkcijos, šie skaičiavimai užima daug laiko. Atsižvelgiant į našumo problemas, *ICAR* įgyvendino spartinančiąja atmintį, kuri laiko duomenis apie teigiamai patikrintus failus. Jeigu failas buvo prieš tai patikrintas, nėra tikslo iš naujo perskaičiuoti maišos funkcijos reikšmę jei nebuvo modifikuotas failas. To pasėkoje, kai procesas prašo prieigos prie failo, kuris jau buvo patikrintas, iškart yra suteikiama prieiga prie failo. Kitu atveju jei spartinančioje atmintyje nėra įrašo apie failą, privaloma suskaičiuoti failo kriptografinę maišos funkcijos reikšmę ir palyginti su pradine reikšme duomenų bazėje. Jeigu patikrinimas sėkmingas saugos mechanizmas patalpina informacija apie failą į spartinančiąja atmintį ir suteikia prieiga failo reikalaujančiam procesui.

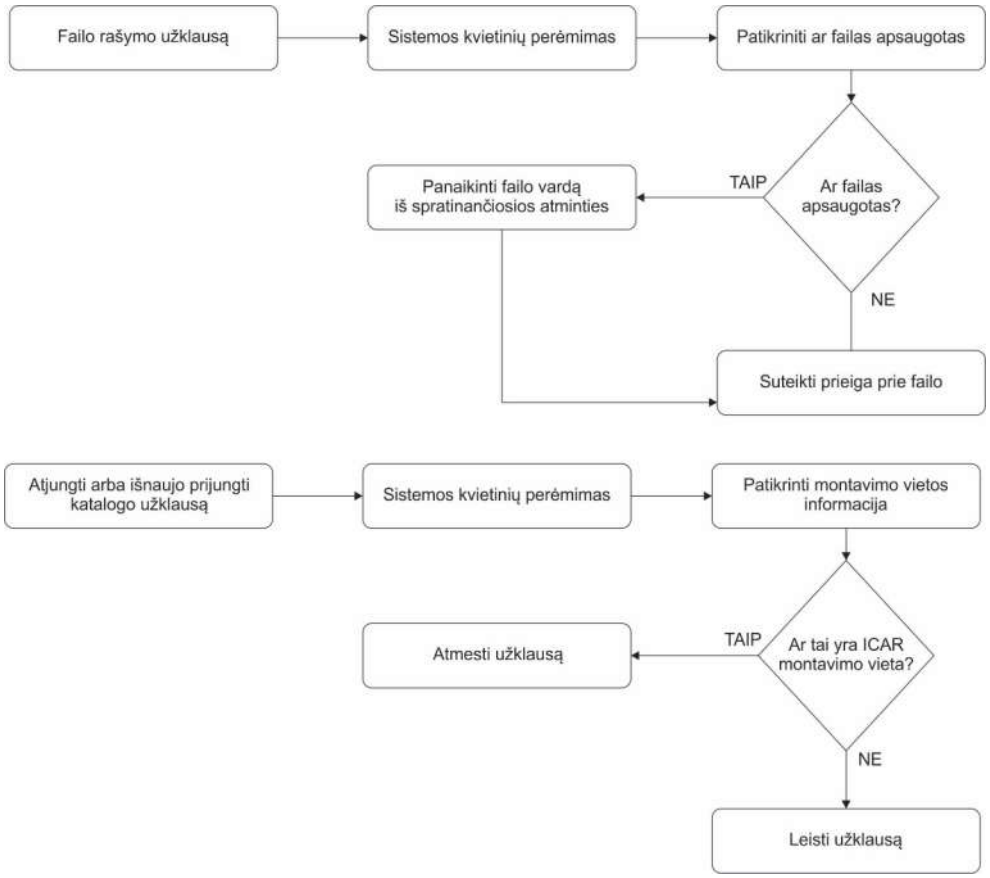
Tačiau jei suskaičiuota maišos funkcijos reikšmė skiriasi nuo pradinės reikšmės duomenų bazėje yra įjungiamas apsaugos mechanizmas, kuris bandys atstatyti originalų failą ir atsarginės kopijos ir perspės sistemų administratorių dėl pažeidimo. Pirmame žingsnyje mechanizmas bandys atstatyti originalų failą kopijuodamas ji iš tik skaitymui esančio talpyklos atsargines kopijos. Jei atstatymas

pavyksta pažeistas failas pakeičiamas originaliu. Tada algoritmas seka žingsnius teigiamam failo patikrinimui ir suteikiama prieiga prie failo. Tačiau jei failo atstatymas nepavyksta, prieiga prie failo yra neleidžiama. Failas tampa nepasiekiamas skaitymui ar vykdymui vartotojui ir veikiantiems procesams sistemoje.



1.4 pav. Failų sistemos apsaugos mechanizmas

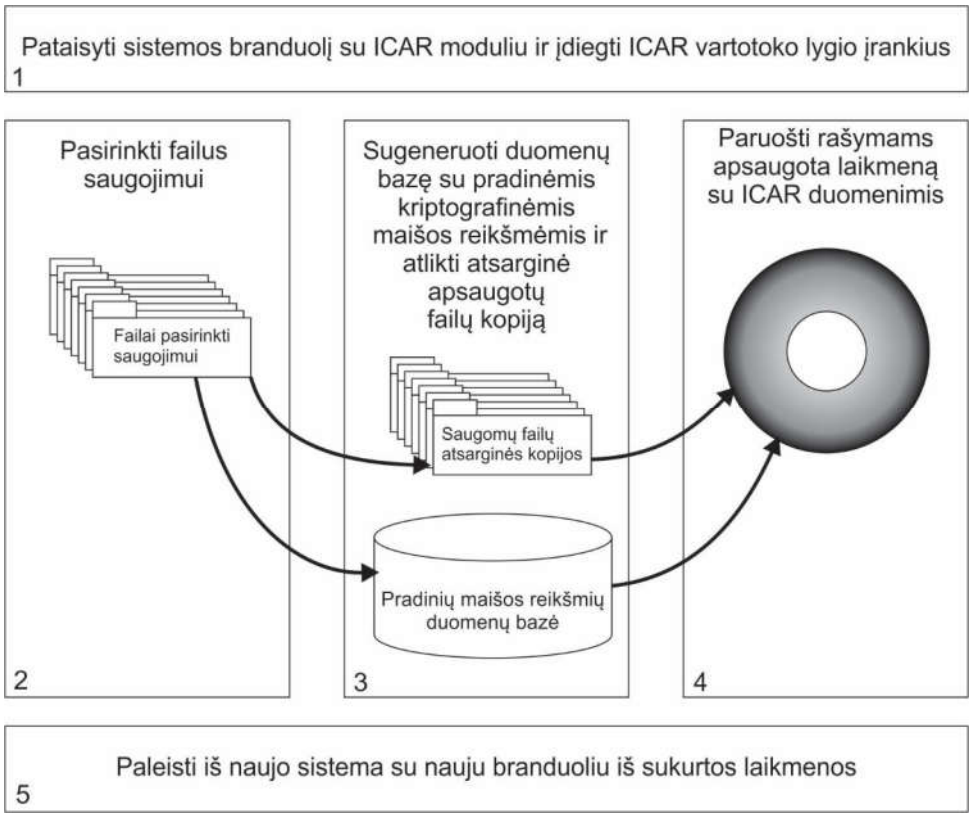
1.5 pav. Pagalbiniai algoritmai yra pavaizduotos du pagalbiniai algoritmai. Pirmas algoritmas atsako už spartinančiosios atminties išvalymą, antrasis algoritmas aprašo, kaip sistema apsisaugoja nuo montavimo taško su *ICAR* informacija numontavimu ar permontavimu.



1.5 pav. Pagalbiniai algoritmai

Pagrindiniai žingsniai įdiegiant *ICAR* 1.6 pav.:

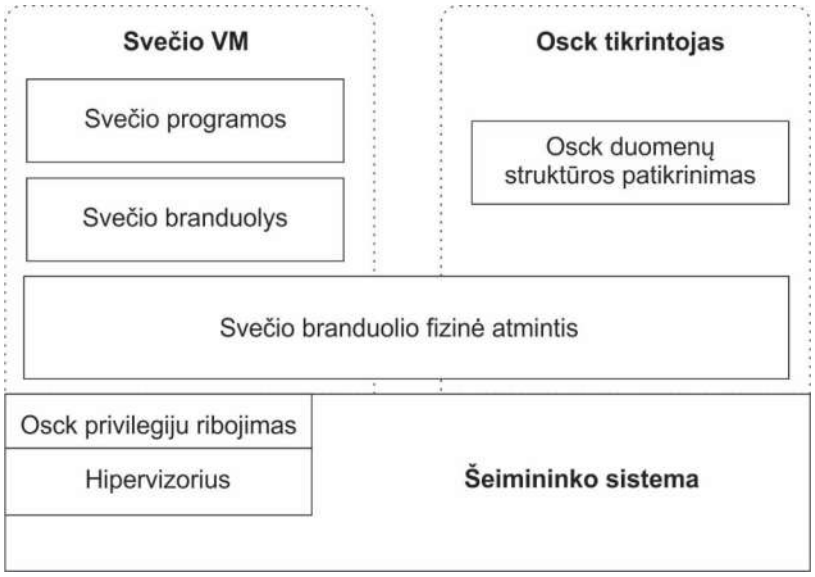
- *ICAR* įdiegimas į operacine sistema. Tai yra, įdiegiamas *ICAR* modulis į sistemos branduolį ir įdiegiamas *ICAR* vartotojo lygio įrankius.
- Pasirinkti apsaugotus failus;
- Sugeneruoti pradines failų kriptografines maišos funkcijos reikšmes;
- Paruošti rašymams apsaugota talpykla kurioje yra pradines failų kriptografines maišos funkcijos reikšmes, apsaugotų failų turinys bei operacinės sistemos branduolys su *ICAR* moduliu.
- Perkrauti sistema su nauju branduoliu ir rašymams apsaugota talpykla.



1.6 pav. Žingsniai apsaugos duomenų bazės atnaujinimo procesui

1.3.3. **OSck**

Kitas metodas, kuris leidžia užtikrinti operacinės sistemos branduolio vientisumą yra *OSck* [16]. *OSck* yra virtualios mašinos paremtas sprendimas, kuris aptinka šakninės programas (*angl. rootkits*). *OSck* veikia atskiroje gijoje su tokiomis pačiomis privilegijomis kaip ir svečio operacinė sistema, tačiau yra atskirta naudojat hipervizorių, tai leidžią tikrintojui išlikti nepažeistam, kai branduolys yra kompromituotas 1.7 pav. Tikrintojas turi prieiga prie svečio operacines sistemos fizinės atminties, tai leidžia atlikti vientisumo tikrinimus, kurie patikrina branduolio duomenų struktūras, kurios yra atmintyje ar jos teisingos. *OSck* aptinka šakninės programas neleisdamas modifikacijų statinėms, nuolatinėms ir dinaminės valdymo perdavimams. Statinius ir nuolatinius valdymo perdavimus apsaugo neleidžiant svečio operacines sistemos branduoliui keisti šių reikšmių po to kai jos yra įdiegtos. Dinaminiai valdymo perdavimai yra apsaugoti tikrinant tipo saugos savybes. Taip pat *OSck* suteikia programavimo aplinką, identišką branduolio kodo rašymui, kad būtų galima patikrinti ne kontrolės duomenis. Galiausiai, *OSck* turi galimybę sustabdyti veikiantį branduolį, kurios patikrinimas nepavyksta dėl to svečio operacinė sistema modifikuoja duomenų struktūras tuo pačiu laiku kai *OSck* tikrina.



1.7 pav. *OSck* architektūrinis modelis

OSck aptinka kenkėjišką kodą veikiančiame operacinės sistemos branduolyje, patikrindamas, kad aparatūros ir atminties reikšmės, naudojamos valdymui perduoti, laikosi tam tikrų saugos savybių. Kad šie patikrinimai užtikrintų reikšmingą saugumą, *OSck* turi užtikrinti, kad kodas, kuris iškviečia valdymo perdavimą, ir kodas, kuris yra perdavimo tikslas, yra kodas, kuris yra originalaus, o ne kenkėjiško branduolio įgyvendinimo dalis. Todėl *OSck* turi valdyti atminties regionus, kurie laikomi galiojančiais branduolio kodais. Panaudodami patikimos įkrovos procesą su technologija kaip TPM [17], kad sistema iš potencialiai nepatikimo branduolio atvaizdo diske, į vykdantį patikimą branduolį, kuriam būtų galimą taikyti *OSck* patikrą. Dinamiškai įkraunami branduolio moduliai taip pat turi tapti atminties regiono dalimi, kurią *OSck* laiko galiojančiu kodu. Šiuo metu *OSck* remiasi baltojo sąrašo ir kriptografinėmis maišos reikšmėmis, kurie nusako kokie moduliai yra patikimi. Manoma, kad šis sprendimas yra pagrįstas, kad būtų užtikrintas aukštas saugumas, tačiau patikimesni sprendimai kaip modulio pasirašymas platintojo yra galimi.

Kai branduolys yra įkraunamas į atmintį jis privalo neperduoti valdymo kenkėjiškam kodui kai operacinė sistema yra vykdoma privilegijuotame lygyje. *OSck* patikrina šią savybę užtikrindama, kad bet kokia seka rodyklių nukreipimų, kuriuos atlieka branduolys nukreips į tokia rodyklę, kuri rodys į saugia funkciją. *OSck* laiko kiekvieną mazgą (tai yra branduolio duomenų struktūra atmintyje) atskirai patikrinant ar duomenų struktūroje esančios rodyklės atitinka tipo grafikus kaip nurodo branduolio kodas. *OSck* teigia, kad kiekviena rodyklė turi rodyti į objektą, kurio duomenų tipas yra nurodytas rodyklėje ir visos funkcijos rodyklės turi būti nukreiptos į saugias funkcijas. Svarbiausias uždavinys yra parodyti, kad naudojant tipinę informaciją iš galimai pažeisto branduolio galima patikrinti, ar bet kuris kelias iš branduolio šakninio objekto į funkcijų rodyklę turi paskatinti saugią funkciją. Čia pateikiame argumentą, kad net kenkėjiškas branduolys negali išvengti aptikimo, sugadindamas *OSck* tipo informacija.

Yra numanomi du priskyrimai branduolio atminties į duomenų tipus, tai yra efektyvus ir susietas priskyrimas. Efektyvus priskyrimas tai priskyrimas, kuri netiesiogiai sukuria branduolio kodo veiksmai. Atminties adresui efektyvus priskyrimas naudoja tipą kurį branduolio kodas netiesiogiai prisiima tuo adresu, remdamasi rodyklių sekų seka. Tai yra jei branduolys seka *struct page* rodyklę

ir atvyksta į atminties adresą 0x100, tai 0x100 atminties adresas yra *struct page* tipo. Darome prielaidą, kad efektyvus priskyrimas atitinka branduolio tipo grafiką - branduolį nenurodo rodyklių į nesusijusį tipą, kai seka jais. šakninės programas siekia pakeisti efektyvų priskyrimą, keisdamas branduolio rodykles taip, kad branduolys manytu, kad kenkėjiška funkcija yra teisėta, nes funkcijų rodyklė nurodo į jį.

Susietas priskyrimas - tai žemėlapis, kurį *OSck* sukuria, pradedant nuo branduolio pirminio kodo tipo informacija ir papildydama potencialiai nepatikima informacija apie kupetoje priskirtus tipus. Atminties vietos branduolio atvaizde (pvz., Funkcijos ir statinės duomenų struktūros) yra susietos su tipu nuorodos metu, kai simboliu vardai (su susijusio tipo šaltiniu) priskiriami adresui. Atminties vietų tipas krūvoje yra susietas, kai dinaminės skiriamosios atminties skenavimas vykdomas vykdymo metu analizuojant branduolio atminties duomenų struktūras. Susieti prikyrimas ir efektyvus priskyrimas sutiks dėl visų globalių šaknų. globalinės šaknys yra simboliai, sukompiliuoti į branduolį, ir juos saugo *OSck*, saugodamas branduolio kodą, kuris naudoja šiuos simbolius kaip pastovias vertes. Be to, susietas prikyrimas sujungia visų saugių funkcijų įėjimo taškus į jų tipo parašą. Tiek globalines, tiek saugioms funkcijoms, kompromituotas branduolys negali keisti susieto prikyrimo, nes *OSck* nesiremia dinaminėms branduolio būsenomis tiems adresams.

Esant šakinėms programoms, susietas priskyrimas turi skirtis nuo efektyvaus priskyrimo bent vienoje vietoje: kenkėjiškos funkcija. Branduolys tiki, kad atmintis kenkėjiškos funkcijos yra tinkama funkcija, o *OSck*, kuris susieja saugių funkcijų adresus, kai branduolys yra susietas, ir nesiremia keičiamų funkcijų rodiklių reikšmėmis.

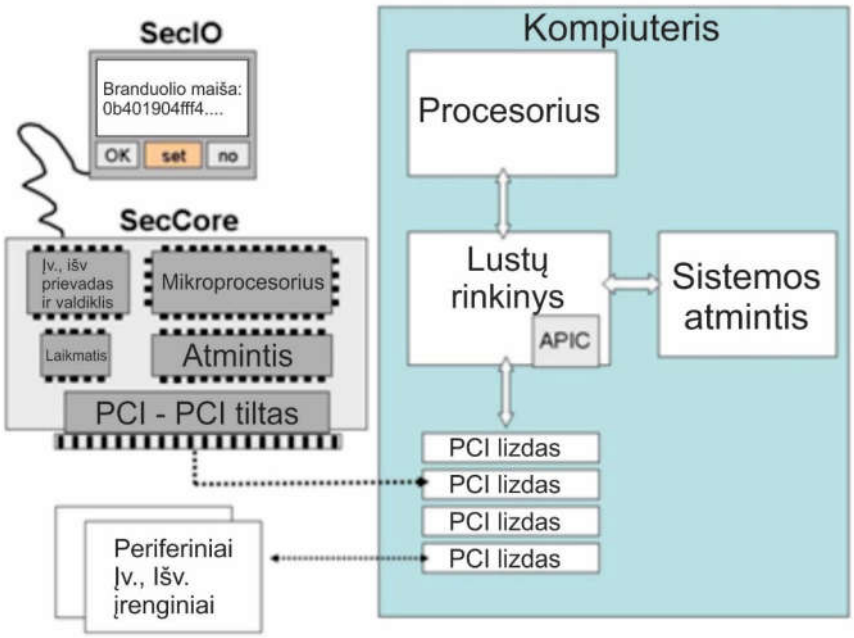
Atakos nuo, kurios neapsaugo *OSck* yra jei puolikas gauna laikina skaitymo ir rašymo prieiga į branduolio atmintį per branduolio klaidą ar pasinaudodamas klaida *setuid* programoje arba prieidamas prie „/dev/kmem“ sąsają. užpuolikas, turintis prieigą prie branduolio atminties, gali sugadinti branduolio būseną ir sukelti branduolio gedimą. *OSck* neapsaugo nuo šių pakeitimų ar panašių DOS atakų. Vietoj to, *OSck* tikslas yra aptikti prasmingus branduolio būsenos pakeitimus: modifikacijas, kurios palengvina tolesnę kenkėjišką veiklą taip, kad sistemos administratorius jų neaptiktų.

1.3.4. **SecCore**

Aparatinis metodas, kuris leidžia užtikrinti kompiuterinės sistemos vientisumą. Per PCI magistralę prijunktas įrenginys, kuris hierarchiniu būdu pirmiausia patikrina operacinės sistemos branduolio vientisumą, o toliau patikrintas branduolys patikrina likusią sistema [18]. Sistema susideda iš *SecCore* kuris yra realizuotas aparatinėje įrangoje bei komunikavimo sistema pavadinta *SecIO*. *SecCore* yra laikomi viešojo rakto infrastruktūros raktai ir programine įranga, kuri leidžia apskaičiuoti kriptografinės maišos funkcijas. Taip pat jis gali pasiekti kompiuterio darbine atminti bei pertraukti procesorių. *SecIO* yra fiziškai prijunktas prie *SecCore* ir jis yra nepasiekiamas iš pagrindinio kompiuterio.

SecCore yra įterptine sistema kurioje veikia operacinė sistema ir programinė įranga ir ji yra kviečiama pagal laikmati. Paveiksle 1.8 pav. yra parodyta kaip yra prijunktas *SecCore* per PCI sąsaja. *SecCore* turi bendras savybes kaip ir kituose kriptografinėse lustuose tokiuose kaip TPM [17]. Tačiau skiriasi nuo kitu kriptografinių lustų tuo kad reikia šių sąlygų

1. Vidiniai SecCore resursai nebūtų pasiekiami pagrindinio kompiuterio procesoriaus arba lusto rinkinio.
2. SecCore turi pasiekti dalį pagrindinio kompiuterio darbinę atmintį.
3. SecCore turi galimybe sustabdyti sistema kai to prireikia.



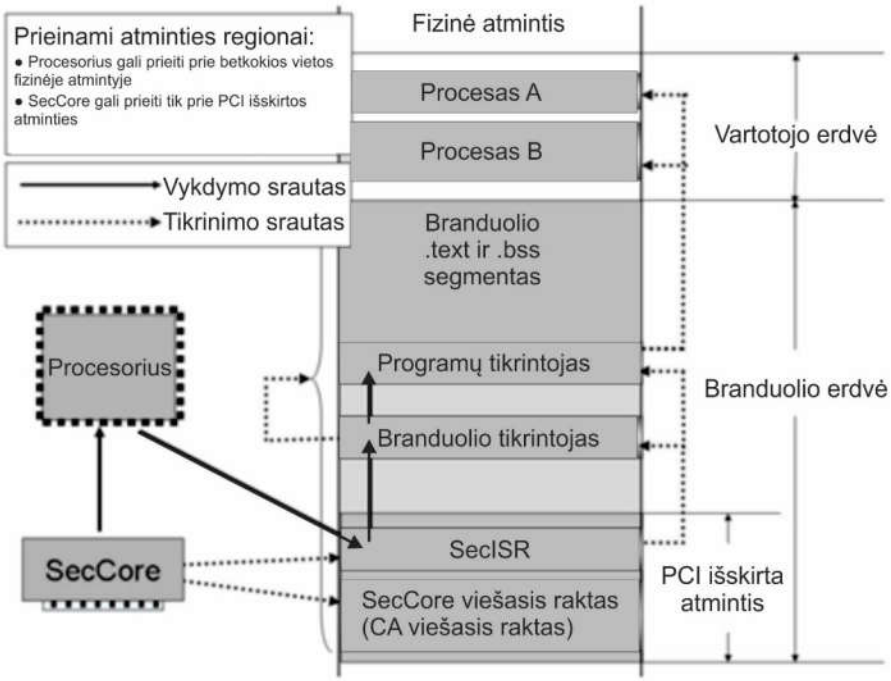
1.8 pav. SecCore komponentų diagrama

Iš pagrindinio kompiuterio operacinės sistemos pusės, kai yra įdėtas ir aptiktas *SecCore*, operacinė sistema suteikia nemaskuojama pertraukties vektorių, sukonfigūruojamas programiškai išskirta atmintis, išprašo įvesties, išvesties regioną, įgalina DMA ir sukuria įrenginiui specialų failą. Komunikacija tarp pagrindinio kompiuterio ir *SecCore* yra dvipusė. Iš pagrindinio kompiuterio į *SecCore* komunikacija vyksta įrašant į *SecCore* komandų registrą komandą ir duomenis, šie duomenys yra perduodami per PCI atmintį. *SecCore* atnaujina savo būsenos registrus kai operacija yra atlikta ir pagrindinis kompiuteris gali nuskaityti. Operacijos dažniausiai atliekamas atidarant *SecCore* įrenginio failą ir atliekant *ioctl* sisteminį kvietimą. Komunikacija iš *SecCore* į pagrindinį kompiuterį vyksta per pertrauktį, taip yra gaunamas pagrindinio kompiuterio procesoriaus dėmesys, toliau komunikacija vyksta kaip ir iš pagrindinio kompiuterio į *SecCore* įrenginį. Taip pat paveiksle 1.8 pav. galime matyti prijunktą *SecIO* prie *SecCore* tai yra mažas įvesties ir išvesties įrenginys kuris gali būtų nebrangus įrenginys, pavyzdžiui, kaip vienspalvis ekranas su skaičiuotuvo stiliaus klaviatūrą arba aukštesnės klases *SecIO* gali turėti ir liečiama ekraną. Klaviatūra gali turėti kelis specialus mygtukus kaip „gerai“, „ne“ ir „nustatyti“. Manykime, kad visos įvestys ir išvestys iš *SecIO* yra saugios tai yra jo nėra matomos ar pažeidžiamos pagrindinio kompiuterio programų.

Branduolio vientisumo tikrinimas metodas yra suskaičiuojamas kriptografinė maišos funkcijos reikšmė paleisties metu, kuri toliau nuolatos yra iš naujo patikrinama, kad įsitikinti jog nebuvo programinės įrangos pakeitimų. *SecCore* periodiškai tikrina branduolio *.text* dalį. Naudojant *SecCore* kad būtų stebimas veikiantis branduolys turi keletą limitų. Jis negali patikrinti veikiančių branduolio

modulių taip pat negali patikrinti vartotojo procesų. Dėl to, kad užtikrinti visos sistemos vientisumą yra pasitelkiama hierarchiniu patikrinimu.

Patikrinimas yra išplečiamas taip, kad visa veikianti programinė įranga yra patikrinamą. Paveiksle 1.9 pav. yra pavaizduotas hierarchinis tikrinimas. Vietoj viso branduolio tikrinimo *SecCore* patikrina tik kritinę branduolio *.text* dalį kuria pavadino *SecISR*. Ši dalis yra pertraukiu aptarnavimo paprogramė, ji bus įvykdoma pagrindinio kompiuterio procesoriuje kai yra gaunama signalas iš *SecCore* nuo šio taško *SecISR* tampa pasitikėjimo šaknis (*angl. root-of-trust*). *SecISR* tai yra pradžios paprogramė, ji patikrina ir paleidžia branduolio tikrintoja ir programų tikrintoja kurios yra kitos branduolio funkcijos. Branduolio tikrintojas ir programų tikrintojas yra kitos pasitikimos programos hierarchijoje. Branduolio tikrintojas patikrina visa branduolio *.text* dalį bei jo modulius, o programų tikrintojas patikrina kitu veikiančių programų vientisumą pavyzdžiui kaip antivirusinės programos.



1.9 pav. Hierarchinio tikrinimo schema

Šis hierarchinis tikrinimas turi keletą pranašumų:

- *SecCore* žino tik apie *SecISR*. *SecISR* yra branduolio *.text* dalyje jos adresas, dydis ir vientisumas galima būtų numatytas iš ankščiau ir laikomas *SecCore*.
- Visi kiti patikrinimas yra programiniai ir jie gali būti vykdomi pagrindinio kompiuterio procesoriuje, hierarchinis patikrinimas garantuoja kad tikrintojas branduolyje negali būti išjunktas ar pakeistas. Be to yra geriau perduoti darbus į pagrindinį procesorių jei tai daryti *SecCore* nes jis yra daug kartų silpnesnis nei pagrindinis procesorius.
- Nors ir *SecCore* gali pasiekti pagrindinio kompiuterio atmintį tai nereiškia, kad gali pasiekti visa atmintį. Dažniausiai PCI pažymėta atmintis yra nedaugiau nei 64 kilobaitai.

Tačiau lieka viena problema. Tikrintojai veikia pagrindiniame kompiuteryje ir vientisumo duomenys yra prieinami kompiuterio procesoriui, vietoj *SecCore*. Šie duomenys tiek failuose tiek atmintyje yra

pažeidžiami. Jei tikrintojai gali pasiekti, tai gali ir virusai. Norint apsaugoti duomenis juos reikia pasirašyti su *SecCore* privačiuoju raktu, paskui viešasis raktas yra perduodamas pagrindiniam kompiuteriui, kad vėliau būtų patikrinamas parašas. Tačiau tai sukelia papildoma problema viešasis raktas gali būti sukeistas ir parašas suklostuotas. Norit išspręsti šią problemą reikia viešąjį raktą įterptį į branduolį ir leisti *SecCore* ji patikrinti. Tikrinimo technika tokia pat kaip ir *SecISR*.

Sprendžiant programinės įrangos atnaujinimo problema yra naudojamas *SecIO* tai leidžia atskirti kur buvo žmogaus paleistas prašymas nuo programos paleisto prašymo. Įtraukdami žmogų į šį procesą galime nustatyti kurie atnaujinimai buvo legaliai paleisti.

Kelios atakos nuo kurių galime apsisaugoti naudojant *SecCore*:

- Ataka prieš programas. Jei virusas užkrečia programą tai virusas ir programa yra aptinkami antivirusinės. Jei virusas užkrečia antivirusinės įrankį, tai kai antivirusinės įrankis yra įkraunamas į atmintį, jis bus aptiktas programų tikrintojo. Jei virusas pakeičia kokia nors branduolio paprogramę norint pasislėpti, bus aptiktas branduolio tikrintojo, jei virusas bandys pašalinti *SecISR* jis iškart bus aptiktas *SecCore*. Jei ir užpuolikas paleis kelias atakas vienu metu, norint nutraukti tikrinimo grandinė, jos nepavyks tol kol veiks *SecCore*.
- Ataka prieš pradinę vientisumo informacija. Virusas negali pakeisti *SecCore* viešojo rakto, nes jis yra įkraunamas į branduolį ir yra tikrinamas *SecCore* Virusas negali pakeisti vientisumo informacijos nes jis yra apsaugotas su *SecCore* parašu. Tai reiškia kad negalima suklastoti parašo ir pakeisti vientisumo informacijos, kad nebūtų aptiktas *SecCore* taip pat negalima apgauti programos tikrintojo kad būtų skaitomas suklastotas viešasis raktas nepadarius jam pakeitimų.
- Ataka prieš nelegalius atnaujinimus. Virusas negali privesti *SecCore* pasirašyti ant kažko netikėto, nes visa laiką yra reikalaujama vartotojo patvirtinimo. Vartotojui negali būtų suklastotas pranešimas nes jis yra rodomas *SecIO* ekrane. Taip pat virusas negali suklastoti vartotojo įvesties, nes jis neturi kaip valdyti *SecIO*.

1.3.5. **HyperCheck**

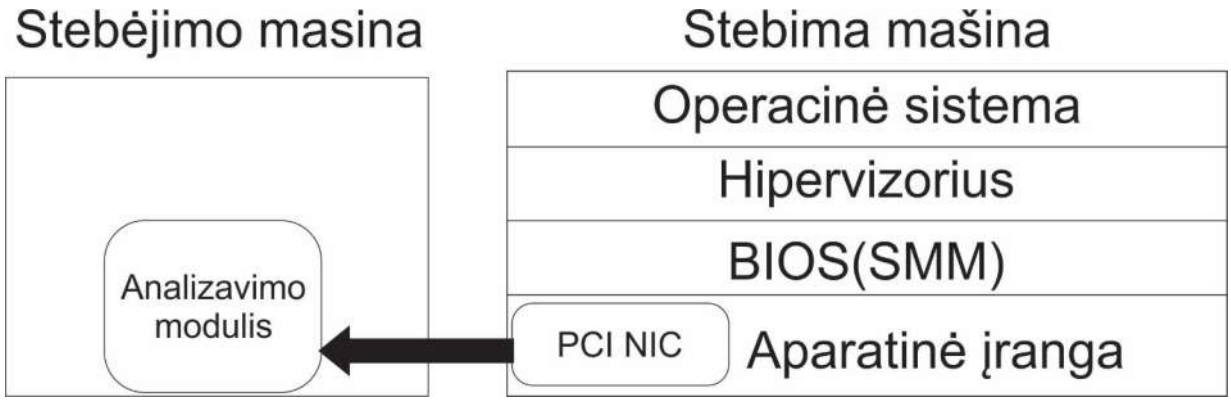
Kitas aparatinis sprendimas leidžiantis užtikrinti operacinės sistemos ar hipervizoriaus vientisumą yra *HyperCheck* [19]. Šis metodas naudoja CPU SMM, kuris yra visuose x86 sistemose, kad būtų sukurtas dabartinės atvaizdas CPU registrų ir atminties būsena apsaugotos mašinos. Ši informacija yra saugiai persiunčiama naudojant tinklo plokštę į nutolusi stebėjimo masiną. Tada stebėjimo masina gali patikrinti ar buvo pakeitimu palyginant naujai sukurtą atvaizdą ir su atvaizdu, kuris buvo sukurtas stebimos mašinos užsikrovimo metu. Jei atvaizdai nesutampa, yra persėjamas administratorius, kad būtų vykdomas tolimesnis tyrimas.

HyperCheck veikia BIOS lygyje. Jis sugeba sudaryti pilną vaizdą stebimos mašimos, į kurią įeina visa fizine atmintis ir CPU registrai naudojant SMM ir bendra PCI aparatinį įrenginį. Be to šis metodas leidžia apsiginti nuo atakų, kurios išjungia ar blokuoja PCI įrenginius dėl to, kad stebėjimo masina gali aptikti tokias DOS atakas.

Kas yra sistemos valdymo režimas (*angl. System management mode*), tai yra atskiras procesoriaus režimas nuo apsaugoto režimo ar realaus režimo. Jis užtikrina skaidrų sistemos valdymo funkcijų mechanizmą pvz. energijos valdymo ir sistemos saugumo. SMM yra diegta pagrindinė įvesties-išvesties sistemoje, BIOS. Į SMM įeinama per sistemos valdymo pertrauktį, SMI, kai yra signalas paduodamas į SMM pertraukimo koją. Mikroprocesorius automatiškai išsaugo visą būseną atskiroje

adresų erdvėje, vadinamoje sistemos valdymo RAM, ir įeina į SMM režimą vykdyti SMI tvarkyklę. Programa įvykdo *rsm* instrukcija, kad išeitu iš SMM režimo. SMRAM yra nepasiekiamas iš kitų CPU režimų kai nėra SMM režime; todėl ji gali būtų kaip patikima saugojimo vieta duomenims.

HyperCheck susideda iš trijų pagrindinių komponentų tai yra: fizinės atminties surinkimo modulis, analizavimo modelis ir CPU registrų tikrinimo modelis. Fizinės atminties ir CPU registrų tikrinimo modeliai veikia saugomam kompiuterį, o analizavimo modulis veikia atskiram stebėjimo kompiuterį. Atminties surinkimo modulis nuskaito atminties turinį apsaugotos mašinos ir ją nusiunčia analizavimo moduliui, kad būtų patikrinta ar yra, kokie kenksmingi pakeitimai. CPU registrų modulis nuskaito CPU registrus ir patikrina jų reikšmes.

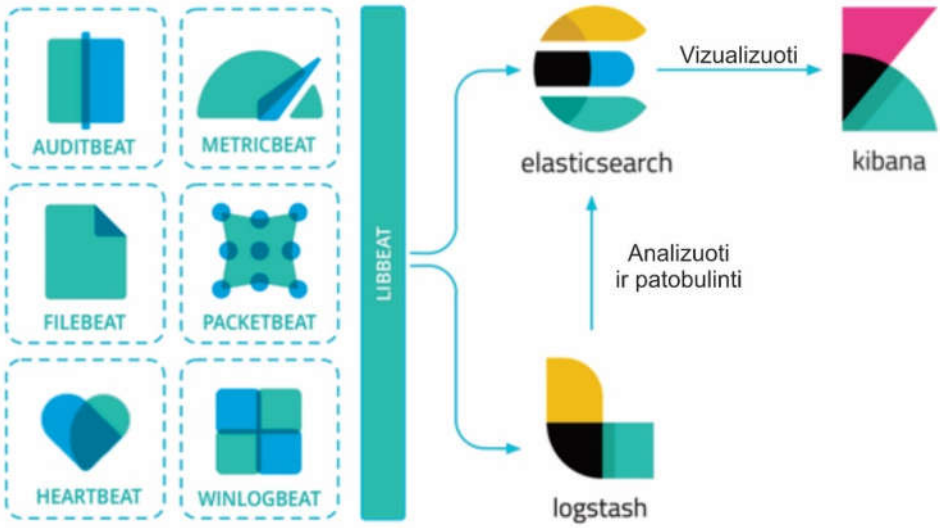


1.10 pav. *Hypercheck* architektūrinė schema

Pagrindinis dizaino principas pavaizduotas 1.10 pav., *HyperCheck* neturėtų pasikliauti jokia programinę įranga veikiančia apsaugotoje mašinoje išskyrus BIOS. Kadangi SMM kodas yra BIOS ir stebėjimo mašina turėtų būti vienintelė patikima kompiuterinė bazė (*angl. TCB*) sistemoje. Todėl galime naudoti SMM, kad būtų nuskaityti CPU registrai ir atminties turinys, o tada naudoti PCI *Ethernet* plokštę, ši informacija būtų išsiųsta į stebėjimo mašina. Paprastai *Ethernet* plokštės yra PCI įrenginiai su įjungtu magistralės režimu ir gali perskaityti fizinę atmintį per DMA, nereikia CPU pagalbos. SMM yra nepriklausomas veikimo režimas ir gali būti neprieinamas iš kitų CPU režimų, kuriuose veikia hipervizoriai ir privilegijuoti domenai.

1.3.6. **Elasticsearch Beats**

Beats tai yra atviro kodo agentai, kurie surenka duomenis iš sistemos ir juo išsiunčia į *Elasticsearch* sistema. Duomenų vizualizacijai gali būti naudojama programa *Kibana* 1.11 pav.



1.11 pav. Elasticsearch *Beats* architektūra

Beats susideda iš kelių komponentų:

- Auditbeat
- Filebeat
- Heartbeat
- Journalbeat
- Metricbeat
- Packetbeat

Kiekvienas komponentas renka skirtingus duomenis apie sistemą, jei vienas iš komponentų nėra reikalingas, to komponento galima nediegti į sistemą, taip nebus kaupiami nereikalingi duomenys ir sistema bus mažiau apkrauta nereikalingomis paslaugomis. [20]

Auditbeat komponentas stebi vartotojų veiklą ir procesus. Jis gali aptikti kritinių failų pakeitimus, identifikuoti potencialius saugumo politikos pažeidimus bei surinkti duomenis iš *Linux* audito karkaso. Failų vientisumui stebėti yra naudojamos žinutės kurioje yra failo meta duomenys bei apskaičiuota failo kriptografinė maišos funkcijos reikšmė. Šios žinutės yra siunčiamos į *Elasticsearch* steką arba su *Logstash* suderinama sistemą. Jei žinučių *Auditbeat* negali išsiųsti dėl laikinų tinklo sutrikimų, jos yra laikinai laikomos lokaliame diske iki tol kol sutrikimai yra pašalinami ir žinutes išsiunčiamos, taip užtikrinant, kad svarbūs įvykiai nėra praleidžiami dėl tinklo gedimo.

Filebeat tai komponentas skirtas sistemos žurnalų persiuntimui ir centralizavimui. Jis surenka žurnalo įvykius ir juos persiunčia į *Elaticsearch* arba *Logstash* indeksavimui. Kiekvienam stebimam žurnalo failui yra sukuriamas surinkėjas, kuris ieško naujų įrašų žurnale ir juos persiunčia į *Elaticsearch* arba

Logstash. Jeigu *Elaticsearch* ar *Logstash* yra perkrauta *Filebeat* automatiškai pradės siųsti mažiau duomenų, kol apkrova sumažės.

Heartbeat tai komponentas, kuris tikrina paslaugų būsena ir pasiekiamumui tikrinti. Tikrinimui galima naudoti ICMP *echo* prašymo paketus, TCP susijungimus arba HTTP. Naudojant ICMP *echo* prašymo paketus tiesiog yra patikrinama ar paslauga yra pasiekiama, su TCP susijungimais galima nustatyti tam tikrus gautinius taškus, kurie gražina nustatytas reikšmes ir taip nustatant ar paslauga tiekiama teisingai. Su HTTP susijungimu galima gauti būsenų kodus ar net turinį kurio pagalba galima nustatyti ar paslauga veikia teisingai. Taip pat TCP ir HTTP stebėtojai palaiko SSL/TLS šifravimą.

Journalbeat kaip ir *Filebeat* suranka sistemos žurnalus ir juos persiunčia į *Elaticsearch* arba *Logstash* tačiau jis tik surenka duomenis iš *systemd* žurnalo.

Metricbeat komponentas suranka iš sistemos metrikas ir statistikas ir jas persiunčia į *Elaticsearch* arba *Logstash*. Komponentas gali surinkti tokius duomenis kaip procesoriaus apkrovas, atminties panaudojimą, kietojo disko užimtumą, tinklo apkrova bei kitus statistinius sistemos duomenis.

Packetbeat tai yra realaus laiko tinklo paketų analizatorius. Jo veikimas yra toks, kad jis gaudo tinklo srautą tarp kelių aplikacijos serveriu, dekoduoja aplikacijos lygio protokolus, koreliuoja prašymus su atsakymais ir įsirašo įdomius kiekvienos transakcijos laukus, taip padėdamas lengviau pastebėti problemas susijusias su aplikacijomis kaip programavimo klaidas ar našumo problemas.

1.4. Analizės išvados

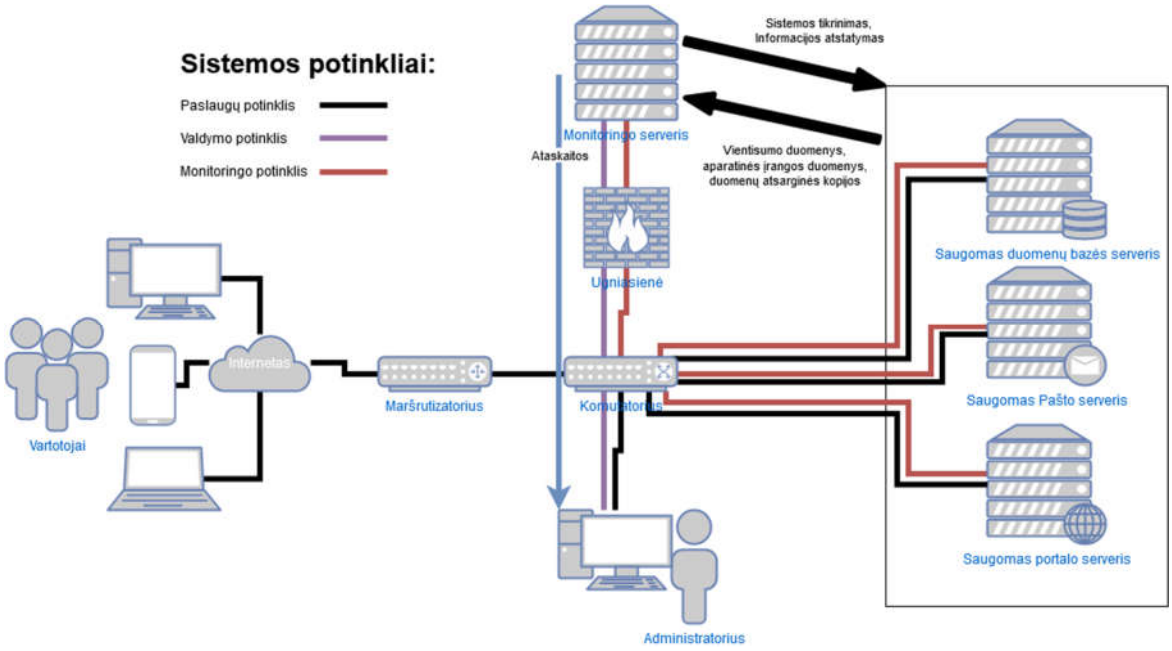
- Neužtikrinus kompiuterinės sistemos vientisumo, duomenys esantys patalpinti informacinėje sistemoje gali prarasti savo prasmę bei vertę ir taip sutrikdyti įmonės darbą. Taip pat pažeidus sistemos vientisumą, pakeitus sistemos nustatymus ar įdiegus papildomą neautorizuota aparatinę įranga, gali būti pažeistas iš sistemoje esančių duomenų konfidencialumas ar pasiekiamumas.
- Užtikrinti kompiuterines sistemos vientisumą galima įvairiais būdais, tai galima atlikti naudojant programinius sprendimus arba aparatinius, tačiau nėra vieno bendro sprendimo, kad būtų užtikrintas sistemos vientisumas.
- Vieni metodai pasitelkia trečiąsias šalis, kad gauti patikima žetoną ir jo pagalba yra skaičiuojama maišos reikšmė, kiti metodai talpina maišos reikšmes į blokų grandinę, kad apsunkinti jų modifikaciją, taip pat vieni metodai siunčia vientisumo duomenis į atskirą patikimą serveri, kad būtų atliktas patikrinimas taip atsiskiriant nuo mašinos kuri potencialai gali būti pažeista, dar kiti metodai leidžia užtikrinti sistemos atmintyje vientisumą.
- Yra metodų leidžiančių atstatyti pažeistą sistemos vientisumą.
- Taip pat analizuotus metodus galima išplėsti, kad atliktų aparatinės įrangos vientisumo patikrą.
- Analizuoti produktai turi problema susijusia su programinės įrangos atnaujinimu. Metodai, kurie remiasi aparatiniai įranga, reikalauja kiekviena karta atnaujinus programinę įrangą patvirtinti ar tai leistinas atnaujinimas ir tai privalo padaryt žmogus. Su programiniais metodais kiekvieną karta reikia pergeneruoti vientisumo duomenų bazes ir jas paskui saugiai talpinti, pavyzdžiui perkelti į tik skaitymui skirtas talpyklas.
- Atlikus analizė buvo iškelti šie uždaviniai:

- Sukurti kompiuterinės sistemos vientisumo užtikrinimo modelį pasiremiant keliais analizuotais metodais ir produktais;
- Remiantis sudarytu modeliu sukurti sistemos prototipą;
- Naudojantis prototipu atlikti sistemos greitaveikos bei lygiagrečių sistemų tikrinimo tyrimus;
- Palyginti sukurtą sistemą su dabar egzistuojančiais produktais.

2. Kompiuterinės sistemos vientisumo užtikrinimo modelis

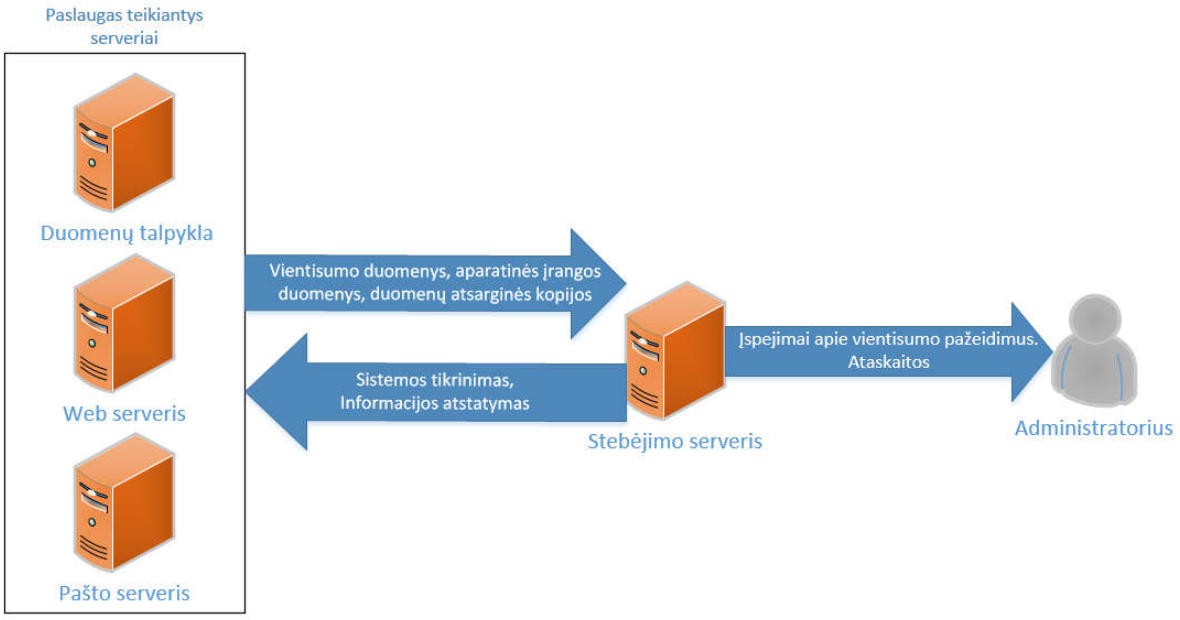
Šiame skyriuje aprašomas siūlomas kompiuterinės sistemos vientisumo užtikrinimo modelis.

2.1. Koncepcinis kompiuterinės sistemos vientisumo užtikrinimo modelis



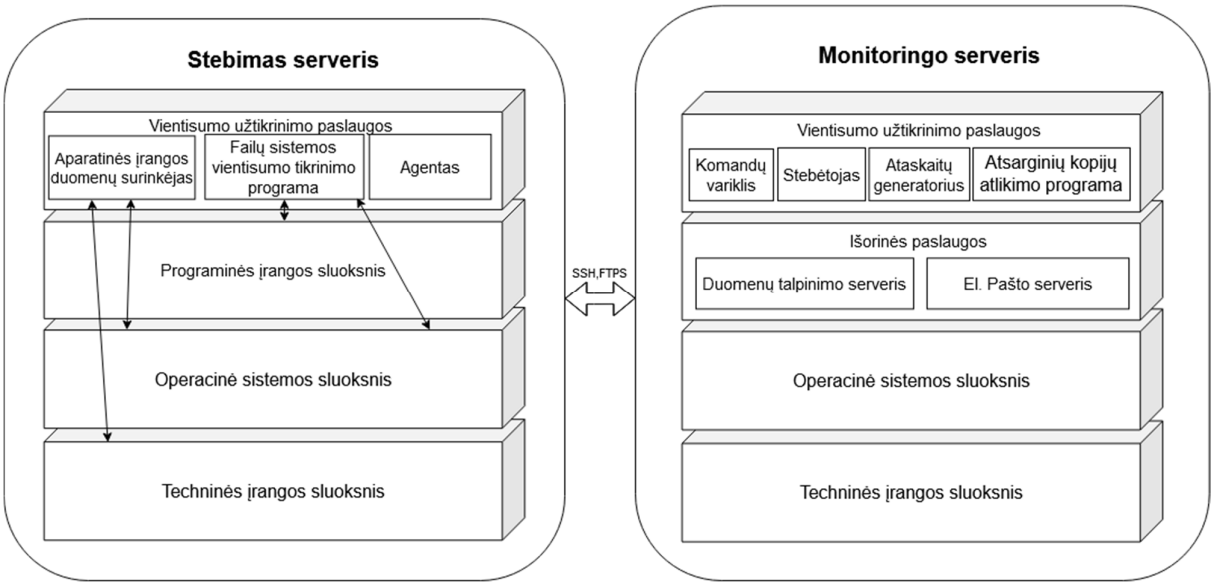
2.1 pav. Aukšto lygio kompiuterinės sistemos vientisumo užtikrinimo modelio schema

2.1 pav. pavaizduota aukšto lygio kompiuterinės sistemos vientisumo užtikrinimo modelio schema. Schemoje yra pavaizduoti serveriai, kurie yra prijunkti prie monitoringo serverio per jiems skirtą tikrinimo potinklį. monitoringo serveris gali aptarnauti daug stebimų serverių. Per šį potinklį, monitoringo serveris gali prisijungti prie saugomų serverių ir patikrinti šių serverių vientisumą. Monitoringo serveris nėra tiesiogiai pasiekiamas iš interneto ir yra apsaugotas ugniasiene per kuria yra leidžiami prisijungimai į stebėjimo serverį tik per valdymo potinklį. Saugomi serveri taip pat negali prisijungti prie monitoringo serverio tačiau jie gali „pasibelsti“, taip informuojant stebėjimo serverį jog reikia atlikti vientisumo tikrinimą serveryje, kuris „beldėsi“.



2.2 pav. Koncepcinis kompiuterinės sistemos vientisumo užtikrinimo modelis.

Ši supaprastinta sistemos koncepcija, į kuria yra sutelktas šis darbas, yra pavaizduotas 2.2 pav. Ši koncepcija sudaro stebėjimo serveris, kuriame yra saugomi duomenys apie stebimus serverius šie duomenys yra laikomi tik skaitymo režimu, kad jų nebūtų galima pakeisti. Pakeisti duomenis galima tik atnaujinimo metu patvirtinus administratoriui. Įvykus vientisumo pažeidimui iš stebėjimo serverio yra paimamos atsarginės failų kopijos ir jos gražinamos į pradinė būsena prieš pažeidimui įvykstant. Administratoriui yra siunčiamos ataskaitos apie kompiuterinės sistemos vientisumo pažeidimus ir ar pažeidimai buvo ištaisyti.



2.3 pav. Siūloma sistemos architektūra

Sistemos architektūra 2.3 pav. yra sudaryta iš kelių modulių, kurie atlieka tam tikras funkcijas:

1. Stebimas serveris:

   - **Failų sistemos vientisumo tikrinimo programa** – Programinė įranga skirta failų sistemos vientisumui užtikrinti. Numatoma naudoti *Tripwire* [13]. *Tripwire* pagalba yra surenkami duomenis apie dabartine sistemos būsena, tai yra paskaičiuojamos maišos funkcijos sumos stebimiems failams ir surenkami kiti parametrai apie stebimus failus kaip failo dydis, *inode* numeris, modifikavimo data, failo teises ir kitos.
   - **Aparatinės įrangos duomenų surinkėjas** – Programinė įranga skirta surinkti duomenis apie dabartinę aparatinės įrangos būseną. Čia numatoma pasitelkti keliomis programomis kaip *dmidecode* [21] ir *Lshw* [22]. Duomenų surinkėjas surinka duomenis iš *dmidecode* ir *Lshw*, šiuos duomenis išskaido pagal kategorijas ir patalpina į atskirus failus, kad būtų galima paskui patikrinti kokie pakeitimai aparatinėje įrangoje buvo atlikti. *Dmidecode* surenka duomenis iš SMBIOS apie sistemos BIOS, procesorių, darbinę atmintį, pagrindinę plokštę. *Lshw* – papildo *dmidecode* surinkdamas duomenis apie įrenginius prijungtus per PCI jungtis, standžiųjų diskų informacija, vaizdo įrenginius, tinklo įrenginius.
   - **Agentas** – tai naujai kuriama tarnyba, kuri išsiunčia pranešimą į monitoringo serverį, kad stebimam serverio reikia atlikti vientisumo tikrinimą. Agentas siunčia pranešimą, kai prijungiama ar atjungiama aparatinė įranga arba kai yra modifikuojamas stebimas, kritinis sistemos failas.

2. Monitoringo serveris:

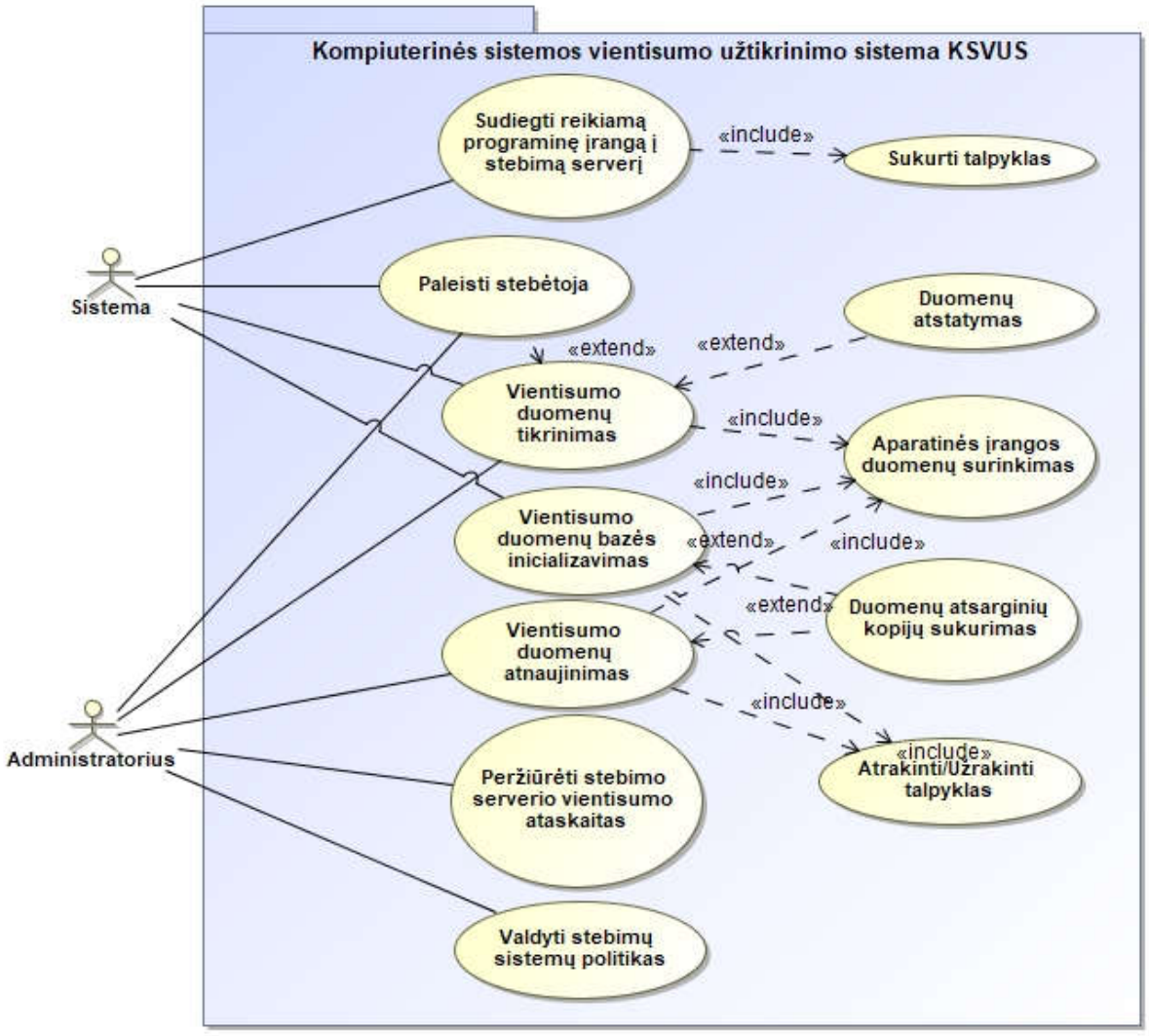
   - **Komandų variklis** – tai yra komponentas, kuris iškviečia funkcijas stebimame serveryje pasitelkiant sukurtais, pagalbiniais *bash* scenarijais, taip pat ir atlieka funkcijas ir monitoringo serveryje, funkcijas kaip talpyklų Atrakinimas ir užrakinimas.
   - **Stebėtojas** – tai kuriam programa, kuri laukia pranešimų iš agentų apie reikalingą vientisumo patikrinimą stebimuose serveriuose.
   - **Ataskaitų generatorius** – tai įrankis, kuris surinks visas tarpines ataskaitas kaip *Tripwire* vientisumo pažeidimų ataskaitą, aparatinės įrangos pokyčių ataskaitą, failų iš atsarginių kopijų atstatymų ataskaitą ir vientisumo tikrinimo įvykdymo ataskaitą, jas apdoros ir apjungs į viena ir išsiųs administratoriams patikrinimui.
   - **Atsarginių kopijų atlikimo programa** – tai įrankis, kuris atliks atsarginė kopiją tų failų, kurie yra nurodyti saugos politikoje, šie failai bus patalpinti į duomenų talpinimo serverį.
   - **El. Pašto serveris –** Šio komponento pagalba ataskaitų generatorius siųs ataskaitas apie stebimo serverio vientisumo būseną bei klaidas, jei jos buvo nustatytos administratoriams.
   - **Duomenų talpinimo serveris –** šio komponento pagalba bus saugojami stebimų serverių vientisumo duomenys bei atsarginės duomenų kopijos. Kiekvienas stebimas serveris turi savo atskirą talpykla, kurios rašymo teisės bus valdomos komandų variklio.

Modulius, kuriuos reikės sukurti, norint įgyvendinti modelį yra: Aparatinės įrangos duomenų surinkėjas, agentas, komandų variklis, stebėtojas, ataskaitų generatorius ir Atsarginių kopijų atlikimo programa.

Komunikacija tarp stebimo serverio ir stebėjimo serverio vyksta naudojant SSH protokolą.

Šio protokolo dėka komandos ir duomenys yra siunčiami šifruotu kanalu. Failų nusiuntimui į stebėjimo serveri yra naudojamas FTP protokolas, tačiau FTP protokolas yra atviro teksto protokolas

ir duomenis bei komandos yra siunčiamos neapsaugotai, dėl šios priežasties rekomenduojama naudoti FTPs, kuris yra saugus FTP, duomenys bei komandos yra šifruojamos SSL/TLS pagalba.



2.4 pav. Vientisumo užtikrinimo sistemos panaudos atvejų diagrama

Paveiksle 2.4 pav. yra pateikta vientisumo užtikrinimo sistemos panaudos atvejų diagrama. Šitoje diagramoje matome, kokiomis funkcijomis administratorius gali naudotis. Detalesnei panaudos atvejų aprašymai pateikti lentelėje 2.1.

2.1 lentelė Panaudos atvejai

Panaudos atvejis:	**Sudiegti reikiama programinę įrangą į stebimą serverį**
Tikslas: Paruošti stebimą serverį, kad būtų galima stebėti sistemos vientisumą	Aprašymas: Į stebimą serverį reikia sudiegti reikiamą programinę įrangą, kad būtų galimą stebėti sistemos vientisumą. Taip pat šiuo metu yra sukuriamos talpyklos stebėjimo serveryje.
Aktorius: Sistema	
Susiję panaudos atvejai: Sukurti talpyklas	

Panaudos atvejis:	**Sukurti talpyklas**
Tikslas: Paruošti stebėjimo serveri talpyklas kurias naudosis stebimi serveriai	Aprašymas: Stebėjimo serveryje sukuriama atskira talpykla, kurioje yra saugoma informacija apie dabartinę stebimo serverio vientisumo būseną bei atsarginės duomenų kopijos.
Aktorius: Sistema	
Susiję panaudos atvejai: Sudiegti reikiama programinę įrangą į stebimą serverį	

Panaudos atvejis:	**Vientisumo duomenų bazės inicializavimas**
Tikslas: sukurti pradinę stebimos sistemos vientisumo duomenų atvaizdą	Aprašymas: Sukurti duomenų baze su pradinę stebimos sistemos vientisumo informacija bei surinkti duomenis apie dabartinę aparatinės įrangos būseną ir patalpinti į stebėjimo serverio talpyklą. Taip pat atlikti ir failų, kurie pažymėti atsargines kopijas.
Aktorius: Sistema	
Susiję panaudos atvejai: Aparatinės įrangos duomenų surinkimas Atrakinti/užrakinti talpyklas Duomenų atsarginių kopijų atlikimas	

Panaudos atvejis:	**Aparatinės įrangos duomenų surinkimas**
Tikslas: surinkti duomenis apie aparatinę įrangą	Aprašymas: Surenka duomenis apie dabartinę stebimos sistemos aparatinės įrangos būseną, kad butų galima nustatyti ar buvo sistemoje pasikeitimų.
Aktorius: Administratorius, Sistema	
Susiję panaudos atvejai: Vientisumo duomenų bazės inicializavimas Vientisumo duomenų tikrinimas	

Panaudos atvejis:	**Vientisumo duomenų tikrinimas**
Tikslas: Patikrinti ar sistemos vientisumas yra išsaugotas	Aprašymas: Atliekamas stebimos sistemos vientisumo patikrinimas pagal sudaryta politika, jei atsiranda neatitikimų su duomenų baze kuri yra stebėjimo serveryje, yra siunčiama ataskaita administratoriui su failų pavadinimais kuriu vientisumas yra pažeistas. Be to jei failas buvo pažymėtas, kad reikia atlikti atsarginę kopija tas failas yra iškart atstatomas ir informuojamas administratorius kad buvo pažeidimas ir ar pavyko atstatyti failą. Taip pat yra surenkami duomenis apie dabartinė aparatinės įrangos būsena ir patikrinama ar nebuvo pasikeitimų nuo paskutinio atnaujinimo.
Aktorius: Administratorius, Sistema	
Susiję panaudos atvejai: Duomenų atstatymas Aparatinės įrangos duomenų surinkimas	

Panaudos atvejis:	**Paleisti stebėtoja**
Tikslas: Paleisti paslauga kuri stebimam serveriui išsikviesti sistemos vientisumo tikrinimą	Aprašymas: Paleidžiamas stebėtojas stebėjimo serveryje, kuris leidžia klausosi įvykiu iš stebimų serverių ir jei reikia paleidžia vientisumo tikrinimą
Aktorius: Administratorius, Sistema	

Susiję panaudos atvejai: Vientisumo duomenų tikrinimas	

Panaudos atvejis:	**Duomenų atstatymas**
Tikslas: Atstatyti duomenis iš atsarginių kopijų vykus vientisumo pažeidimui	Aprašymas: Įvykus stebimos sistemos vientisumo pažeidimui jei yra pažeisto failo atsarginė kopija stebėjimo serveryje, tai failas yra atstatomas iš šios rezervinės kopijos.
Aktorius: Administratorius, Sistema	
Susiję panaudos atvejai: Vientisumo duomenų tikrinimas	

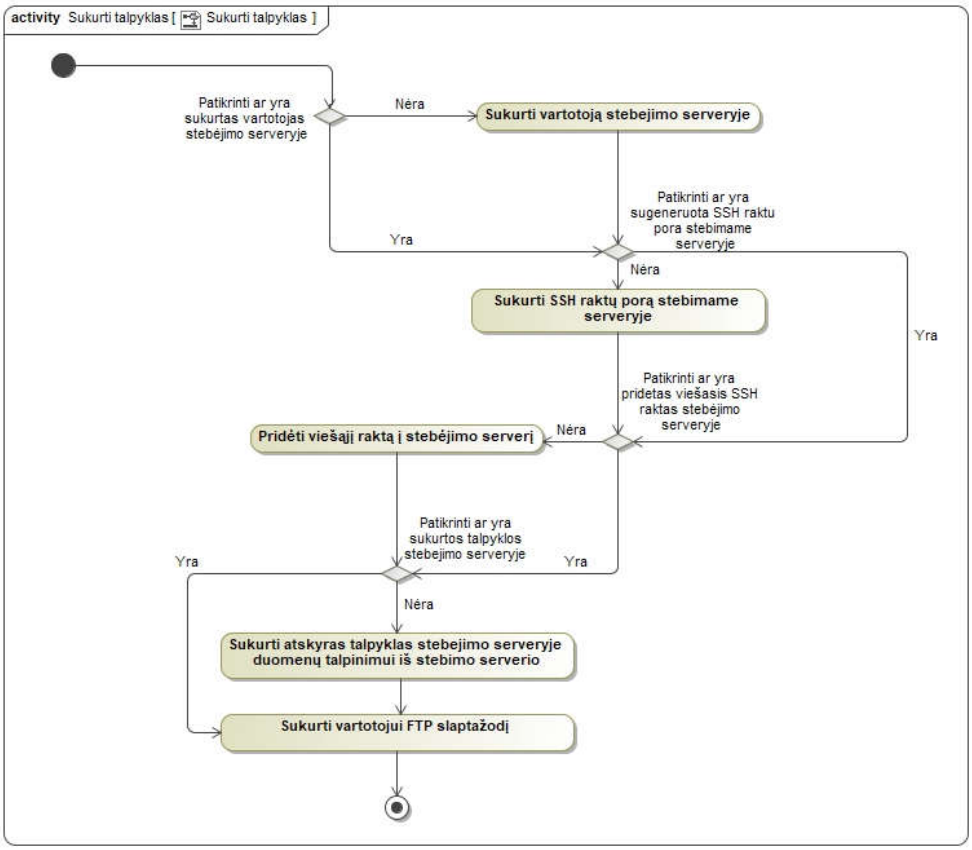
Panaudos atvejis:	**Vientisumo duomenų atnaujinimas**
Tikslas: Atnaujinti turimus vientisumo duomenis	Aprašymas: Jei reikia stebimą sistemą atnaujinti arba pakeisti jos konfigūraciją, reikia atlikti vientisumo duomenų atnaujinimą, atlikti naujas failų atsargines kopijas bei atlikti aparatinės įrangos tikrinimą ir išsaugoti juos stebėjimo sistemos talpykloje.
Aktorius: Administratorius	
Susiję panaudos atvejai: Duomenų atsarginių kopijų atlikimas Atrakinti užrakinti talpyklas Aparatinės įrangos duomenų surinkimas	

Panaudos atvejis:	**Duomenų atsarginių kopijų atlikimas**
Tikslas: Atlikti pažymėtų failų atsargines kopijas	Aprašymas: Failai, kurie pažymėti, kad reikia atlikti atsargines kopijas yra nukopijuojami į stebėjimo serverio talpykla.
Aktorius: Administratorius	
Susiję panaudos atvejai: Vientisumo duomenų atnaujinimas	

Panaudos atvejis:	**Peržiūrėti stebimo serverio vientisumo ataskaitas**
Tikslas: Leisti aktoriui peržiūrėti stebimos sistemos vientisumo duomenų ataskaitas	Aprašymas: Administratorių yra siunčiamos ataskaitos apie stebimo serverio vientisumo pažeidimus, atstatytus failus ir aparatinės įrangos pasikeitimus.
Aktorius: Administratorius	

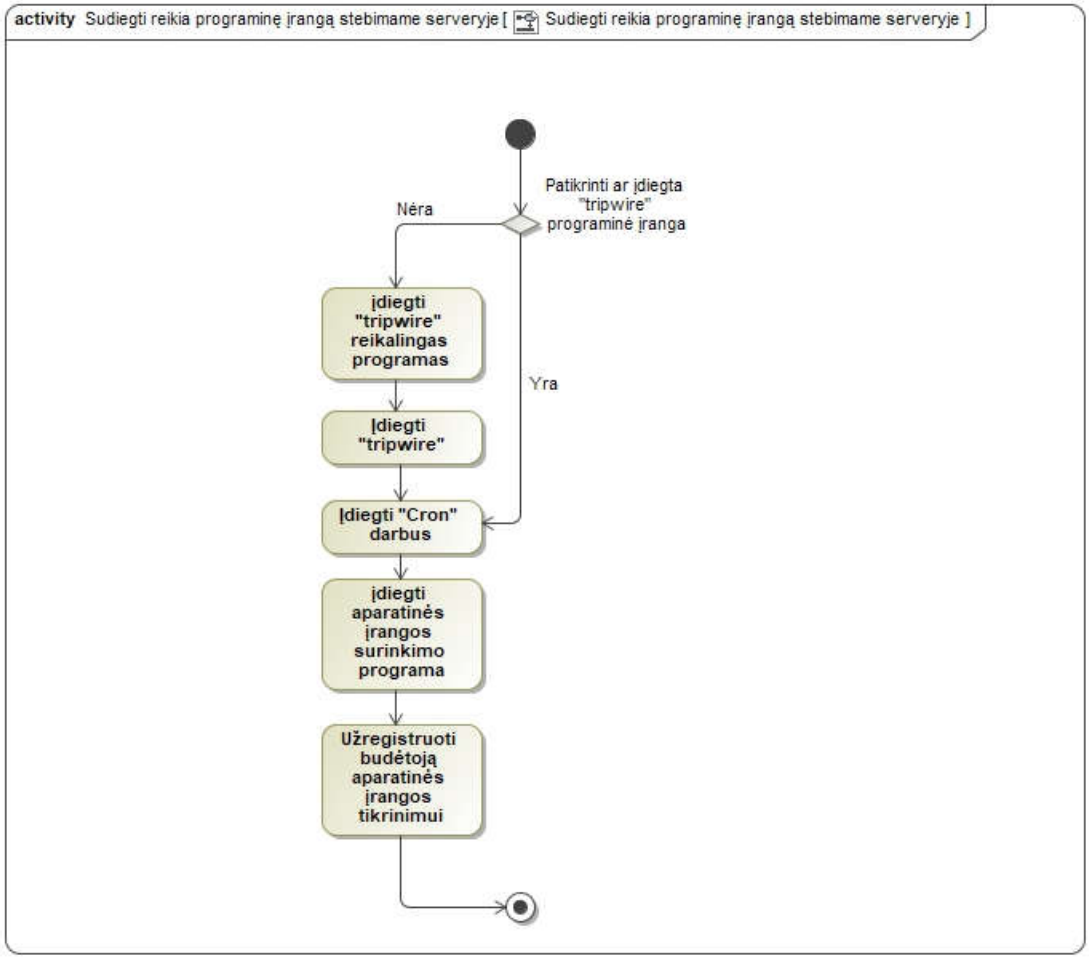
Panaudos atvejis:	**Valdyti stebimų sistemų politikas**
Tikslas: Leisti sudaryti bei keisti stebimų sistemų vientisumo politikas	Aprašymas: Administratorius sudaro politikas kokių failų kokius parametrus reikia stebėti, ar reikia daryti tam tikrų failų atsargines kopijas ir ar reikia stebėti aparatinės įrangos pasikeitimus.
Aktorius: Administratorius	

Toliau yra aprašytos panaudos atvejų veiklos diagramos.



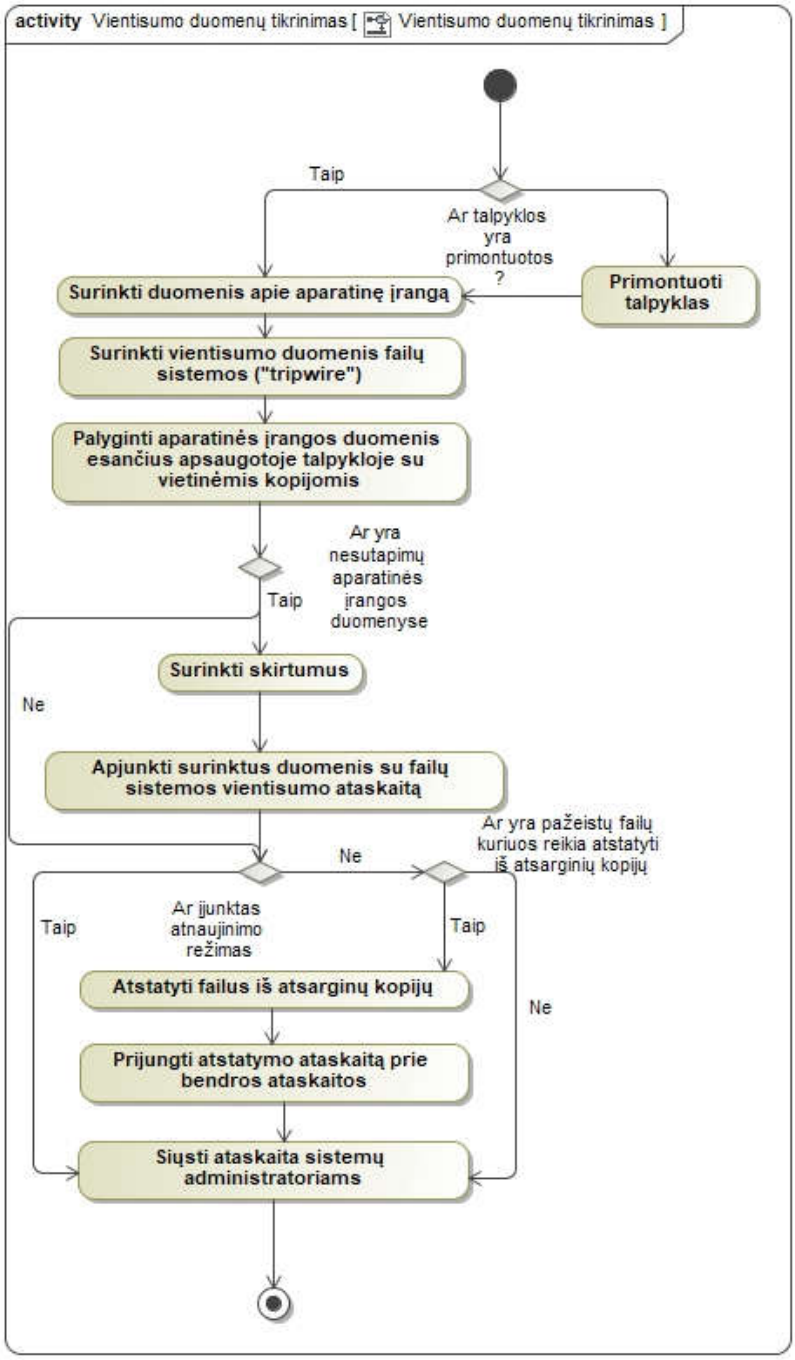
2.5 pav. Panaudos atvejo „Sukurti talpyklas“ veiklos diagrama

2.5 pav. pavaizduotas „Sukurti talpyklas“ panaudos atvejo veiklos diagrama. Kad sistema išliktų agnostinė yra atliekami tikrinimai su kiekvienu žingsniu, kad butų galima atlikti užduotis kelis kartus jei to reiktų. Sistema sukuria vartotoją stebėjimo serveryje, kad paskiau naudojantys to vartotojo prisijungimais galėtų pasiekti savo talpyklas, taip pat yra sukuriami SSH raktai ir jei įtraukiami į stebėjimo serverį. Čia galioja apribojimai, kad stebimas serveris galėtų tik informuoti stebėjimo serveri kad reikia atlikti tikrinimą, nes atėjo laikas ar serveris buvo iš naujo paleistas ar suveikė budėtojas. Taip pat apribojimai yra iš kokių adresų galimą prisijungti prie naudojant šiuos prisijungimo duomenis. Yra sukuriamos talpyklos kurios priklauso tik tam serveriui ir vartotojui. Kiti serveriai ir vartotojai negali matyti vienas kitų talpyklas. Yra sukuriamas vartotojui slaptažodis, vėliau galėtų prisimontuoti šią talpyklą per *SSHfs*.



2.6 pav. Panaudos atvejo „Sudiegti reikiama programinę įrangą stebimame serveryje“ veiklos diagrama

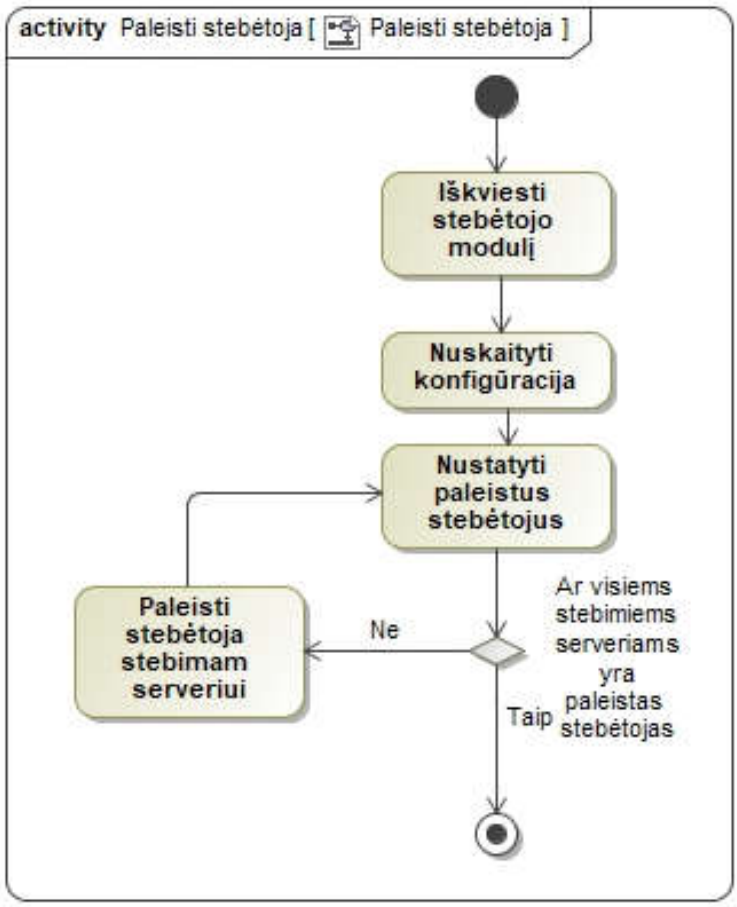
Panaudos atvejyje „Sudiegti reikiama programinę įrangą stebimame serveryje“ 2.6 pav. yra patikrinama ar jau yra sudiegta reikiama programinė įranga ir įdiegiamos tarnybos kurių pagalba bus informuojamas stebėjimo serveris, kad reikia atlikti vientisumo tikrinimą. Įdiegiami *cron* darbai, po tam tikro laiko kurį nustato administratorius butų vykdomas tikrinimas arba serveriui kai yra iš naujo paleistas, kad iškart būtų atliktas tikrinimas taip pat yra registruojamas budėtojas kuris tikriną ar nebuvo pakeitimų su aparatinę įrangą pvz. prijunktas USB raktas.



2.7 pav. Panaudos atvejo „Vientisumo duomenų tikrinimas“ veiklos diagrama

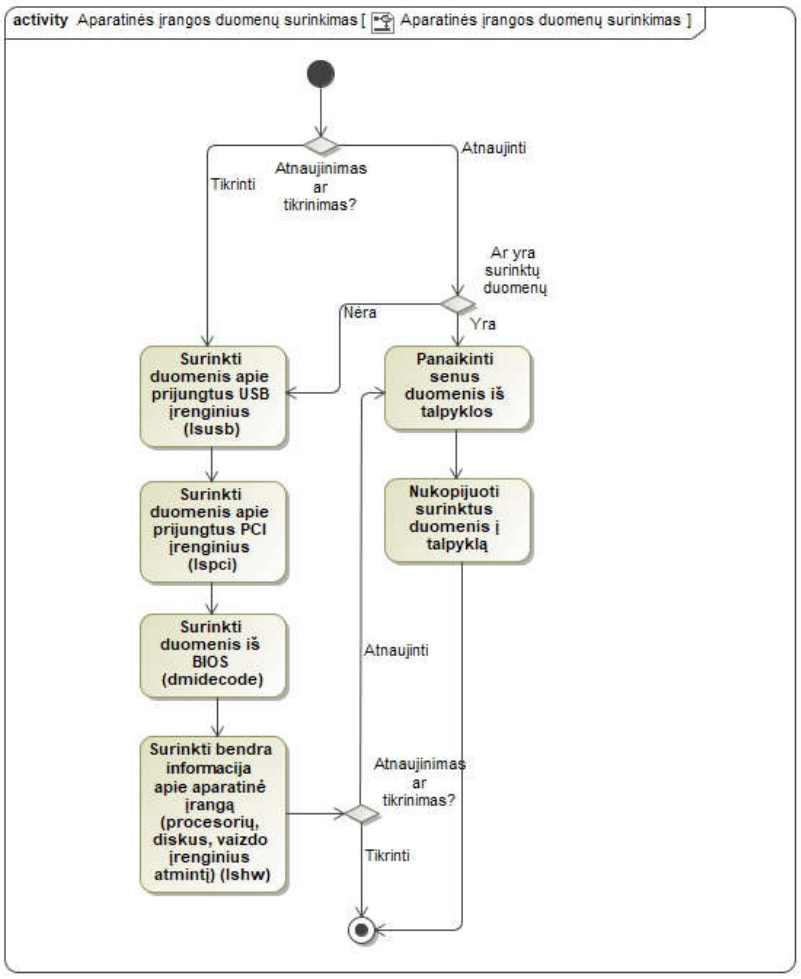
Panaudos atvejis „Vientisumo duomenų tikrinimas“ 2.7 pav. atlieką sistemos vientisumo tikrinimą. Yra patikrinama ar galimą atlikti tikrinimą tai yra ar talpyklos yra primontuotos prie serverio jei nėra juos yra primontuojamos ir pranešama, kad reikėjo primontuoti iš naujo talpyklas, toks atvejis gali nutikti jei buvo tinklo sutrikimų. Yra surenkami duomenys apie aparatinė įrangą (detaliau kitame panaudos atvejyje) ir taip pat yra surenkami duomenys apie failų sistemos vientisumą pasinaudojus *Tripwire* programinę įrangą. Yra atliekamas palyginimas tarp aparatinės įrangos duomenų esančių stebėjimo serverio talpykloje ir ką tik surinktuose, jei yra neatitikimų atliekamas skirtumų patikrinimas ir ši informacija apie prisegama prie pagrindinės ataskaitos. Jei nėra įjunktas atnaujinimo

režimas ir įjunktas failų atstatymas, failai kurių vientisumas buvo pažeistas ir buvo atliktos jų atsarginės kopijos yra atstatomos iš talpyklos, kartu su turiniu yra atstatomos ir failo teisės ir failo atributai. Jei buvo tokių failų, kuriuos reikėjo atstatyti, yra prisegama apie ataskaitos ar kokie failai buvo pažeisti ir ar pavyko atstatyti ar ne. Visa bendra ataskaitą yra siunčiama administratoriams.



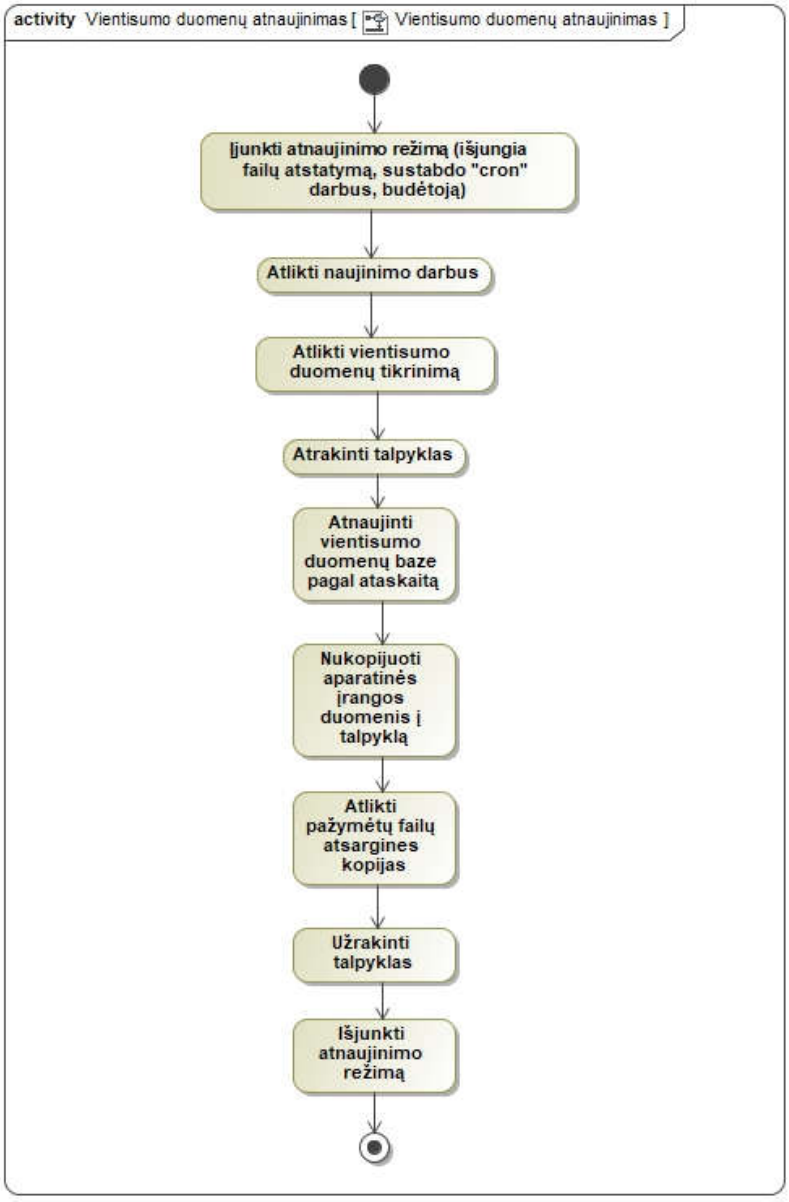
2.8 pav. Panaudos atvejo „Paleisti stebėtoja“ veiklos diagrama

Panaudos atvejyje „Paleisti stebėtoją“ 2.8 pav. yra iškviečiamas stebėtojo modulis, kuris nuskaito stebimų serverių sąrašą ir paleidžia kiekvienam stebimam serveriui po nepriklausoma paslauga, kuris klausosi įvykių iš stebimo serverio (prijunka ar atjunkta aparatinė įranga, pasikeitė kritiniai sistemos failai) ir nutikus įvykiui yra atliekamas sistemos vientisumo patikrinimas.



2.9 pav. Panaudos atvejo „Aparatinės įrangos duomenų surinkimas“ veiklos diagrama

Panaudos atvejyje „Aparatinės įrangos duomenų surinkimas“ 2.9 pav. pasinaudojus įrankiais kaip *lsusb*, *lspci*, *dmidecode*, *lshw* yra surenkama dabartinė aparatinės įrangos būsena jei yra atliekamas vientisumo duomenų atnaujinimas duomenys yra nukopijuojami į talpyklą kad paskui būtų galimą žinoti kokie įrenginiai buvo prijunkti ar pašalinti.



2.10 pav. Panaudos atvejo „Vientisumo duomenų atnaujinimas“ veiklos diagrama

Panaudos atvejis „Vientisumo duomenų atnaujinimas“ 2.10 pav. yra reikalingas, kai sistemoje teisėtai yra pakeičiami nustatymai, atnaujinama programinė įranga, ar prijungiama ar atjungiama aparatinė įranga, būtų galima atnaujinti vientisumo duomenų bazes bei kurios aparatinės įrangos dalys turi likti prijunktos. Atnaujinimo metus stebimas serveris yra pažeidžiamas, todėl administratorius turi prižiūrėti, kad nebūtų nesankcionuotų pakeitimų tiek programinėje įrangoje tiek aparatinėje. Atnaujinimo metu yra išjungiami *cron* darbai, failų atstatymas, budėtojas, tam, kad jei susižadintų tikrinimas nebūtų prarastas administratoriau naujinimo darbas. Administratoriui atlikus naujinimo darbus yra atliekamas vientisumo tikrinimas, kad galėtų patikrinti ar visi pakeitimai yra teisėti prieš atnaujinant vientisumo duomenų bazę. Jei viskas yra gerai, yra atrakinama talpykla (įgalinamas rašymas į talpyklą) ir atnaujinama vientisumo duomenų bazė pagal paskutinę ataskaitą. Atliekamas aparatinės įrangos tikrinimas ir duomenys nukopijuojami į talpyklą. Taip pat yra atliekamas

pažymėtų, pagal politiką, failų atsarginės kopijos. Pabaigoje yra užrakinama talpykla kad nebūtų nesankcionuotai pakeisti duomenų ir išjungimas atnaujinimo režimas.

Sistema renka duomenis apie failus, kurie yra aprašyti politikoje, ir tai yra failų:

- Atidarymo laikas;
- Priskirtų blokų skaičius;
- *inode* sukūrimo ir modifikacijos laikas ir data;
- Identifikacinis numeris įrenginio kuriame yra failo *inode*;
- Savininko grupė;
- *inode* numeris;
- Ar failas yra augantis;
- Modifikacijos data ir laikas;
- Jungčių skaičius;
- Failo teises;
- Identifikacijos numeris įrenginio į kuri rodo *inode* galioja tik įrenginių objektams;
- Failo dydis;
- Failo tipas;
- Savininko vartotojo numeris.

Taip pat galima apskaičiuoti failų maišos funkcijos reikšmes naudojant šiuos algoritmus ir juos saugoti:

- CRC-32;
- HAVAL;
- MD5;
- SHA1.

Sistema tikrina ar buvo pridėtų, modifikuotų ar panaikintų failų stebimuose kataloguose pagal nustatytą politiką.

Sistema surenka duomenis ir apie aparatinė įrangą pasitelkdama keletą įrankių:

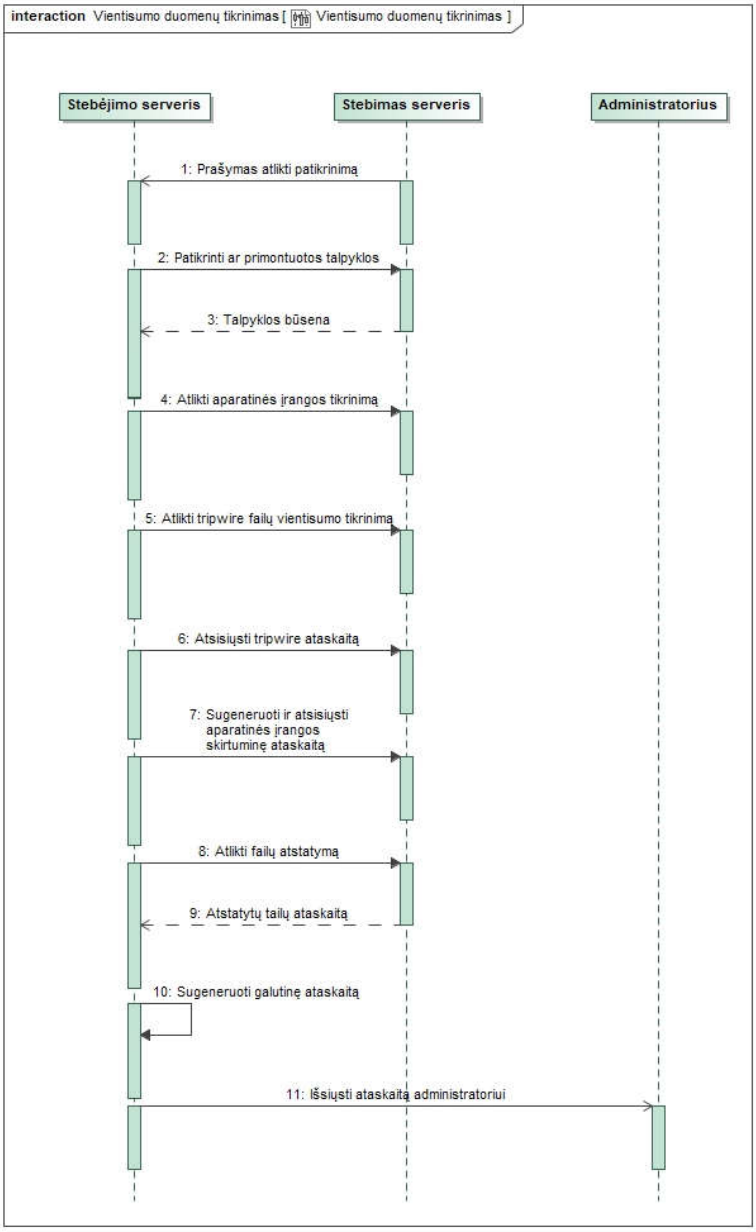
1. *lsusb* – informacija apie prijunktus USB įrenginius;
2. *lspci* - informacija apie prijunktus PCI/PCI-E įrenginius;
3. *dmidecode* – informacija apie įrenginį iš BIOS;

    - BIOS (versija);
    - Pagrindinę plokštę (modelį, serijinį numerį);
    - Procesorių (modelis, serijinis numeris);
    - Darbinę atmintį (Kiekis, modelis, serijinis numeris, užpildytas vietas);

4. *lshw* – informacija apie įrenginį iš OS;

    - informacijos kaupikliai (USB, SATA, PATA, SCSI) (modelis, serijinis numeris, dydis, montavimo vieta failų sistemoje);
    - vaizdo įrenginiai (modelis);
    - Tinklo įrenginiai (modelis, mac adresas, būsena).

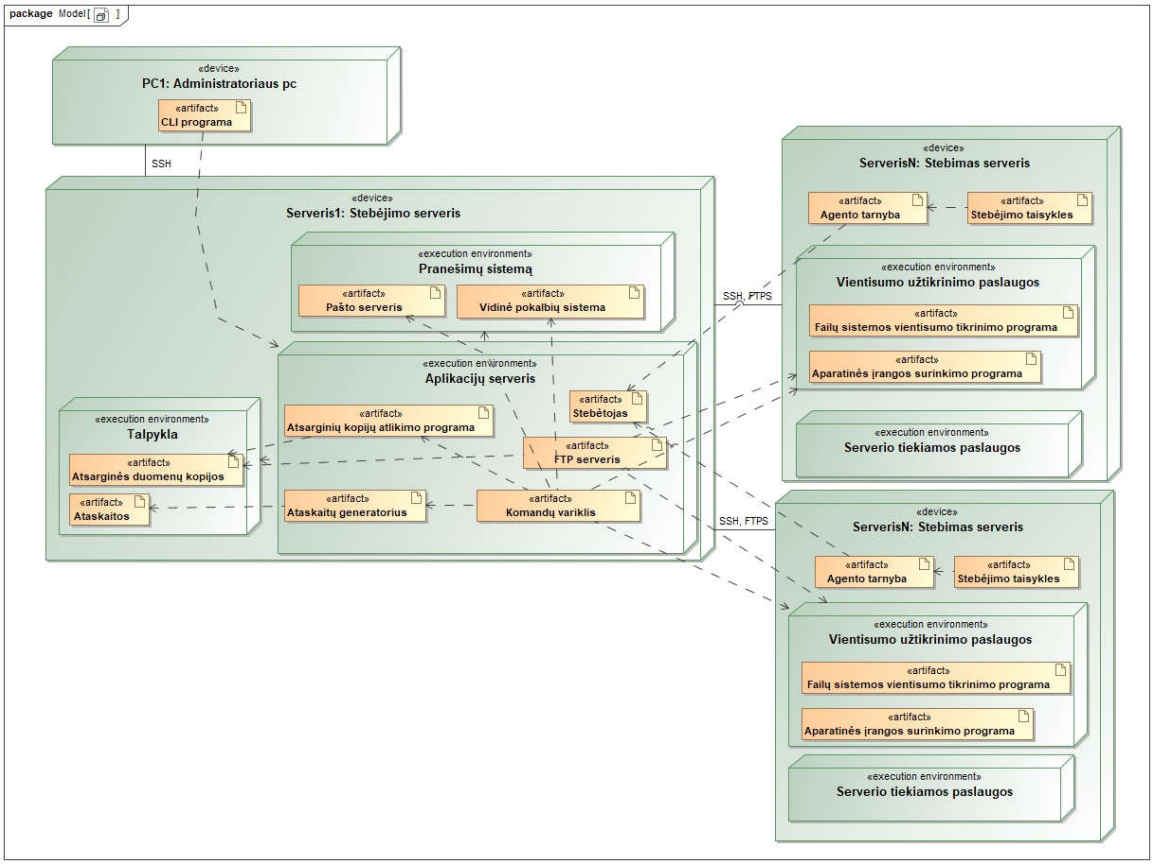
Taip pat sistema aptinka jei yra prijungiama ar atjungiama papildoma įrangą ir sukuria skirtuminę ataskaita tarp standarto ir kai buvo prijungta ar atjungta įranga, pavyzdžiui prijunktas USB raktas.



2.11 pav. Vientisumo duomenų tikrinimo ir atstatymo sekų diagrama

2.11 pav. Vientisumo duomenų tikrinimo ir atstatymo sekų diagramoje yra pavaizduota, kaip stebėjimo serveris komunikuoja su stebimu serveriu, stebimas serveris gali inicijuoti prašymą atlikti sistemos vientisumo tikrinimą, pvz. prijunktas USB raktas arba praėjo laiko intervalas nuo paskutinio tikrinimo. Sužadinus tikrinimą stebėjimo serveris prisijungia prie stebimo serverio per SSH protokolą pasitelkiant komandų variklių, ir patikrintiną ar visos reikalingos talpyklos yra primontuotos. Jei talpyklos primontuotos yra kviečiamos programos atlikti vientisumo tikrinimui, tai yra surenkami duomenys apie aparatinę įranga ir surankami stebimų failų vientisumo duomenys, šie duomenys yra atsisiunčiami į stebėjimo serverį, kad paskui būtų galimą sugeneruoti bendra ataskaitą apie sistemos vientisumą. Tai pat čia yra atliekamas sistemos vientisumo atstatymas, jei reikalingas ir yra sukuriama

ataskaitą apie failų atstatymą, joje yra informacija, kokie failai buvo atstatyti ir ar buvo klaidų atstatymo metu. Ši informacija yra pridedama prie galutinės ataskaitos ir išsiunčiama administratoriams patikrinimui.



2.12 pav. Sistemos diegimo diagrama

Diegimo diagramoje 2.12 pav. pavaizduotos sistemos dalys. CLI programa, atsarginių kopijų atlikimo programa, ataskaitų generatorius, aparatinės įrangos surinkimo programa, stebėtojas yra dalys, kurias reikia papildomai įgyvendint, kad sistema galėtų teisingai veikti. Komandų varikliui taip pat reikia suprogramuoti komandas, kurias galėtų vykdyti sistema, kad būtų galima atlikti visas numatytas funkcijas.

2.2. Išvados

- Pasiūlytas kompiuterinės sistemos vientisumo užtikrinimo modelis skirtas stebėti aparatinę įrangą bei programiniai įrangai ir užtikrinti, kad sistemoje neatsitiktų nenumatytų pakeitimų;
- Sukurtas modelis leidžia administratoriams nesunkiai atlikti aparatinės bei programinės įrangos naujino darbus ir naujas vientisumo reikšmes patalpinti monitoringo serveryje;
- Pasiūlyta sistema netik perspėja administratorius dėl įvykusiu sistemoje neautorizuotų pakeitimų, bet ir bando atstatyti sistemos vientisumą;
- Sistemą turi didelį pritaikomumo laipsnį. Administratoriai gali pritaikyti jų reikiamiems panaudos atvejams.

3. Kompiuterinės sistemos vientisumo užtikrinimo prototipas ir tyrimas

3.1. Kompiuterinės sistemos vientisumo užtikrinimo prototipas

Pagrindinė sistemos dalį sudaro komandų variklis, naudojant šį įranki yra kviečiami scenarijai, kurie ir atlieką sistemos vientisumo užtikrinimą. Komandų variklis yra įgyvendinamas naudojant *Ansible* [23] automatizacijos, konfigūracijos ir taikomųjų programų diegimo įrankis.

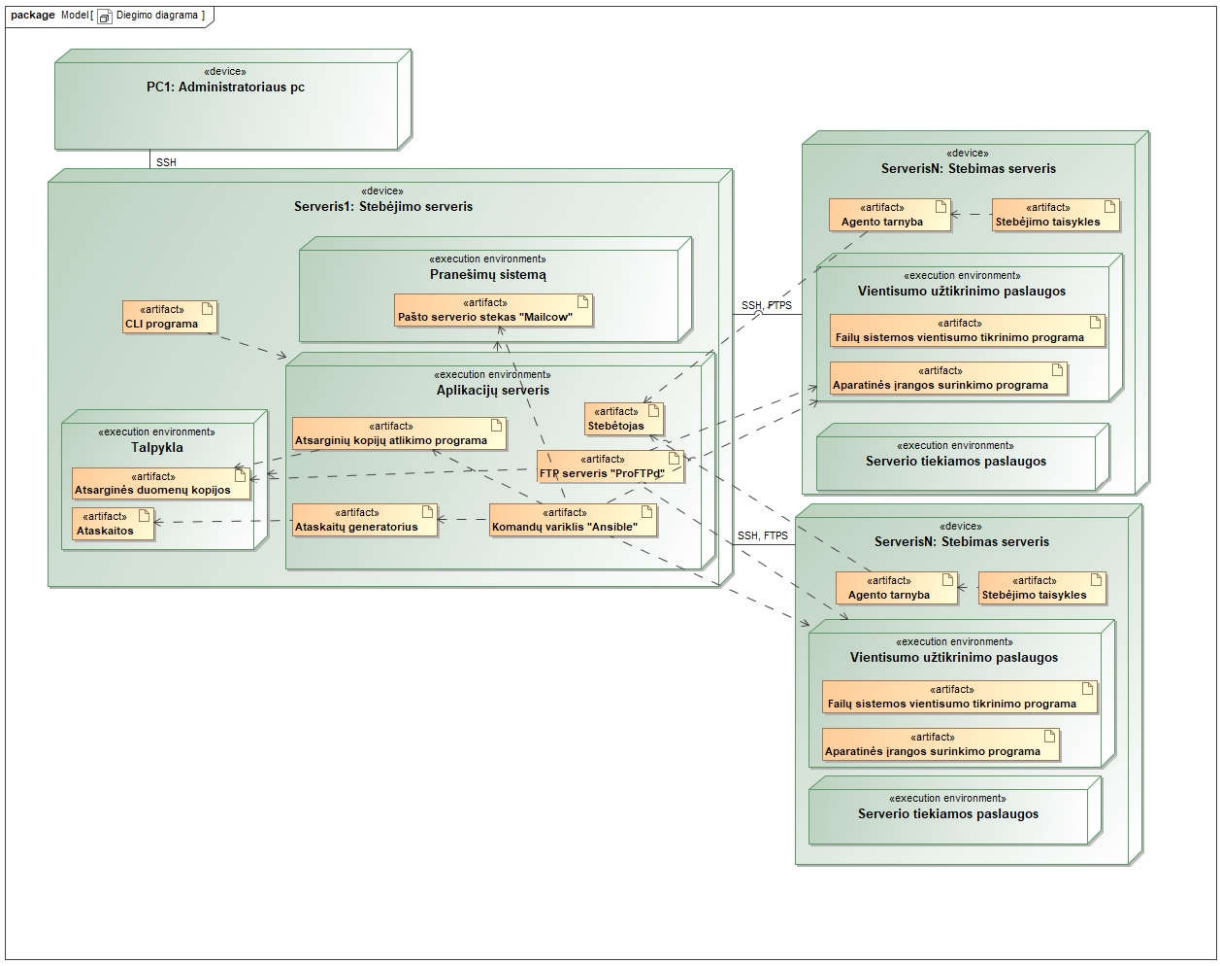
Sistema susideda iš 9 *Ansible* scenarijų, kurie ir atlieka pagrindinius sistemos veiksmus.
Taip pat yra sukurti 2 papildomi *bash* scenarijai, kurių pagalba yra atliekamos nurodytų failų atsarginės kopijos ir jų atstatymas įvykus vientisumo pažeidimui. Failų atstatymas veikia patikrinus *Tripwire* ataskaitą. Jei yra randami įrašai apie tuos failus, kurie yra pažymėti, kad yra reikalingos atsargines kopijas, ir jei jie buvo ištrinti ar pakeisti, yra kviečiamas *bash* scenarijus, kuris bando atstatyti failus iš rezervinių kopijų ir sukuria papildoma ataskaita kurioje nurodyta, kurios failus pavyko atstatyti ir kurių nepavyko.

Taip pat yra dar sukurtas 1 papildomas *bash* scenarijus, kuris surenka duomenis apie dabartinę aparatinės įrangos būseną ir patalpina juos į atskirus failus pagal kategorijas.

Suprogramuota stebėtojo programa, kuri laukia iškvietimų iš stebimų serverių, kad reikia atlikti sistemos vientisumo tikrinimą. Ši programa stebi specialų užrakto failą, kuris yra sukuriamas kai stebimas serveris bando prisijunkti prie stebėjimo sistemos per SSH protokolą. Kiekvienas stebimas serveris turi savo užrakto failą tokiu būdų galima atskirti kuriam serveriui reikia atlikti tikrinimą

Be to yra suprogramuotas papildomas scenarijus, kurio pagalba yra suagreguojami duomenys tai yra *Tripwire* vientisumo pažeidimų ataskaita, atstatytų failų ataskaita, aparatinės įrangos pasikeitimų ataskaitą bei klaidos jei jos atsirado, kokiame nors vienam iš scenarijų ir jie yra išsiunčiami administratoriui patikrinti.

Visa prototipo schema yra pavaizduota paveiksle 3.1 pav.

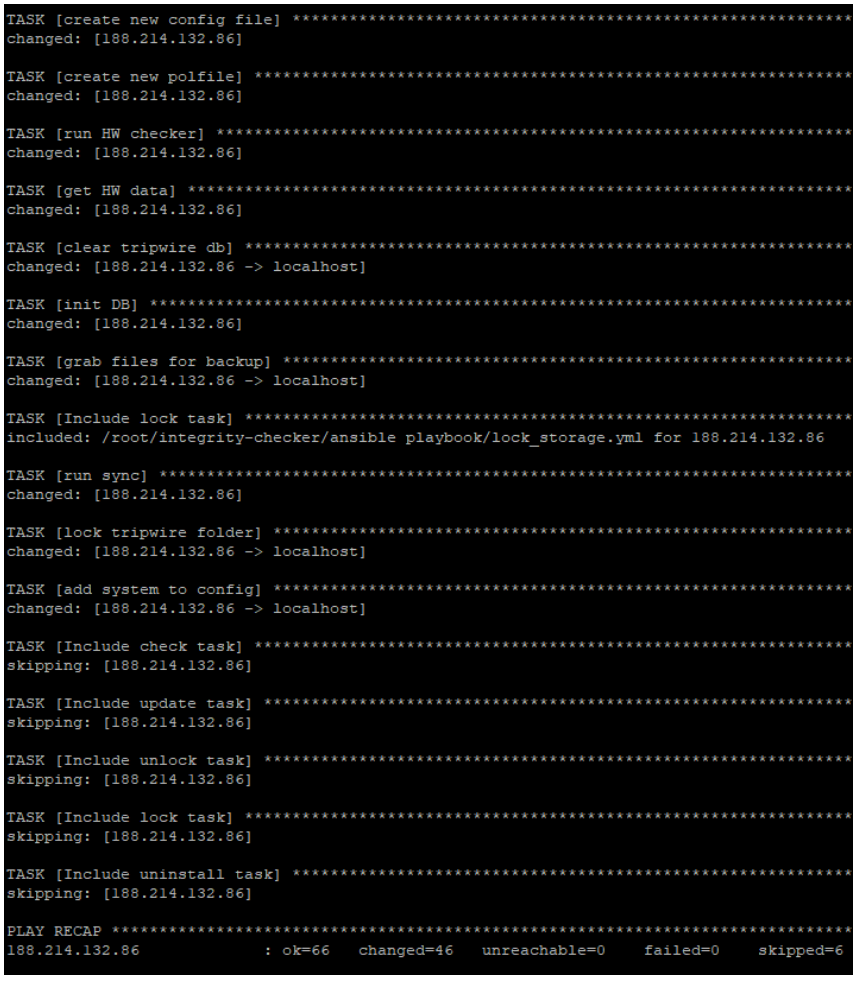


3.1 pav. Prototipo schema

Visos sistemos funkcijos:

- Vartotojų, raktų sukūrimas stebėjimo serveryje. Skirta, kad būtų galima atskirti stebimus serverius viena nuo kito ir tikrinimo iškvietimui iš stebimo serverio.
- Sukurti talpyklas stebėjimo serveryje ir jas primontuoti stebimame serveryje; Kiekvienas stebimas serveris turi savo talpyklą, kuri yra skirta saugoti vientisumo duomenimis, ji yra valdoma iš stebėjimo serverio.
- Atrakinti ir užrakinti talpyklas. Talpyklos galima įjunkti arba išjunkti rašymo režimą, rašymo režimas yra įjungimas tik duomenų atnaujinimo metu. Darbinė būsena yra be rašymo teisiu taip apsaugant vientisumo duomenis nuo nesankcionuotų pakeitimų.
- *Tripwire* įdiegimas į stebimą serverį. Programa skirta failų vientisumo duomenų surinkimui.
- Atlikti vientisumo duomenų tikrinimą. Atlieka duomenų vientisumo tikrinimą, tikrinamą ar nebuvo pakeisti duomenys lyginant su duomenų baze esančia talpykloje.
- *Udev* taisyklių įdiegimas. *Udev* taisykles skirtos iškviesti vientisumo tikrinimą jei yra prijunktas arba atjunktas aparatinis įrenginys.
- Aparatinės įrangos informacijos surinkimas. Surenkama informacija apie dabartinę aparatinės įrangos būseną, šie duomenys yra patalpinami į failus pagal kategorijas.
- Failų atsarginių kopijų sukūrimas. Sistema nukopijuoja pasirinktus failus ir juos patalpina į talpyklą kaip atsarginę kopiją.

– Failų atstatymas iš atsarginių kopijų. Failai kurie buvo pakeisti ar panaikinti nesankcionuotai, sistema juo gali automatiškai atstatyti iš atsarginės kopijos.
– Sugeneruoti ir išsiųsti ataskaitą administratoriams. yra sugeneruojama ataskaitą iš šių komponentų: *Tripwire* vientisumo ataskaitos, Aparatinės įrangos pasikeitimų ataskaitos, atsarginių kopijų atstatymo ataskaitos ir *Ansible* vykdymo išvestis. Ši ataskaita yra siunčiama per elektroninį paštą administratoriams patikrinimui.
– Vientisumo duomenų atnaujinimas. Sistema turi funkcija pagal *Tripwire* ataskaita leidžia, atnaujinti pakeistų failų vientisumo duomenų baze, aparatinės įrangos informacija bei iš naujo atlikti atsargines failų kopijas.
– Šalintojas. Yra funkcija kurios pagalba galima ištrinti vientisumo užtikrinimo sistema, ji yra skirta jei reikia pakeisti sistemos tinklo nustatymus ar serverio pavadinimą arba jei nebereikia stebėti serverio.
– Stebėtojas. Yra sistemoje funkcija, kuri leidžia iškviečia stebėtoja kiekvienai stebimai sistemai. Šis stebėtojas laukia žinutės iš stebimo serverio ir ją gavęs yra atliekamas tos sistemos vientisumo tikrinimas.

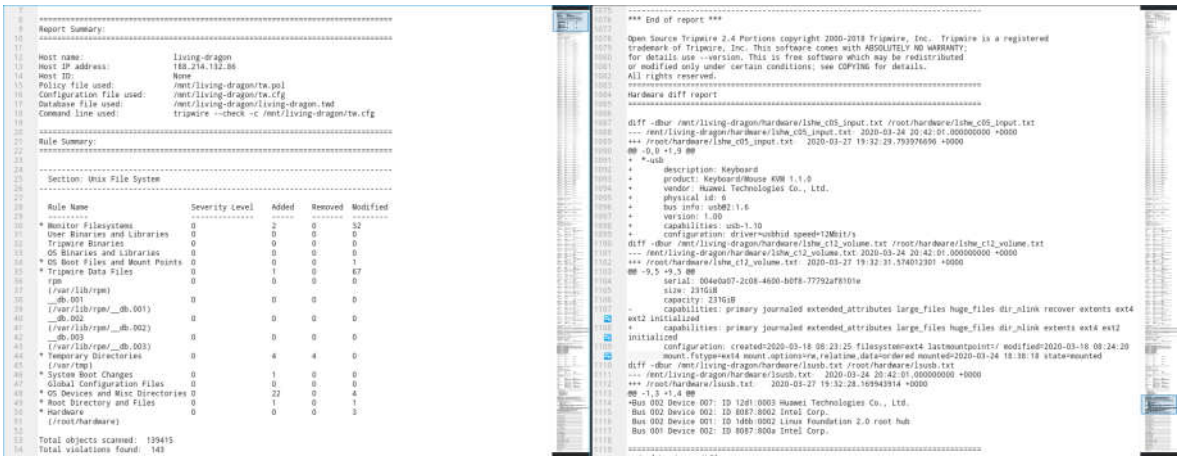


3.2 pav. Stebimos Sistemos paruošimas

3.2 pav. parodyta iškarpa iš sistemos diegimo scenarijaus, šis scenarijus prisijungia prie stebimo serverio ir sudiegia reikiamas programas, su kuriomis toliau bus tikrinamas sistemos vientisumas, sukuria reikalingus vartotojus stebėjimo serveryje, atidaro ugniasienėje prievadus. Taip pat šis

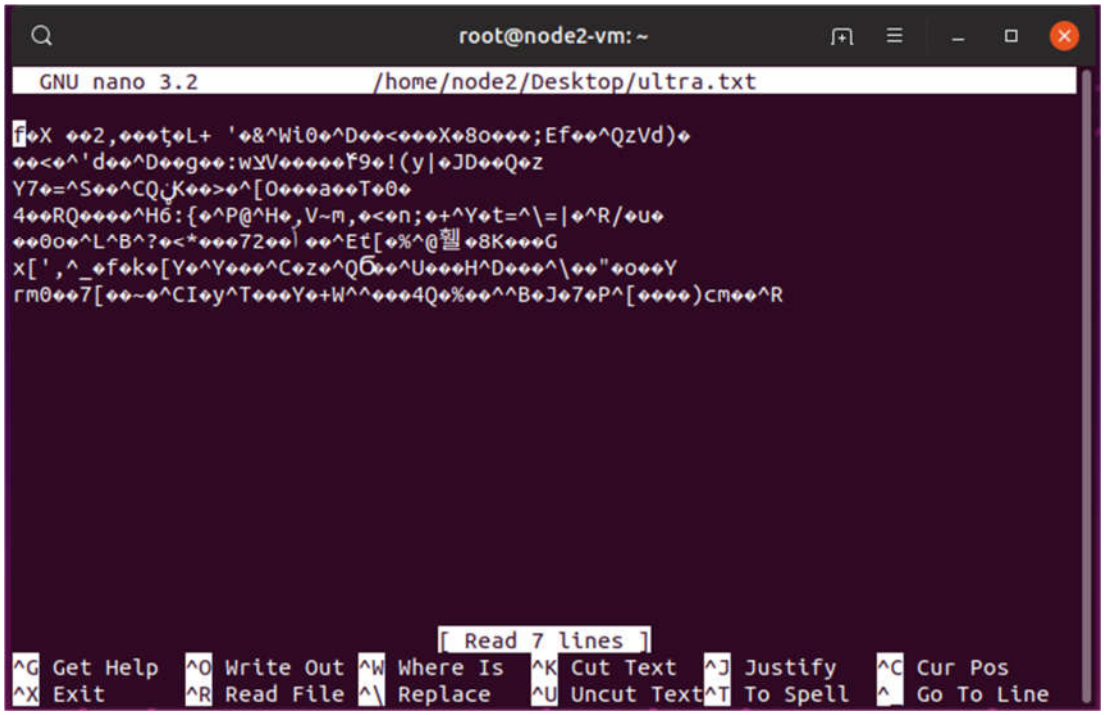
scenarijus pasinaudoja papildomais scenarijais ir paprogramėmis, kurios atlieka ir pradinį vientisumo duomenų surinkimą, aparatinės įrangos duomenų surinkimą bei atsarginių kopijų atlikimą.

Sistemos failų struktūrą susideda iš *Tripwire* konfigūracinių failų, politikos failo, šifruotų raktų, vientisumo duomenų bazės. Taip pat yra laikomi duomenys apie aparatinę įrangą bei failų atsarginės kopijos. Kurių failų atsargines kopijas reikia turėti galima nurodyti politikos faile tai tik saugant būtiniausius duomenis.

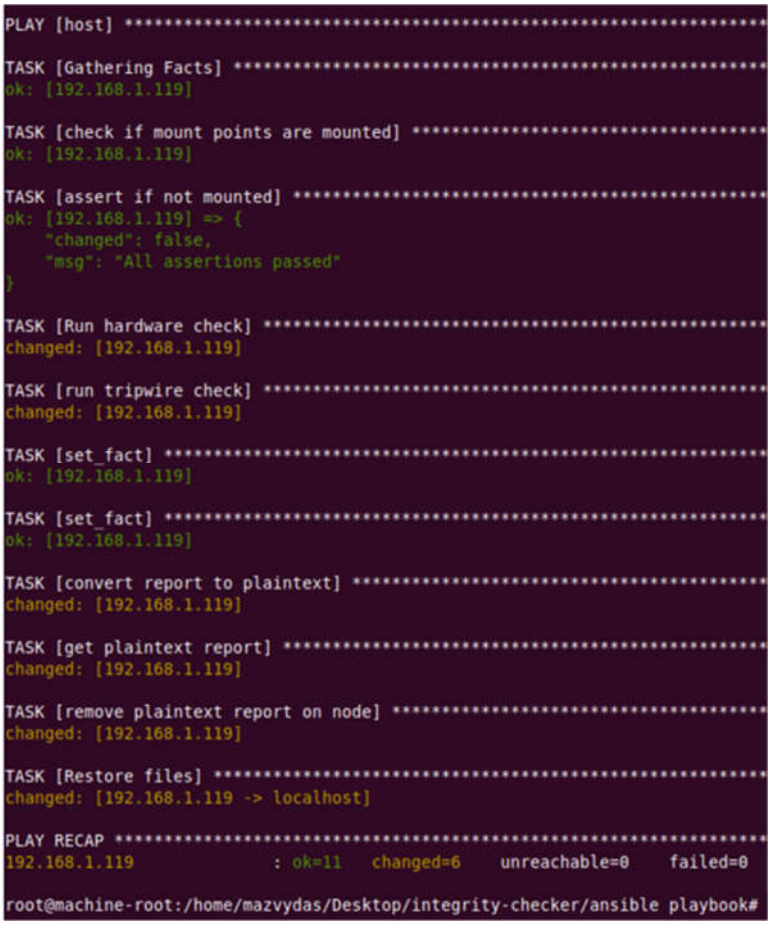


3.3 pav. Ataskaitos iškarpa

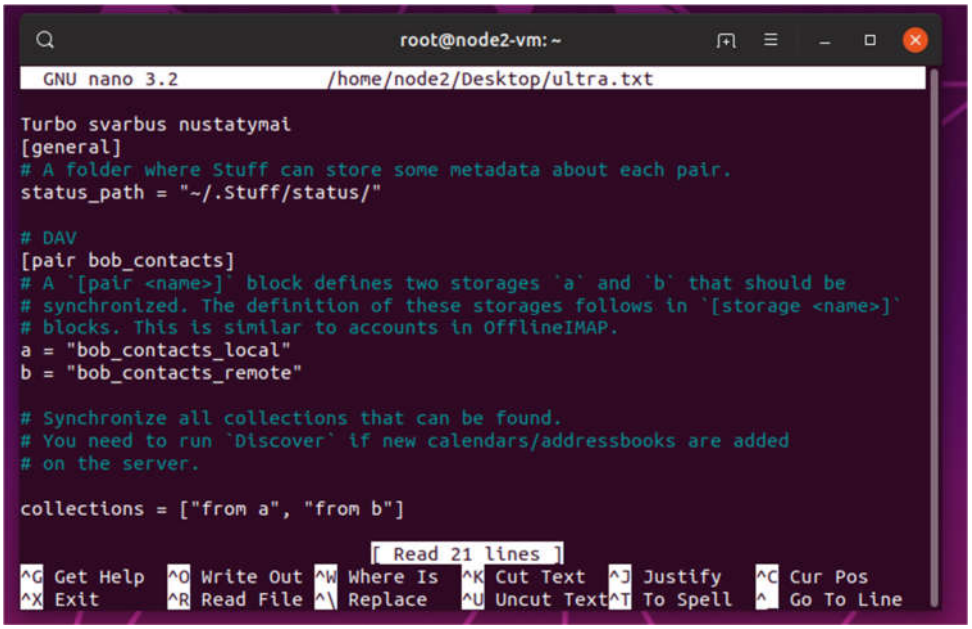
3.3 pav. Pavaizduota iškarpa iš sugeneruotos ataskaitos. Šitoje ataskaitoje galima matyti kiek ir kokie vientisumo pažeidimai buvo rasti. Taip pat ataskaitoje galima matyti, kokie pasikeitimai buvo aptikti aparatinėje įrangoje bei vientisumo tikrinimo komandos išvestis, iš kurio galima nustatyti ar pilnai pavyko atlikti tikrinimą ir ar nebuvo klaidų tikrinimo procese. Ši ataskaita yra siunčiama administratoriams patikrinimui ar visi nustatyti pakeitimai yra sankcionuoti. Jei pažeidimu nebuvo nustatyta tuščia ataskaita nėra siunčiama.



3.4 pav. Sugadintas stebimas failas



3.5 pav. Atliekamas tikrinimas su automatiniu atstatymu



3.6 pav. Atstatytas failas

3.4 pav. , 3.5 pav. , 3.6 pav. matosi sistemos galimybė atstatyti failus, kurių yra daromas rezervinės kopijos. Jei yra sugadintas failas ir buvo sukurtos rezervinės kopijos kuriant ar atnaujinant vientisumo duomenų bazę, tai atlikus tikrinimo scenarijų su automatinių failų atstatymu šie failai bus atstatytas į tokia būsenas, kurios buvo kuriant ar atnaujinant vientisumo duomenų bazę, taip pat yra generuojama papildoma ataskaita jei buvo atliktas automatinis atstatymas ir kokie failai buvo atstatyti ir jei iškilo klaidos, kurie failai nebuvo atstatyti.

3.2. Tyrime naudojama aparatinė ir programinė įranga

Remiantis sukurtu kompiuterinės sistemos vientisumo užtikrinimo prototipu buvo atliktas tyrimas

Tyrimo metu buvo naudojama ši aparatinė įranga:

- 7 Serveriai „Huawei XH620 V3“
- 40 Virtualių serverių „QEMU/KVM“
- Stacionarus kompiuteris „Z270 GAMING M7 (MS-7A57)“

Plačiau apie, kokia naudojama aparatinę įranga yra pateikta 3.1 lentelėje.

Tyrimo metu naudota programinė įranga:

- *Visual studio code 1.28.2*
- *Virtual machine manager 2.2.1*
- *Bash*
- *Time*
- *Nmon*

Virtualūs serveriai buvo patalpinti į 5 „Huawei XH620 V3“ serverius, kad kiekviename fiziniame serveryje būtų nedaugiau nei 8 virtualūs serveriai.

3.1 lentelė Tyrime naudota aparatinė įranga

Stebėjimo serveris Huawei XH620 V3	
Pagrindinė plokštė	BC21HGSB
Procesorius	Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz
Operatyvioji atmintis	Hynix DIMM DRAM Synchronous Registered (Buffered) 2400 MHz 32GB, DDR4
Kietasis diskas	2 Samsung SSD 860 EVO 250GB RAID 1 LSI SAS3008
Grafinė plokštė	Hi1710 [iBMC Intelligent Management system chip w/VGA support]
Tinklo plokštė	Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Operacinė sistema	Ubuntu 19.10 64-bit

6 stebimi serveris Huawei XH620 V3	
Pagrindinė plokštė	BC21HGSB
Procesorius	Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz
Operatyvioji atmintis	Hynix DIMM DRAM Synchronous Registered (Buffered) 2400 MHz 32GB, DDR4
Kietasis diskas	2 Samsung SSD 860 EVO 250GB RAID 1 LSI SAS3008
Grafinė plokštė	Hi1710 [iBMC Intelligent Management system chip w/VGA support]
Tinklo plokštė	Intel Corporation 82599ES 10-Gigabit SFI/SFP+
Operacinė sistema	Ubuntu 18.04.2 LTS 64-bit

40 Virtualūs serveriai	
Pagrindinė plokštė	Standard PC (Q35 + ICH9, 2009)
Procesorius	1 vCPU Intel Core Processor (Skylake, IBRS)
Operatyvioji atmintis	2GB
Kietasis diskas	Virtio block device 16GB SSD
Grafinė plokštė	QXL paravirtual graphic card
Tinklo plokštė	Virtio network device
Operacinė sistema	Ubuntu 18.04.4 LTS

Stacionarus kompiuteris „Z270 GAMING M7 (MS-7A57)“	
Pagrindinė plokštė	Z270 GAMING M7 (MS-7A57)
Procesorius	Intel (R) Core (TM) i7-7700K CPU @ 4.20GHz
Operatyvioji atmintis	Kingston KHX2400C15/16G DIMM DDR4 Synchronous Unbuffered (Unregistered) 2400 MHz (0,4 ns) 16 GB, DDR4
Kietasis diskas	Samsung SSD 960 EVO 500GB Western Digital Black WDC WD2003FZEX-00SRLA0 2TB
Grafinė plokštė	AMD Radeon RX 580 8GB
Tinklo plokštė	Qualcomm Atheros Killer E2500 Gigabit

Operacinė sistema	Manjaro Linux

Stebimų serverių operacinė sistema yra įdiegta naudojantis paslaugų tiekėjo įrankiu. Papildomos paslaugos nebuvo įdiegtos apart tų, kurias įdiegia paslaugų tiekėjas bei tos, kurios yra reikalingos kompiuterinės sistemos vientisumo užtikrinimo sistemai veikti ir tyrimui atlikti.

3.3. Kompiuterinės sistemos vientisumo užtikrinimo sistemos kokybinis tyrimas

Atliekant kuriamos sistemos prototipo tyrimą buvo nustatyta, kad sistema atlieka numatytas funkcijas, tai yra perspėja administratorius įvykus vientisumo pažeidimui, ir jei pažeistas failas nustatytas politikoje, kaip failas, kurio reikia turėti atsarginę kopiją, jis yra atstatomas. Jeigu stebimas serveris yra paleidžiamas iš naujo arba prie serverio yra prijungiamas USB įrenginys, į monitoringo serverį yra išsiunčiama žinutė, kad reikia atlikti vientisumo tikrinimą.

Tačiau po paleidimo iš naujo buvo nustatyta, 1 iš 5 kartų talpykla neprisimontavo prie stebimo serverio teisingai ir tikrinimas sustojo su klaida, ši klaida buvo išsiusta administratoriui kaip ataskaita, kurioje ir buvo galima matyti, kad vientisumo tikrinimas nepavyko ir reikalinga administratoriaus intervencija.

Taip pat buvo nustatyta jei failas, kuris politikoje nustatytas turintis atsarginė kopija, buvo pažeistas ir pakeistas šio failo atributas į „nekintamas“, sistema neatstatė šio failo iš atsarginės kopijos ir administratoriui nebuvo pranešta, kad failas nebuvo atstatytas, tačiau ataskaitoje buvo nurodyta, kad failo vientisumas pažeistas.

3.4. Kompiuterinės sistemos vientisumo užtikrinimo sistemos tikrinimo trukmės ir apkrovos tyrimas

3.7 pav. Pirmojo eksperimento schema

Pirmojo eksperimento metu yra tiriama vientisumo skaičiavimų trukmė ir stebimo serverio apkrova. Per kiek laiko sistema išsiunčia pranešimą administratoriui apie nustatytus vientisumo pažeidimus. Eksperimento schema pavaizduota 3.7 pav. Tyrimas buvo atliktas naudojant tris skirtingas politikas pateiktas lentelėje 3.2 **lentelė** Tyrimui naudojamos politikos. Politikose yra nurodyta, kokius aplankalus sistemai reikia stebėti dėl pokyčių. Pirmojoje politikoje pasirinka stebėti tik aparatinę įrangą, antrojoje politikoje, kartu su aparatinę įranga, pasirinka stebėti programas ir jų konfigūraciją, trečioje politikoje yra stebima visa sistema.

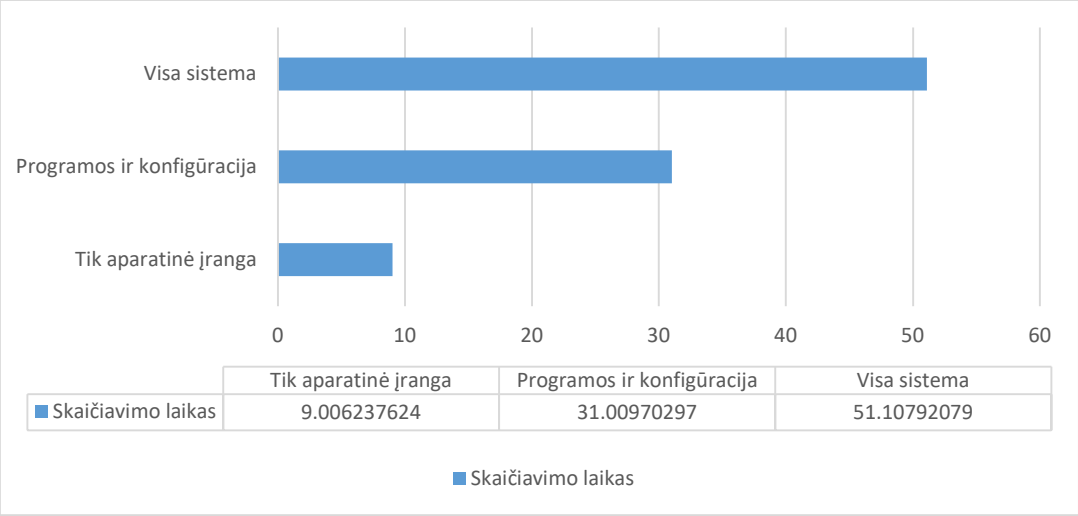
3.2 lentelė Tyrimui naudojamos politikos

Politika	**Stebimi aplankalai**
Tik aparatinė įranga	/dev/

	/root/hardware/
Programos ir konfigūracija	/etc/ /boot/ /dev/ /bin/ /lib/ /sbin/ /usr/bin/ /usr/lib/ /usr/sbin/ /usr/local/ /root/ /var/run/
Visa sistema	/bin/ /boot/ /dev/ /etc/ /lib/ /lost+found/ /mnt/ /opt/ /root/ /sbin/ /usr/ /var/

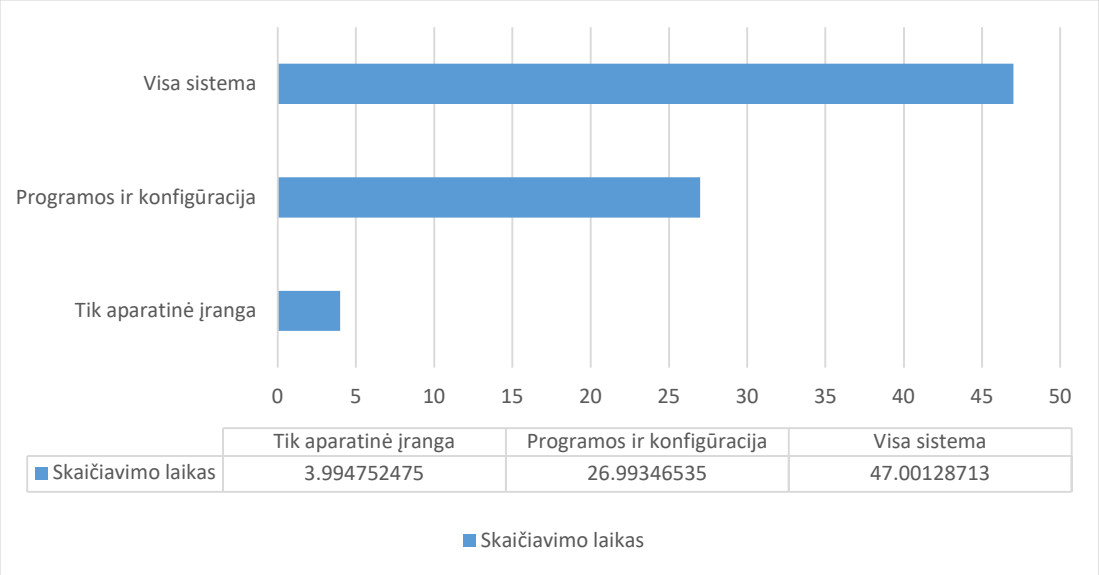
Stebimas serveris kreipiasi į stebėjimo serverį, kad pradėti vientisumo tikrinimą, tačiau tyrimo metu tikrinimo funkcija yra tiesiogiai paleidžiama iš stebėjimo serverio. Vientisumo tikrinimas buvo atliktas 100 kartų su kiekviena politika.

Tyrimo rezultatai yra rašomi į tekstinius failus ir vėliau apdorojami kompiuteriu.



3.8 pav. Vientisumo tikrinimo trukmė nuo tikrinimo pradžios iki pranešimo pristatymo administratoriams

Diagramoje 3.8 pav. pavaizduota vidutinė vientisumo tikrinimo trukmė nuo tikrinimo iškvietimo iki pranešimo pristatymo administratoriams yra nuo 9 sekundžių esant tik aparatinės įrangos tikrinimo politikai ir gali užtrukti iki 51 sekundžių kai yra tikrinama visa sistema. Taip pat galime matyti, kad tikrinant daugiau failų, ilgėja ir laikas, kuris reikalingas tikrinimui atlikti.



3.9 pav. Aparatinės įrangos ir vientisumo duomenų surinkimo trukmė

Diagramoje 3.9 pav. pavaizduotas reikalingas laikas aparatinė įrangos ir vientisumo duomenis surinkti iš stebimo serverio. Iš grafikų matome kad:

- Didėjant tikrinimo apimčiai, ilgėja ir tikrinimo trukmė;
- Be aparatinės įrangos ir vientisumo duomenų surinkimo yra reikalingas papildomas laiko tarpas valdymo išlaidoms, nuo 4,02 sek. iki 5,01 sek.;

Lentelėje 3.3 - 3.5 yra pateikta stebimo serverio apkrovos tikrinimo metu:

3.3 lentelė Stebimos sistemos procesoriaus apkrova kai naudojama politika „Tik aparatinė įranga“

Tik aparatinė įranga	**Vartotojo programos**	**OS branduolys**	**Įvesties/išvesties laukimas**	**Laisvas pajėgumas**
Procesoriaus užimtumas	12.85 %	0.93 %	0.43 %	85.79 %

3.4 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Tik aparatinė įranga“

Tik aparatinė įranga	
Disko užimtumas	2.13 %
Disko skaitymas	65.92 KB/s
Disko rašymas	154.86 KB/s

3.5 lentelė Stebimos sistemos kietų tinklo apkrova, kai naudojama politika „Tik aparatinė įranga“

Tik aparatinė įranga	**Gavimas**	**Siuntimas**
Tinklo apkrova	28.47 KB/s	27.58 KB/s

Iš lentelių matome, kad:

- Yra išnaudojamas 1 sistemos procesoriaus branduolys, likę procesoriaus branduoliai gali atlikti kitus paslaugos tiekimo darbus;
- Vidutinė disko apkrovos minimalios tik 2 %;
- Tinklo apkrovos taip pat minimalios tik 28 Kilobaitai per sekundę.

3.6 lentelė Stebimos sistemos procesoriaus apkrova, kai naudojama politika „Programos ir konfigūracija“

Programos ir konfigūracija	**Vartotojo programos**	**OS branduolys**	**Įvesties/išvesties laukimas**	**Laisvas pajėgumas**
Procesoriaus užimtumas	9.27 %	0.89 %	3.11 %	86.73 %

3.7 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Programos ir konfigūracija“

Programos ir konfigūracija	
Disko užimtumas	43.72 %
Disko skaitymas	65115.16 KB/s
Disko rašymas	226.30 KB/s

3.8 lentelė Stebimos sistemos kietų tinklo apkrova, kai naudojama politika „Programos ir konfigūracija“

Programos ir konfigūracija	**Gavimas**	**Siuntimas**
Tinklo apkrova	4.86 KB/s	5.91 KB/s

Lentelėse 3.6 - 3.8 matoma kad:

- Sistema pilnai išnaudoja 1 procesoriaus branduolį kaip ir tik aparatinės įrangos tikrinimo metu;
- Vidutinė disko apkrova yra padidėjusi sistema išnaudoja apie 43,72 %, užimtumas daugiausia susideda iš disko skaitymo, skaitymo greitis yra apie 65,1 megabaitas per sekundę;
- Vidutinė tinklo apkrovos išlieka minimali.

3.9 lentelė Stebimos sistemos procesoriaus apkrova, kai naudojama politika „Visa sistema“

Visa sistema	**Vartotojo programos**	**OS branduolys**	**Įvesties/išvesties laukimas**	**Laisvas pajėgumas**
Procesoriaus užimtumas	8.43 %	1.14 %	3.58 %	86.85 %

3.10 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Visa sistema“

Visa sistema	
Disko užimtumas	44.95 %
Disko skaitymas	59881.83 KB/s
Disko rašymas KB/s	588.09 KB/s

3.11 lentelė Stebimos sistemos kietų diskų apkrova, kai naudojama politika „Visa sistema“

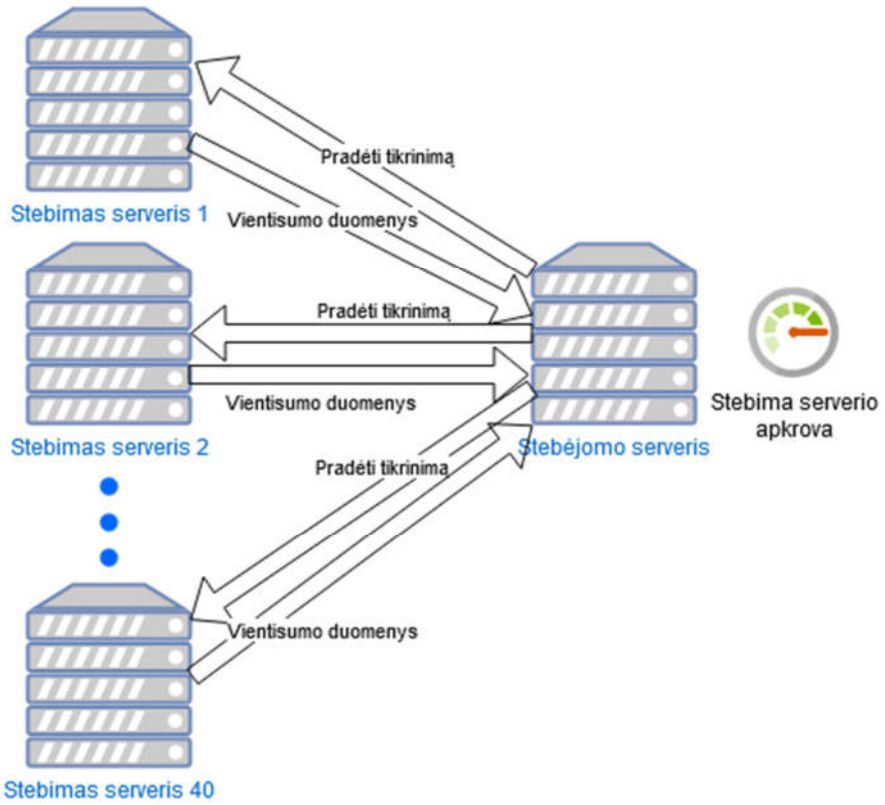
Visa sistema	**Gavimas**	**Siuntimas**
Tinklo apkrova	6.28 KB/s	4.01 KB/s

Kai yra tikrinama visa sistema iš lentelių 3.9 - 3.11 galime matyti, kad:

- Kad sistema naudoja 1 procesoriaus branduolį, tai rodo kad jei būtų galingesnis procesorius vienos gijos darbui tikrinimas būtų atliekamas greičiau. Tačiau taip pat yra paliekami resursai kitiems sistemos darbams.
- Diskų apkrova panaši kaip ir „Programos ir konfigūracija“ atveju apie 45% ir kaip „Programos ir konfigūracija“ disko skaitymas yra didžioji apkrovos dalis.
- Tinklo apkrovos yra minimalios.

Iš gautų rezultatu galime matyti, kad stebima sistema nėra perkraunama vien sistemos tikrinimu, tačiau jei sistemoje tik vienas procesorius tai gali sukelti problemų tikrinimo metu, gali sulėtėti tikrinimo ar tiekiamų paslaugų darbas. Kai yra tikrinama failų sistema disko apkrova išauga, tačiau ji nedaro įtakos sistemos darbui. Tikrinant tik aparatinę įranga sistemos tikrinimas užtrunka apie 4 sekundes, o administratorius perspėja per 9 sekundes, jei yra tikrinama visa sistema tai užtrunka apie 1 minutę.

3.5. Kompiuterinės sistemos vientisumo užtikrinimo sistemos lygiagrečių tikrinimų tyrimas

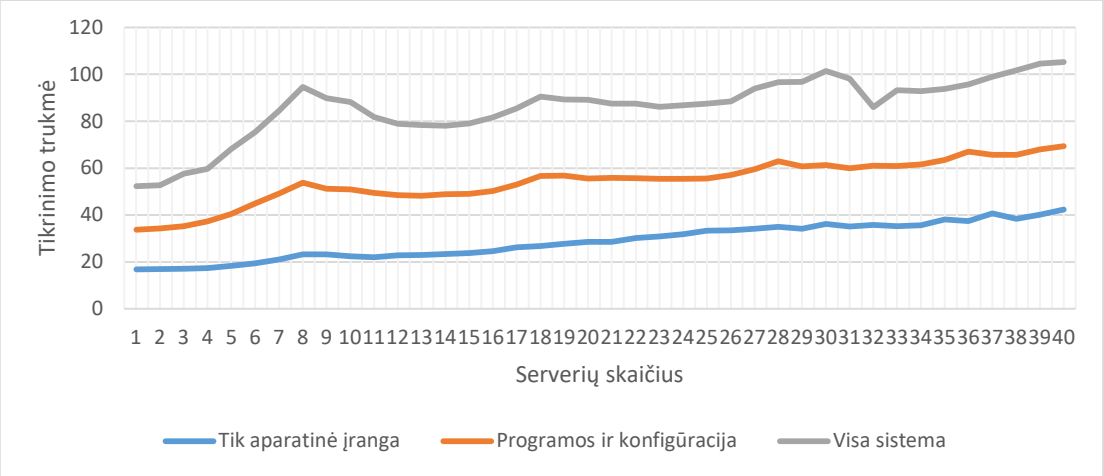


3.10 pav. Antrojo eksperimento schema

Antrojo eksperimento metu buvo tiriama kompiuterinės sistemos vientisumo užtikrinimo sistemos lygiagretus kelių serverių tikrinimas. Šiam tyrimui buvo sukurt 40 virtualių mašinų, kurios buvo patalpintos 5 fizinėse mašinose. Tyrimo metu buvo paleidžiamas tikrinimas pradedant viena mašina, pasibaigus tikrinimui yra 5 minučių tarpas, kad normalizuotųsi sistemos pakrovos rodmenys. Po to yra paleidžiamas tikrinimas su papildoma pridėta mašina, taip yra tęsiama, kol lygiagrečiai yra tikrinami visos 40 virtualios mašinos. Eksperimento schema yra pavaizduota 3.10 pav. Eksperimentas

buvo atliktas su trejomis politikomis (Tik aparatinės įrangos tikrinimas, programos ir konfigūracijos, visa sistema).

Gauti rezultatai buvo rašomi į tekstinius failus ir vėliau apdoroti kompiuteriu.

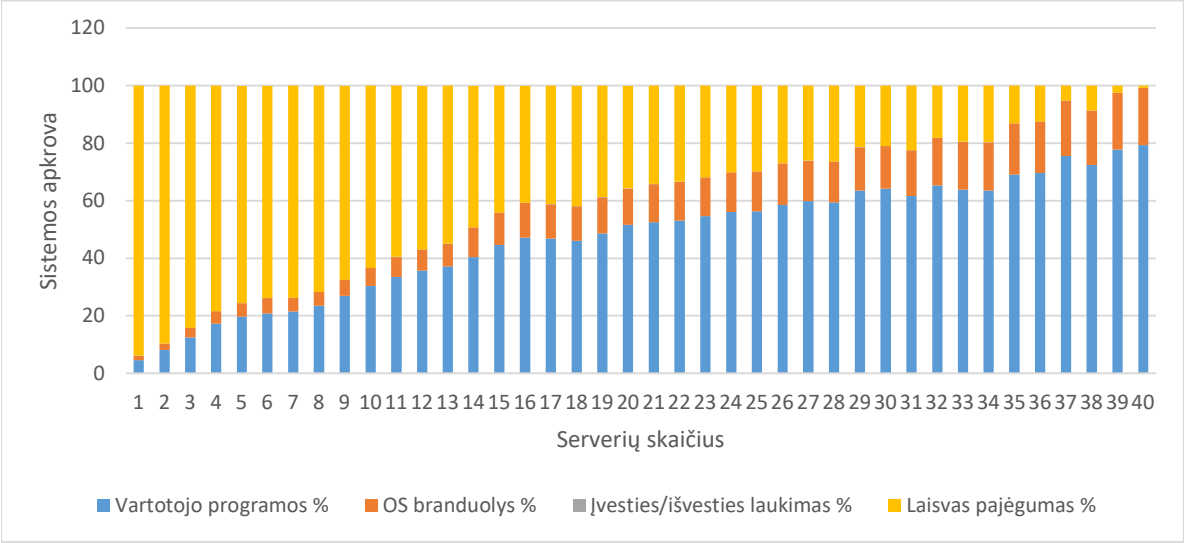


3.11 pav. Vidutinės tikrinimo įvykdymo trukmė esant skirtingoms politikoms

Tyrimo metu buvo išmatuota lygiagrečiai atliktų serverių tikrinimų trukmės, iš gautų rezultatų 3.11 pav. matome, kad:

- Kuo daugiau tikrinimų vyksta lygiagrečiau tuo ilgiau užtrunka sistemos tikrinimas;
- Sistema gali atlaikyti bent 40 tikrinimų vienu metu;
- 40 serverių lygiagretus tikrinimas 2 kartus prailgina tikrinimo trukmė lyginant su vieno serverio tikrinimu.

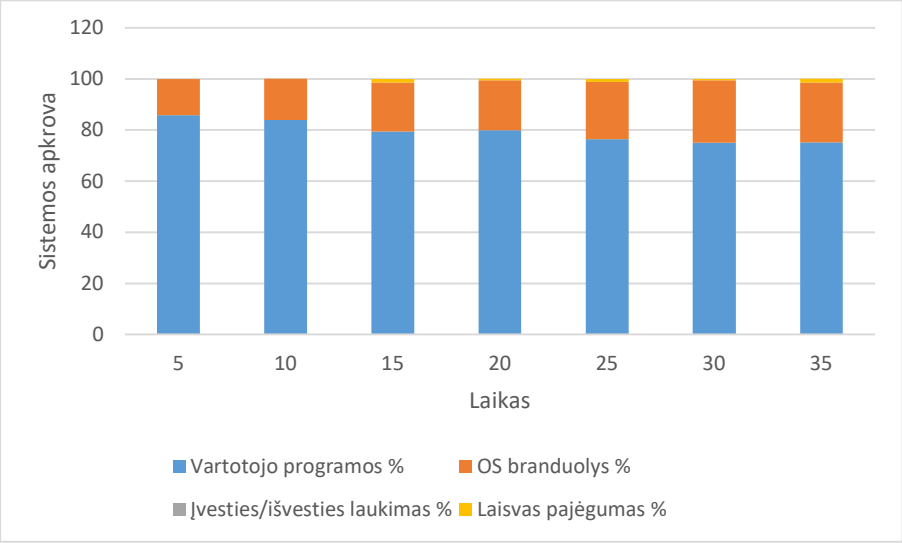
Stebėjimo serverio procesoriaus apkrovos kiekvienos politikos atveju.



3.12 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Tik aparatinės įrangos“ politiką

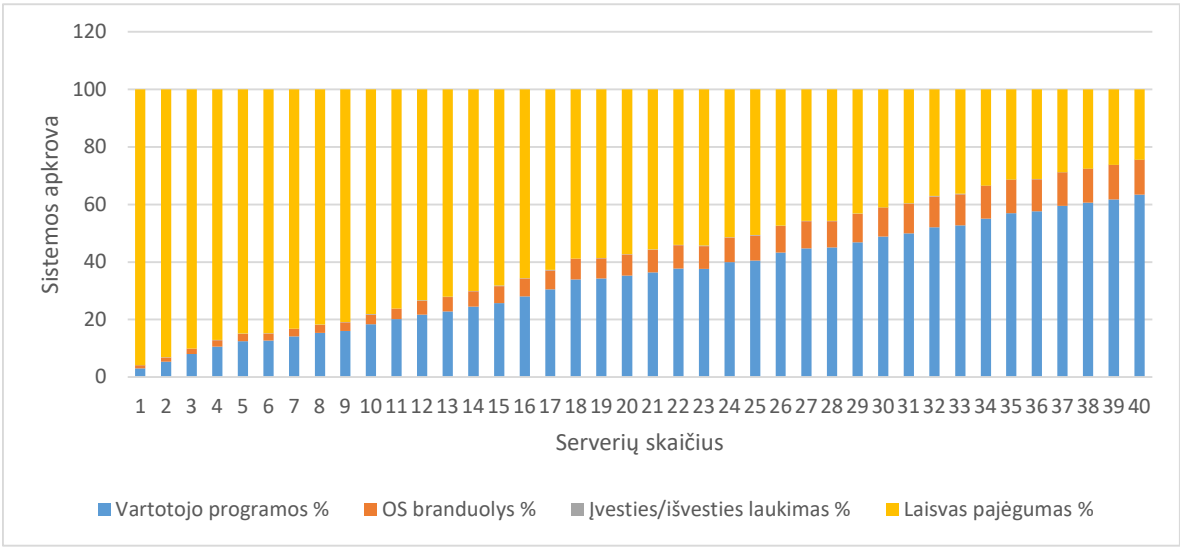
Iš sistemos apkrovos grafiko 3.12 pav. galime matyti, kad:

– Prie 40 lygiagrečiai tikrinamų serverių, stebėtojo sistema nebeturi laisvų procesoriaus resursų, tikrinant daugiau serverių su „Tik aparatinės įrangos“ politika gali stipriai sulėtinti tikrinimo greitį;
– Tikrinant 40 serverių lygiagrečiai stebėjimo serverio operacinė sistema sunaudoja 20% procesoriaus darbo.



3.13 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Tik aparatinės įrangos“

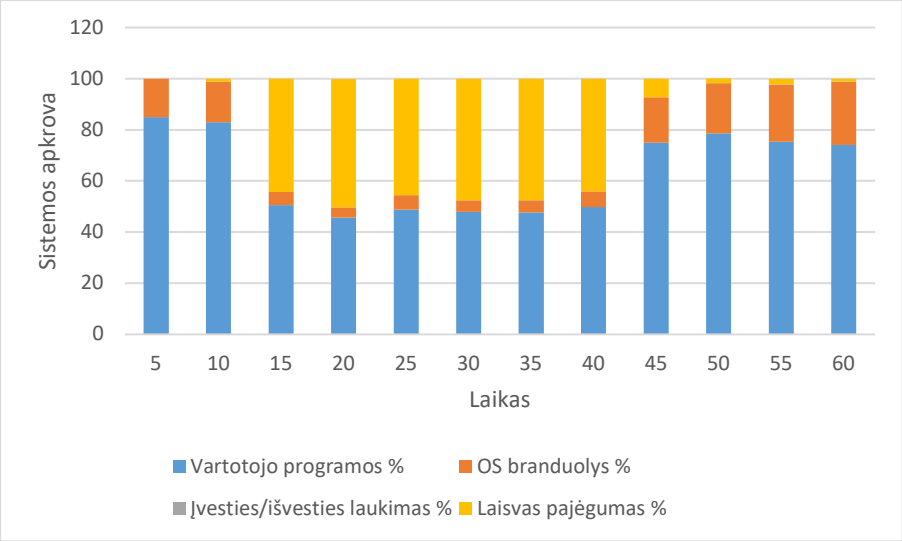
3.13 pav. pavaizduota, kai atliekama 40 tikrinimu lygiagrečiai, stebėjimo sistemos procesorius apkrautas 99,3%, ši apkrova yra pilnai pasiskirsčius viso tikrinimo metu. Tai yra dėl to, kad stebimi serveriai greitai apskaičiuoja vientisumo duomenis ir stebėjimo serveris puola generuoti ataskaitas ir jas siųsti administratoriams.



3.14 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Programos ir konfigūracija“ politiką

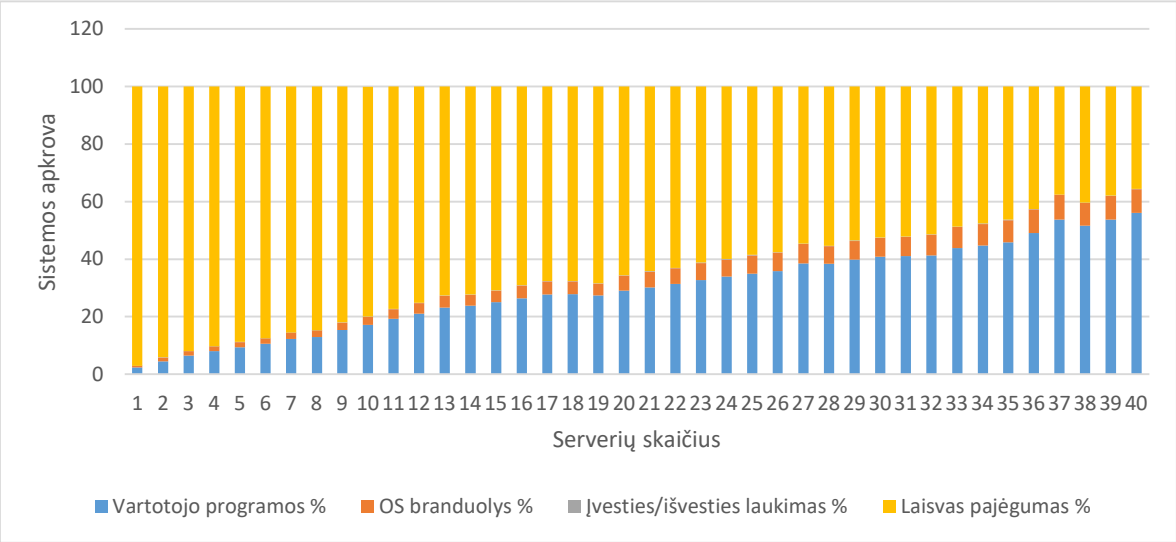
Iš sistemos apkrovos kai naudojama „Programos ir konfigūracija“ politika grafike 3.14 pav. galime matyti, kad:

– Stebėjimo serverio apkrova yra mažesnė, nei kai yra naudojama „Tik aparatinės įrangos“ tikrinimo politika;
– Tikrinant 40 serverių lygiagrečiai stebėjimo serverio operacinė sistema sunaudoja 12% procesoriaus darbo.



3.15 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Programos ir konfigūracija“

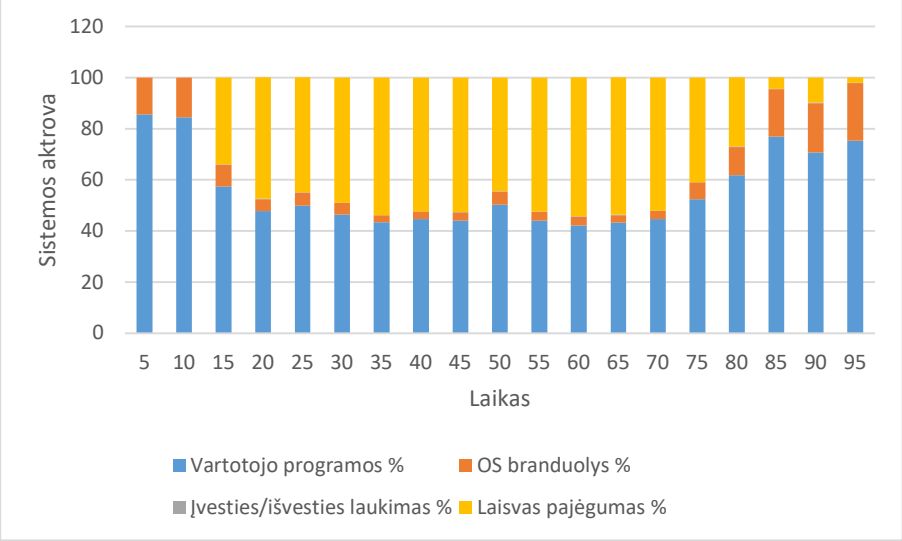
3.15 pav. pavaizduota, kad apkrova stebėjimo serverio pradžioje kyla iki maksimalios kol yra atliekami visi kvietimai ir pradedami lygiagretūs tikrinimai ir nukrenta kai visi stebimi serveriai skaičiuoja vientisumo duomenis. Tuo metu sistema galėtu aptarnauti ir daugiau serverių. Apkrova grįžta prie maksimalios kai stebimi serveriai pabaigia skaičiavimus ir stebėjimo serveris pradeda generuoti ataskaitas ir jas siunčia administratoriams.



3.16 pav. Stebėjimo serverio procesoriaus apkrova kai stebimi serveriai naudoja „Visa sistema“ politiką

Iš sistemos apkrovos kai naudojama „Visa sistema“ politika grafike 3.16 pav. galime matyti, kad:

- Stebėjimo serverio apkrova yra dar mažesnė, nei kai yra naudojama „Programos ir konfigūracija“ tikrinimo politika, dėl to kad skaičiavimai stebimuose serveriuose dar ilgiau užtrunka;
- Tikrinant 40 serverių lygiagrečiai stebėjimo serverio operacinė sistema sunaudoja 8% procesoriaus darbo.



3.17 pav. Sistemos apkrova 40 serverių lygiagretaus tikrinimo metu kuriu politika „Visa sistema“

Taip pat kaip ir „Programos ir konfigūracija“ politikos tikrinimo metu grafike 3.17 pav. matome, apkrova stebėjimo serverio pradžioje kyla iki maksimalios kol yra paleidžiami visi tikrinimai ir nukrenta kol serveriai skaičiuoja vientisumo duomenis. Apkrova grįžta prie maksimalios kai stebimi

serveriai pabaigia skaičiavimus ir stebėjimo serveris pradeda generuoti ataskaitas ir jas siunčia administratoriams.

3.6. Kompiuterinės sistemos vientisumo užtikrinimo sistemos palyginimas su esamomis sistemomis

Šiame skyriuje yra aprašytas palyginimas su kuriama kompiuterinės sistemos vientisumo užtikrinimo sistema ir jau sukurtomis sistemomis. Palyginimas atliktas su įrankiu *Tripwire* [13] ir Elasticsearch B.V kompanijos įrankiu *AuditBeat*. [24]

3.12 lentelė Funkcionalo palyginimas

	Kuriama sistema KSVUS	***Tripwire***	***Auditbeat***
Užtikrina Failų sistemos integralumą	TAIP	TAIP	TAIP
Pranešimai būdai	Paštu, pokalbių sistema (*slack, MS teams*)	Paštu	*Kibana*, *Logstash*
Pranešimas yra siunčiamas kai	Atliekamas vientisumo tikrinimas	Atliekamas vientisumo tikrinimas	Įvykus vientisumo pažeidimui
Atlieka failų atsargines kopijos	TAIP	NE	NE
Tikrina Aparatines įrangos integralumą	TAIP	NE	NE
Naudoja šifruota komunikacija	TAIP	TAIP	TAIP
Centralizuotai valdomas	TAIP	NE	NE
Gauna informacija iš Linux audito karkaso	NE	NE	TAIP
Stebėti vartotojų prisijungimus prie stebimos sistemos	NE	NE	TAIP

Iš 3.12 lentelė galima matyti kiekvienos sistemos turima funkcionalumą. Kuriama sistema turi visas funkcijas, kurias turi *Tripwire* įrankis bei turi papildomas funkcijas leidžiančias atstatyti sistemos vientisumą, tikrinti aparatinės įrangos vientisumą ir leidžia centralizuotai valdyti stebimas sistemas. Tačiau lyginant su *Auditbeat*, kuriamos sistemos trūkumas yra, kad pranešimai yra siunčiami po vientisumo pažeidimo kai yra atliekamas sistemos tikrinimas. Taip pat, kuriama sistema neturi galimybes gauti audito informacijos iš Linux audito karkaso, kuris leidžia matyti, kokie sisteminiai kvietiniai buvo kviečiami. Taip pat kuriama sistema neturi galimybes nustatyti, kokie vartotojai yra prisijungia prie sistemos ir kokius pakeitimus vartotojai atlieka.

3.7. Išvados

- Remiantis suprojektuotu modeliu, yra sukurtas prototipas, kuris leidžia užtikrinti serverių aparatinės ir failų sistemos vientisumą bei pažeistų failų vientisumui atstatyti.
- Pirmojo eksperimento metu buvo tiriama vientisumo skaičiavimų trukmė bei stebimos sistemos apkrova. Per kiek laiko sistema išsiunčia pranešimą administratoriui apie nustatytus vientisumo pažeidimus. Tyrimas buvo atliktas naudojant tris skirtingas politikas pateiktas lentelėje.
- Išmatuotos trukmės bei stebimos sistemos apkrovos yra:
    - Esant politikai „Tik aparatinė įranga“ visa procedūra yra atliekama per 9 sekundes, esant politikai „Visa sistema“ ši trukmė išauga iki 51 sekundės;

    - Stebimame serveryje yra maksimaliai apkraunamas 1 procesoriaus branduolys;
    - Kietojo disko apkrovos gali išaugti iki 50% kai yra atliekamas failų sistemos vientisumo duomenų skaičiavimas;
    - Tinklas nėra apkraunamas kai yra atliekamas tikrinimas;
    - Be aparatinės įrangos ir vientisumo duomenų surinkimo yra reikalingas papildomas laiko tarpas valdymo išlaidoms, nuo 4,02 sek. iki 5,01 sek.
- Antrojo tyrimo metu buvo tiriama kompiuterinės sistemos vientisumo užtikrinimo sistemos lygiagretus kelių serverių tikrinimas
    - Išmatuotos trukmės bei stebėjimo serverio vidutinės procesoriaus apkrovos bei apkrovų pasiskirstymas viso tikrinimo procese;
    - Kuo daugiau tikrinimų vyksta lygiagrečiau tuo ilgiau užtrunka sistemos tikrinimas;
    - Sistema gali atlaikyti bent 40 tikrinimų vienu metu;
    - 40 serverių lygiagretus tikrinimas 2 kartus prailgina tikrinimo trukmė lyginant su vieno serverio tikrinimu;
    - Kai stebimi serveriai turi mažai skaičiavimų, stebėjimo serverio apkrovos iškyla stipriai, esant 40 serverių ir politikai „Tik aparatinė įranga“ stebėjimo serverio procesoriaus apkrovos yra maksimalios;
    - Kai stebimi serveriai atlieka daug skaičiavimus (politikuose „Programos ir konfigūracija“ arba „Visa sistema“) stebėjimo serveryje tuo metu apkrovos ženkliai nukrenta iki 50 %;
    - Tikrinant daug serveriu vienu metu, stebėjimo serverio operacinė sistema gali išnaudoti nuo 8.3 % iki 20 % procesoriaus darbo kontekstų perjungimams bei kitiems OS reikmėms.
- Kuriama sistema turi visas *Tripwire* funkcionalumą bei turi papildomas funkcijas leidžiančias atstatyti sistemos vientisumą, tikrinti aparatinės įrangos vientisumą;
- Tačiau kuriama sistema turi ir trūkumų lyginant su *Auditbeat* sistemą.

4. Išvados

- Neužtikrinus kompiuterinės sistemos vientisumo, duomenys esantys patalpinti informacinėje sistemoje gali prarasti savo prasmę bei vertę ir taip sutrikdyti įmonės darbą. Taip pat pažeidus sistemos vientisumą, pakeitus sistemos nustatymus ar įdiegus papildomą neautorizuota aparatinę įranga, gali būti pažeistas iš sistemoje esančių duomenų konfidencialumas ar pasiekiamumas.
- Užtikrinti kompiuterines sistemos vientisumą galima įvairiais būdais, tai galima atlikti naudojant programinius sprendimus arba aparatinius, tačiau nėra vieno bendro sprendimo, kad būtų užtikrintas sistemos vientisumas.
- Analizuoti produktai turi problema susijusia su programinės įrangos atnaujinimu. Metodai, kurie remiasi aparatiniai įranga, reikalauja kiekviena karta atnaujinus programinę įrangą patvirtinti ar tai leistinas atnaujinimas ir tai privalo padaryt žmogus. Su programiniais metodais kiekvieną karta reikia pergeneruoti vientisumo duomenų bazes ir jas paskui saugiai talpinti, pavyzdžiui perkelti į tik skaitymui skirtas talpyklas.
- Pasiūlytas kompiuterinės sistemos vientisumo užtikrinimo modelis skirtas stebėti aparatinę įrangą bei programiniai įrangai ir užtikrinti, kad sistemoje neatsitiktų nenumatytų pakeitimų.
- Pasiūlyta sistema netik perspėja administratorius dėl įvykusiu sistemoje neautorizuotų pakeitimų, bet ir bando atstatyti sistemos vientisumą.
- Sistemą turi didelį pritaikomumo laipsnį. Administratoriai gali pritaikyti jų reikiamiems panaudos atvejams.
- Atlikus eksperimentinius tyrimus nustatyta:
    - Esant politikai „Tik aparatinė įranga“ visa procedūra yra atliekama per 9 sekundes, esant politikai „Visa sistema“ ši trukmė išauga iki 51 sekundės;
    - Stebimame serveryje yra apkraunamas tik 1 procesoriaus branduolys;
    - Kietojo disko apkrovos gali išaugti iki 50% kai yra atliekamas failų sistemos vientisumo duomenų skaičiavimas;
    - Be aparatinės įrangos ir vientisumo duomenų surinkimo yra reikalingas papildomas laiko tarpas valdymo išlaidoms, nuo 4,02 sek. iki 5,01 sek.;
    - Kuo daugiau tikrinimų vyksta lygiagrečiau tuo ilgiau užtrunka sistemos tikrinimas;
    - 40 serverių lygiagretus tikrinimas 2 kartus prailgina tikrinimo trukmė lyginant su vieno serverio tikrinimu;
    - Kuriama sistema turi visas *Tripwire* funkcionalumą bei turi papildomas funkcijas leidžiančias atstatyti sistemos vientisumą, tikrinti aparatinės įrangos vientisumą.

Literatūra

[1] P. Pinheiro, M. Aparicio ir C. Costa, „Adoption of cloud computing systems," įtraukta *Proceedings of the International Conference on Information Systems and Design of Communication*, New York, 2014.

[2] K. J. Biba, „Integrity Considerations for Secure Computer Systems," MITRE Corporation, Bedford, 1977.

[3] G. Sivathanu, C. P. Wright ir E. Zadok, „Ensuring data integrity in storage: techniques and applications," įtraukta *StorageSS '05 Proceedings of the 2005 ACM workshop on Storage security and survivability*, Fairfax, 2005.

[4] P. S. Bystrov, „Understanding DMA Malware," įtraukta *Detection of Intrusions and Malware, and Vulnerability Assessment.*, Berlin, Heidelberg, 2013.

[5] A. Markuze, A. Morrison ir D. Tsafrir, „True IOMMU Protection from DMA Attacks: When Copy is Faster than Zero Copy," įtraukta *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, 2016.

[6] M. G. Solomon ir M. Chapple, Information Security Illuminated, Sudbury: Jones and Bartlett Publishers, 2005.

[7] H. Delfs ir H. Knebl, Introduction to Cryptography, Springer, Berlin, Heidelberg, 2015.

[8] M. A. Fauzan ir E. Paulus, „A Framework to Ensure Data Integrity and Safety," *Journal of Computing and Applied Informatics,* t. 2, nr. 1, pp. 1-11, 2018.

[9] H. Watanabe, T. Iwama, K. T. Murata, S. Ushiyama ir Y. Muto, „Improvement of the Integrity Verification Application Using Timestamp Mechanism for Distributed File System," įtraukta *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, Vasteras, 2014.

[10] B. Liu, X. L. Yu, S. Chen, X. Xu ir L. Zhu, „Blockchain Based Data Integrity Service Framework for IoT Data," įtraukta *IEEE International Conference on Web Services (ICWS)*, Honolulu, 2017.

[11] I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev ir L. Yalansky, „Ensuring Data Integrity Using Blockchain Technology," įtraukta *20th Conference of Open Innovations Association (FRUCT)*, St. Petersburg, 2017.

[12] G. H. Kim ir E. H. Spafford, „The design and implementation of tripwire: a file system integrity checker," įtraukta *CCS '94 Proceedings of the 2nd ACM Conference on Computer and communications security*, Fairfax, 1994.

[13] Tripwire, Inc., „Open Source Tripwire," [Tinkle]. Available: https://github.com/Tripwire/tripwire-open-source. [Kreiptasi 13 12 2018].

[14] J. Kaczmarek ir M. R. Wrobel, „Operating system security by integrity checking and recovery using write-protected storage," *IET Information Security,* pp. 122-131, March 2014.

[15] C. A. Adukkathayar, G. S. Krishnan ir G. Sasikumar, „Advanced integrity checking and recovery using write-protected storage for enhancing operating system security," įtraukta *2015 10th International Conference on Computer Science & Education (ICCSE)*, 2015, Cambridge.

[16] O. S. Hofmann, A. M. Dunn, S. Kim, I. Roy ir E. Witchel, „Ensuring operating system kernel integrity with OSck," įtraukta *ASPLOS XVI Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, Newport Beach, 2011.

[17] Trusted Computing Group., „TPM Main Specification," 2007. [Tinkle]. Available: http://www.trustedcomputinggroup.org/resources/tpm_. [Kreiptasi 07 01 2019].

[18] L. Wang ir P. Dasgupta, „Kernel and Application Integrity Assurance: Ensuring Freedom from Rootkits and Malware in a Computer System," įtraukta *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, Niagara Falls, 2007.

[19] F. Zhang, J. Wang, K. Sun ir A. Stavrou, „HyperCheck: A Hardware-Assisted Integrity Monitor," *IEEE Transactions on Dependable and Secure Computing,* t. 11, nr. 4, pp. 332-344, 2014.

[20] Elasticsearch B.V, „Elastic Stack and Product Documentation," [Tinkle]. Available: https://www.elastic.co/guide/index.html. [Kreiptasi 29 04 2020].

[21] J. Delvare ir A. Cox, „dmidecode," 14 09 2018. [Tinkle]. Available: https://www.nongnu.org/dmidecode/. [Kreiptasi 09 06 2019].

[22] L. Vincent, „Hardware Lister (lshw)," [Tinkle]. Available: https://ezix.org/project/wiki/HardwareLiSter. [Kreiptasi 09 06 2019].

[23] Red hat, Inc., „Ansible is Simple IT Automation," 2019. [Tinkle]. Available: https://www.ansible.com/. [Kreiptasi 09 06 2019].

[24] Elasticsearch B.V., „Auditbeat overview," [Tinkle]. Available: https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-overview.html. [Kreiptasi 29 04 2020].