KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGA

EDGARAS LUKOŠEVIČIUS

ĮVYKIŲ ŽURNALŲ VIENTISUMO AUDITO METODŲ TYRIMAS

Magistro baigiamasis darbas

Vadovas
lekt. dr. Dangis Rimkus

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGA

EDGARAS LUKOŠEVIČIUS

ĮVYKIŲ ŽURNALŲ VIENTISUMO AUDITO METODŲ TYRIMAS

Magistro baigiamasis darbas

Vadovas
lekt. dr. Dangis Rimkus

Recenzentas
doc. dr. Jevgenijus Toldinas

AUTORIŲ GARANTINIS RAŠTAS

DĖL PATEIKIAMO KŪRINIO

2013 - 05 - 23 d.
Kaunas

Autorius, Edgaras Lukoševičius **,**
(vardas, pavardė)

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis magistro darbas (toliau vadinama – Kūrinys) "ĮVYKIŲ ŽURNALŲ VIENTISUMO AUDITO METODŲ TYRIMAS"
(kūrinio pavadinimas)

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

Edgaras Lukoševičius _____
(vardas, pavardė) (parašas)

SANTRAUKA

Ne be priežasties sistemos įvykių žurnalai yra vienas pirminių taikinių po sėkmingo įsilaužimo į sistemą. Sistemos įvykių žurnaluose yra kaupiama įvairi informacija pradedant nuo įprastų sistemos būklės ir resursų sąnaudų pranešimų ir baigiant jautria informacija, tokia kaip transakcijų duomenys, naudotojų autentikacijos duomenys ir kita konfidencialia informacija. Tačiau, dažniausiai įvykių žurnalai taikiniu tampa todėl, kad juose yra kaupiami duomenys apie sistemos sukompromitavimą, naudotas atakos technikas, nepavykusius prisijungimus, saugumo pažeidžiamumus, kuriais buvo pasinaudota įsilaužiant į sistemą. Dauguma šios informacijos galėtų būtų panaudota atlikti analizei po sistemos sukompromitavimo. Kiekvienas patyręs įsibrovėlis norės pašalinti sistemos sukompromitavimo pėdsakus taip siekdamas užkirsti kelią sistemos administratoriui(-ams) nustatyti už įsibrovimą atsakingą subjektą ir galimai pašalinti saugumo spragas. Norint, kad įvykių žurnalai būtų saugūs ir jais būtų galima pasitikėti reikia imtis atitinkamų priemonių, kad užkirsti kelią neautorizuotam įvykių žurnalų redagavimui ar kitokiam klastojimui, o jeigu toks klastojimas įvyksta, tai turi būti akivaizdu.

Analitinėje dalyje nustatėme pagrindinius atakų tipus, kurie gali būti panaudoti prieš sistemos įvykių audito žurnalus, atlikome pagrindinių GNU/Linux sistemose naudojamų įvykių žurnalų registravimo servisų (programų) analizę, taip pat atlikome esamų apsaugos metodų, kurie gali apsaugoti įvykių audito žurnalus, analizę. Padarėme išvadą, kad kiekvienas metodas turi privalumų bei trūkumų ir nėra vieno universalaus metodo, kuris gebėtų apsaugoti nuo visų analitinėje dalyje aprašytų atakų, todėl metodai yra modifikuojami arba naudojami įvairūs šių metodų deriniai. Vieni svarbiausių tokių metodų kriterijų yra greitaveika ir resursų sąnaudos.

Praktinėje dalyje atlikome metodų, kurių analizė buvo atlikta analitinėje dalyje, tyrimus ir palyginome juos tokiais atžvilgiais: saugių audito žurnalo sudarymo greitis, sudarytų saugių žurnalų audito greitis, atsparumas atsisakymo aptarnauti atakai, kuri gali būti sukelta dėl didelio įvykių srauto netgi iš patikimų sistemų, ir talpyklos resursų sąnaudos. Maišos grandinių metodas tyrimo metu buvo greičiausias tiek žurnalų sudarymo, tiek audito metu, taip pat atspariausias atsisakymo aptarnauti atakai bei reikalavo mažiausiai talpyklos vietos sąnaudų, tačiau šis metodas nėra atsparus bazinėms atakoms, tokioms kaip nukirtimo ataka. Kita vertus Merklio ir istorijos medžiai derinyje su RSA arba DSA parašais yra atsparūs daugumai atakų, tačiau yra lėtesni metodai. Apibendrinant galima daryti išvadą, kad Merklio medis, lyginant su istorijos medžiu, yra tinkamesnis žurnalų auditui, atsparesnis atsisakymo aptarnauti atakai ir reikalauja mažiau talpyklos vietos sąnaudų. Istorijos medis yra šiek tiek greitesnis žurnalų sudarymo metu, tačiau prastesnis visais kitais atžvilgiais.

ANALYSIS OF LOG INTEGRITY AUDITING METHODS

SUMMARY

System logs are one of the primary targets after successful break-in and for a good reason. System logs contain data ranging from usual system health and resource consumption reports to sensitive data such as transaction data, authentication data of users and other confidential information. But usually system logs become primary targets because they contain data about system compromise, attack techniques, failed logins and exploits used in the break-in. Most of this information could be used in post-mortem analysis. Every experienced intruder wishes to erase traces of the compromise to prevent system administrator(s) from fixing security holes and determining the subject responsible from system compromise. To make the audit log secure, we must take appropriate measures to prevent attacker from modifying system logs, and if they do – it must be evident.

In the analytical part of this work we have described most common attack types that can be used against system logs and the most common logging services (daemons) used in the GNU/Linux family operating systems. We have also made an analysis of present security methods that could solve this particular problem. We have concluded, that every method has its pros and cons and there is no single method that could resist against every attack type described in analytical part of this work so the methods are usually modified or used in combination with other methods. One of the most important criteria for these methods or combinations are efficiency, performance and resource consumption.

In the research part of this work we have made a research on some of the methods described in the analytical part and made a comparison in aspect of: secure log building performance, auditing performance of secure event log, resistance against denial-of-service attack, that can be intentionally caused by flooding system with intense stream of logs and storage consumption. Hash-chain was the fastest and the most resistant method against denial of service attacks, it also requires least storage space compared to Merkle tree and history tree, but it lacks resistance against basic attack methods such as trunctation attack. On the other hand Merkle and history tree in combination with RSA or DSA signing are resistant to the most of the attacks, but are slower. In conclusion, Merkle tree is better at verifying audit logs, storage consumption and resistance agains denial of service attack. History tree is only slightly faster at building system logs but is worse at other scenarios.

TURINYS

PAVEIKSLĖLIŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

CIA CIA triada yra viena iš kertinių informacijos saugumo principų. CIA triadą (dar kitaip vadinamą trikampį) sudaro: konfidencialumas, vientisumas, prieinamumas (angl. „Confidentiality, Integrity and Availability“)

FI Išankstinis vientisumas, tai metodo savybė, kuria pasižymintis metodas garantuoja praeities duomenų saugumą, tačiau negarantuoja ateities. Įvykių (audito) žurnalų kontekste tai yra metodas, kuris garantuoja, kad po sistemos ar įvykių žurnalų sukompromitavimo piktavalis negalės nepastebimai pakeisti (sunaikinti ar kažkaip klastoti) įvykių žurnalo turinio esančio praeityje (prieš sukompromitavimą), tačiau negarantuoja, kad piktavalis negalės keisti ateities duomenų. (angl. „Forward Integrity“)

FS Išankstinis saugumas, tai metodo savybė, labai panaši į FI. Įprasti skaitmeniniai prašai turi būdingą silpnybę: jeigu slaptas raktas yra sukompromituotas, tai visi parašai, netgi sugeneruoti prieš sukompromitavimą, yra nebepatikimi. Šiai silpnybei adresuoti buvo pasiūlytas išankstinio saugumo skaitmeniniai parašai: jie garantuoja, kad praeities parašai yra saugūs netgi sukompromitavus esamą slaptą raktą.[18] Panaši koncepcija gali būti pritaikoma ir kitose kriptografijos srityse (pavyzdžiui, FSPRG). (angl. „forward-security“)

FSPRG Išankstiniu saugumu (FS) paremtas pseudo-atsitiktinis generatorius. Tai pseudo-atsitiktinis generatorius, kuris garantuoja, kad iš pradinis slaptas raktas, iš kurio yra generuojami nauji raktai, bus saugus netgi sukompromitavus vieną iš raktų. (angl. „forward-secure pseudo-random generator“)

FssAgg Išankstinio saugumo nuoseklus agregavimas. Tai metodas, paremtas FS, kurio esmė yra iš pradinio sertifikato generuoti naujus FS ypatybe pasižyminčius sertifikatus, o po to šiuos visus sertifikatus apjungiant į vieną bendrą sertifikatą. Taip yra taupomos talpyklos sąnaudos ir atsispiriama kai kurioms atakų rūšims. (angl. „forward-secure sequential aggregation)

GNU Tai nemokama, atviro kodo operacijų sistema, dažnai vadinama tiesiog Linux, tačiau tai yra klaidinga, kadangi Linux yra tik operacijų sistemos branduolys, o vartotojo erdvės (angl. „user-space“) įrankių visuma, leidžianti vartotojams dirbti su operacijų sistema, yra GNU (angl. rekursyvus trumpinys „GNU Not UNIX“)

IETF Ne pelno siekianti organizacija apsiimanti su interneto technologijomis susijusių protokolų ir kitų sričių standartizavimu (angl. The Internet Engineering Task Force)

PRF Pseudo-atsitikinė funkcija – tai funkcija, kurios neįmanoma atpažinti tarp kitų tokių pačių funkcijų esančių rinkinyje. PRG yra PRF dalis (angl. „pseudorandom function“)

PRG Pseudo-atsitiktinis generatorius – tai deterministinė procedūra, kuri iš pradinio „grūdo“ (angl. „seed“) geba generuoti naujas pseudo-atsitiktines reikšmes taip, kad jokia statistinė ar kitokia analizė negali nustatyti generatoriaus algoritmo pagal jo išvedamas reikšmes (angl. „pseudorandom generator“)

TCSEC Patikimų kompiuterinių sistemų įvertinimo kriterijai – tai knyga, dar kitaip vadinama „oranžine knyga“ (angl. „Trusted Computer System Evaluation Criteria“ arba „Orange book“)

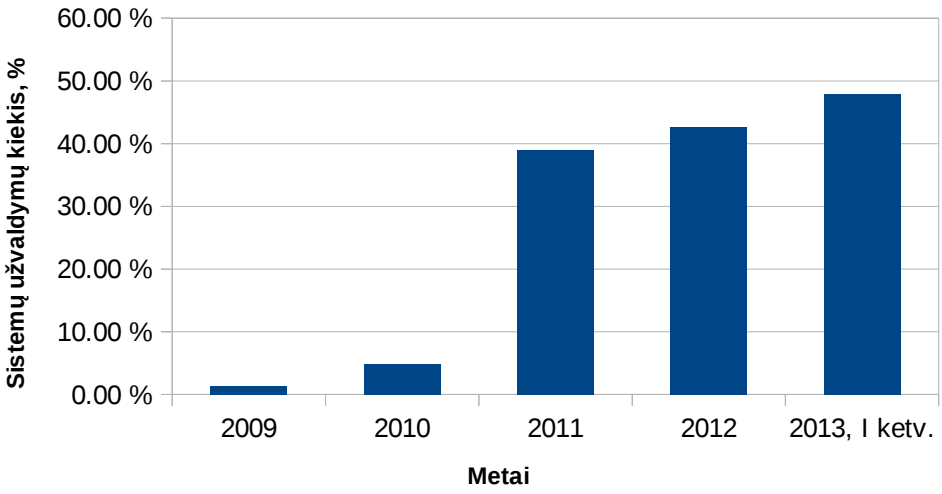
WORM Vienkartinio įrašymo-daugkartinio skaitymo atmintis – tai atminties tipas į kurį įrašius duomenis jų nebeįmanoma niekaip koreguoti. Rašymo apribojimas garantuoja, kad duomenų tokioje atmintyje nebebus galima koreguoti, todėl duomenys tampa atsparūs klastojimui. Vienas populiariausių tokios atminties pritaikymų yra CD ir DVD kompaktinės plokštelės.

1. PROBLEMA

Ne be pagrindo po įsiskverbimo į sistemą įvykių registravimo žurnalai tampa vienu iš pirminių taikinių – įvykių registravimo žurnalai yra prastai apsaugoti arba išvis neapsaugoti, o juose yra saugoma jautri informacija pradedant įprastais pranešimais apie sistemos veiklą, resursų išnaudojimą, automatines užduotis ir baigiant nepavykusiais prisijungimais prie sistemos, slaptažodžiais, konfidencialia, vidine, įmonės informacija ir įkalčiais apie įsiskverbimą į sistemą. Dauguma šių įvykių gali būti panaudota post-mortem analizei po sistemos sukompromitavimo. Patyręs sistemos puolėjas norėdamas išlikti nepastebėtas sieks pašalinti sistemos sukompromitavimo įkalčius ir nuslėpti sistemos puolimo metodą tam, kad sistemų administratoriai negalėtų nustatyti ir sutvarkyti atakos metu išnaudotų sistemos spragų [1], todėl vienas pirmųjų veiksmų, kurios atlieka sistemos puolėjas, yra įkalčių apie jo būvimą sistemoje sunaikinimas – įvykių žurnalo išvalymas, modifikavimas ar kitoks duomenų klastojimas. Dažniausiai įvykių žurnalai yra saugomi kaip paprasto teksto failai, kuriuose įvykiai yra registruojami kaip paprasto teksto eilučių sekos, o dažniausia šių failų apsauga yra tiesiog failų sistemų leidimų paskirstymas (skaitymas, rašymas) vartotojams. Toks apsaugos mechanizmas sąlygoja tai, kad nėra jokių garantijų, kad įvykių registravimo žurnaluose sukaupti duomenys yra teisingi ir/ar autentiški. Po sistemos sukompromitavimo sistema atsiduria visiškoje įsibrovėlio valioje [2].

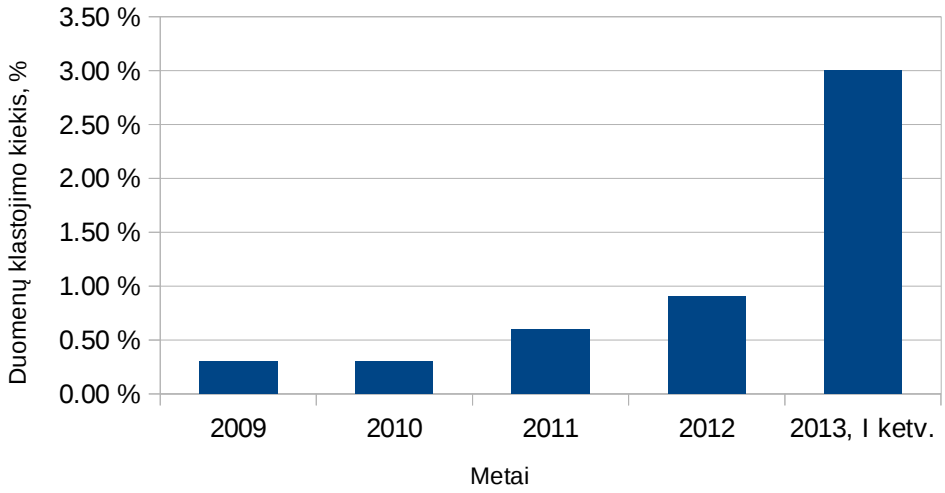
Kadangi šiuose žurnaluose sukauptą informaciją galima panaudoti nusikalstamai veikai nustatyti bei atskleisti ir panaudoti šiuos įkalčius teisme, tai tokia informacija yra itin svarbi teisėsaugai. Dauguma šalių turi teisės aktus, kuriuose numatytas įvykių registravimo žurnalų duomenų kaupimas bei pateikimas teisėsaugos institucijoms, pavyzdžiui, Lietuvoje [3]. Bet dėl įvykių registravimo žurnaluose esančių duomenų nepatikimumo turimais duomenimis negalima arba pasitikėti išvis arba juos reikia papildomai ir kruopščiai tikrinti, be to tokius įkalčius galima užginčyti teisme.

2009-2013m. Lietuvos Respublikos nacionalinio elektroninių ryšių tinklų ir informacijos saugumo incidentų tyrimo padalinio statistikos duomenimis[1] sistemų užvaldymo (įsilaužimo) (žr. Pav. 1) ir duomenų klastojimo (žr. Pav. 2) incidentų kiekis nuolat auga, todėl saugi įvykių registravimo sistema, kurios sukauptų duomenų auditu galime pasitikėti, yra vis svarbesnė užduotis.



Pav. 1: Sistemų užvaldymo statistika Lietuvoje

1 Duomenys viešai pateikiami CERT LT svetainėje – https://www.cert.lt/



Pav. 2: Elektroninių dokumentų/duomenų klastojimo statistika Lietuvoje

Tam kad įvykių registravimo žurnalas būtų patikimas būtina naudoti metodus galinčius užkirsti kelią įvykių registravimo žurnalo klastojimui ir gebančius tiksliai aptikti žurnalų klastojimo faktą. Be to, turbūt, visoms sistemoms yra svarbu ne tik užtikrinti duomenų vientisumą ir audito žurnalų patikimumą tačiau ir atlikti audito žurnalo duomenų sudarymą ir šių žurnalų auditą su minimaliais resursų reikalavimais.

2. TIKSLAS IR UŽDAVINIAI

Ištirti įvykių žurnalų vientisumo audito atlikimo metodus ir nustatyti kuris metodas auditui atlikti reikalauja mažiausiai resursų bei suformuluoti rekomendacijas įmonės įvykių žurnalų vientisumo audito politikai.

Tikslo siekiui būtina įvykdyti šiuos uždavinius:

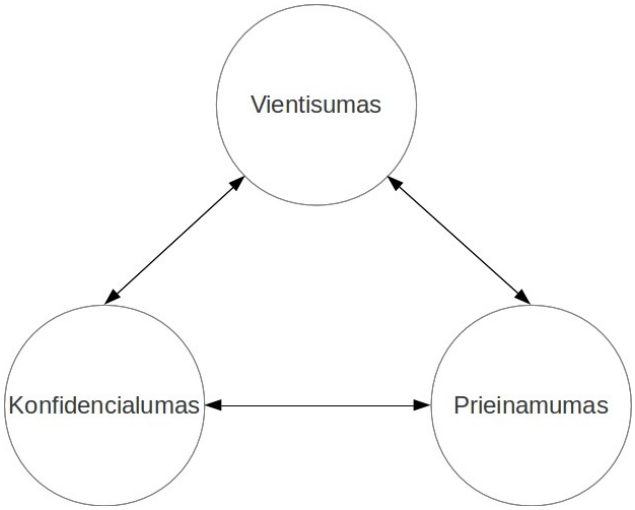
- Atlikti įvykių žurnalų vientisumo audito metodų analizę;
- Ištirti žurnalų vientisumo audito metodus;
- Atlikti žurnalų auditą pasitelkus šiuos metodus;
- Ištirti ir nustatyti kuris metodas auditui atlikti reikalauja mažiausiai resursų;
- Suformuluoti rekomendacijas įmonės žurnalų vientisumo apsaugos metodams ir vientisumo audito politikai.

3. ANALIZĖS DALIS

Oranžinėje knygoje (TCSEC)[4] apie reikalavimus audito žurnalams yra rašoma taip: Audito informacija turi būti kaupiama ir saugoma taip, kad veiksmai, liečiantys saugumo problemas galėtų būti atsekami iki atsakingo subjekto. Patikima sistema privalo gebėti įrašyti visus saugai aktualius įvykius į audito žurnalą. Galimybė kaupti tik pasirinktus audito duomenis yra būtina siekiant sumažinti audito sąnaudas ir atlikti efektyvią analizę. Audito duomenys privalo būti apsaugoti nuo modifikavimo ir neautorizuoto sunaikinimo tam, kad būtų galima atlikti saugumo pažeidimų tyrimą po fakto (pavėluotai).

Šiuo metu įvykių registravimo žurnalų duomenims saugiai kaupti dažniausiai naudojamas būdas yra jų siuntimas į nutolusį kompiuterį, o dažniausias audito atlikimo būdas yra tiesiog fizinis ir/ar pusiau automatinis šių žurnalų skaitymas ir tam tikros informacijos paieška. Tačiau perduodant duomenis į kitą kompiuterį jie vis vien nėra garantuotai apsaugoti. Paketai gali būti perimami siuntimo metu, klastojami, įterpiami, taip pat gali būti nutrauktas ryšys, ko pasėkoje kompiuteris, siunčiantis įvykius į nutolusį kompiuterį, norėdamas apsaugoti duomenis nuo praradimo, turės juos kaupti lokaliai, o lokaliai kaupiami duomenys tuo pačiu tampa įsibrovėlių taikiniu. IETF yra patvirtinusi standartus [5,6], skirtus saugiam įvykių žurnalo įrašų perdavimui tinklu, tačiau, netgi pavykus apsaugoti duomenis perdavimo į kitus kompiuterius metu vis vien lieka lokalaus įvykių kaupimo problema, o apie sistemos sukompromitavimą ir duomenų suklastojimą nebus žinoma, kol šio fakto, galimai labai pavėluotai, nepatikrins saugumo administratorius, nes bus *tikima*, kad perduodami duomenys yra saugūs ir automatizuotas auditas bus atliekamas tik tuo atveju kai įsilaužimo į sistemą padariniai bus akivaizdūs. Įvairiuose šaltiniuose [7,8,9] teigiama, kad yra neįmanoma apsaugoti (arba tai yra ypač sunku) duomenų nuo to laiko kai sistema yra sukompromituojama ir/ar sistemą puolantis asmuo ar asmenys įgyja administratoriaus teises.

Apsaugoti įvykių žurnalus panaudojant kriptografiją galima siunčiant duomenis į nutolusį kompiuterį ir tikrinant jų integralumą nutolusioje sistemoje, tačiau šis metodas, kaip buvo minėta taip pat turi įvairių saugumo problemų. Remiantis CIA kompiuterių sistemų saugumo triada – *konfidencialumas*, *vientisumas*, *prieinamumas*, toks duomenų apsaugos metodas yra iš esmės, potencialiai, nesaugus, nes *prieinamumas* gali bet kuriuo metu išnykti tiek dėl tyčinių veiksmų, tokių kaip atsisakymo aptarnauti atakos, tiek dėl netyčinių žmogiškųjų klaidų ir/ar gamtos jėgų. Taigi, jeigu, kaip įvykių registravimo žurnalų apsaugos mechanizmą, naudosime duomenų siuntimą į nutolusį serverį, tačiau ryšys su nutolusia sistema nebus įmanomas, tai apsaugos triada nebeveiks.



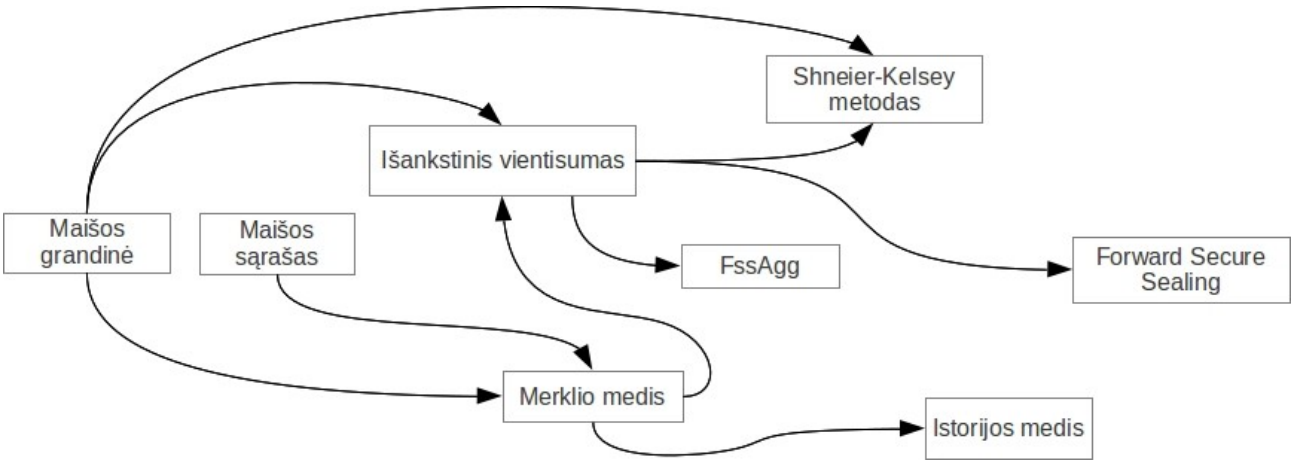
Pav. 3: CIA saugumo modelio triada

Kitas apsaugos būdas yra įvykių žurnalo įrašus autentifikuojančių duomenų (maišos funkcijų eilutes ar kitokių) rašymas į vienkartinio įrašymo laikmenas, pavyzdžiui, CD ar DVD, tačiau šis būdas yra nepraktiškas, reikalaujantis nuolatinės fizinės sistemų priežiūros laikmenų keitimui. Taip pat galima paskutinę maišos funkcijos eilutę nuolat rašyti į vienkartinio įrašymo-daugkartinio skaitymo atmintį – WORM, tačiau šis atminties tipas yra brangus ir sąlyginai mažų panaudos galimybių, todėl yra naudojamas nebent specializuotose sistemose, be to ši atmintis, kaip ir CD, DVD laikmenos, privalo būti nuolat keičiamos. Tačiau kad ir kokiu būdu bus išsaugoti įvykių žurnalų duomenys be specialių metodų nebus įmanoma automatiškai atlikti žurnalų audito ir įvertinti ar duomenys yra patikimi ir nesuklastoti.

Įvairios nuomonės [1,7,8,9,10,11] sutinka ir pagrindžia faktą, kad apsaugoti duomenis sukompromituotoje sistemoje yra labai sunku arba neįmanoma. Kiekvieną apsaugą galima apeiti jeigu ne programiniame lygyje, tai fiziniame (grėsmės nuo kurių reikia apsaugoti audito sistemą yra aprašytos toliau analitinėje dalyje).

Audito žurnalai yra beverčiai, jeigu niekas jų neskaito [9]. Būtent todėl reikalinga programinė įranga, kuri gali stebėti audito žurnalus ir ieško įtartinų pakitimų.

Šiuo metu yra keli metodai, galintys pasiūlyti audito žurnalų apsaugą ir pačių žurnalų audito atlikimą. Tokių metodų sukūrimą skatino įvairios problemos: svarbių kriptografijos problemų sprendimas, audito žurnalų problemų sprendimas, kitų metodų trūkumai ir t.t.



Pav. 4: Metodų, pritaikomų audito žurnalų apsaugai, ryšiai

Kaip matome iš 4 paveikslėlio maišos grandinės yra daugumos metodų pagrindas, kuriuo remiasi kiti metodai, arba kurio problemos lėmė naujų metodų sukūrimą. Toliau skyriuje yra atlikta šių metodų (žr. Pav. 4) analizė – veikimo principai, sprendžiamos problemos, privalumai bei trūkumai. Taip pat šiame skyriuje yra pateikiama esamų įvykių registravimo (ir audito) sistemų apžvalga, jų privalumai, trūkumai, saugumo problemos bei apžvelgiami iki šiol atlikti darbai bei pasiūlyti metodai įvykių registravimo žurnalų auditui atlikti, šių metodų (ne)veiksmingumas, sprendžiamos problemos ir jų pačių saugumo problemos.

3.1. Įvykių registravimo sistemų analizė

Šiuo metu GNU/Linux operacinių sistemų šeimoje yra naudojami du pagrindiniai įvykių registravimo servisai – syslog ir visos jo atmainos: syslog-ng, rsyslog ir t.t., bei gana naujas servisas pavadinimu „the journal“, kuris stipriai skiriasi nuo įprasto syslog. Tačiau Linux OS pasižymi lankstumu, todėl auditorius ir įvykių kaupimo bei apdorojimo sistema gali būti bet kokia programa, kuri sugeba priimti duomenis, juos apdoroti ir kur nors išsaugoti. Šiame skyriuje yra apžvelgiami abu

servisai, jų privalumai ir trūkumai.

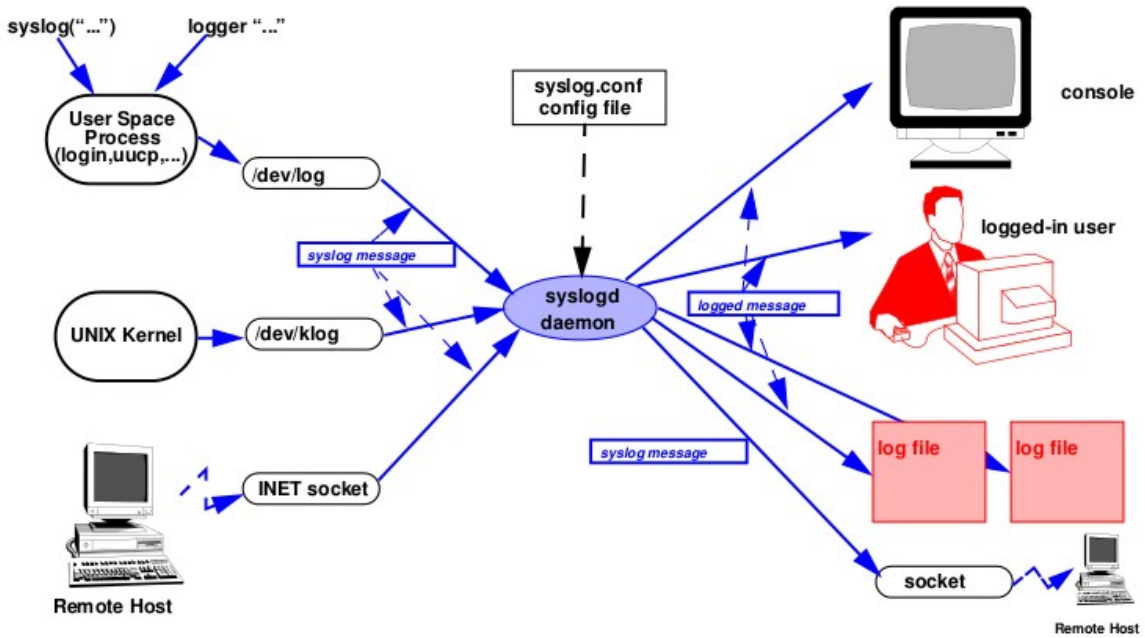
3.1.1. „Syslog“ architektūra ir problemos.

Syslog architektūra buvo sukurta atsižvelgiant į tų dienų aktualijas ir buvo skirta tiesiog įvykiams kaupti, o ne juos apsaugoti ir saugumas tuo metu nebuvo didžiausias sistemos prioritetas, todėl syslog nebeatitinka šių dienų poreikių saugumo klausimu. Be to syslog serveriai dažnai veikia kaip centrinės įvykių kaupimo stotys, kuriose duomenų srautas gali būti itin didelis, todėl reikia kuo mažiau resursų reikalaujančių metodų audito žurnalų generavimui ir auditui atlikti.

Syslog architektūra susideda iš šių dalių:

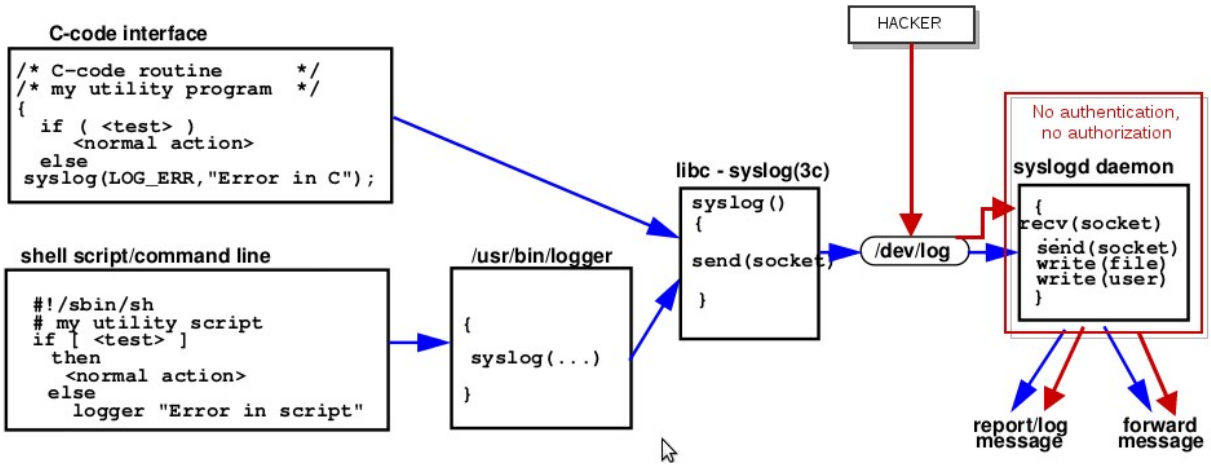
- Žinučių formato specifikacija (syslog.h)
- Sisteminiai iškvietimai ir įrankiai formuojantys žinutes (syslog.c, logger)
- Žinučių perdavimo taškai:
  - /dev/log
  - /dev/klog
  - UDP 514 prievadas
- Servisas priimantis, apdorojantis ir išsiuntinėjantis gautas žinutes į jų paskirtą vietą (syslogd)

Žinutės (įvykiai) syslog architektūroje gali būti išsiųstos ir priimtos keliais būdais. Bet kuri programa pasinaudodama standartinėmis libc bibliotekos funkcijomis gali atverti /dev/log jungtį (angl. „socket“) ir perduoti žinutę. Syslog servisas laukia siunčiamų žinučių taip pat /dev/log jungtyje ir, priklausomai nuo konfigūracijos, atlieka numatytus veiksmus. Šie veiksmai gali būti žinutės ignoravimas (blokavimas), įrašymas į failą, išvedimas į konsolę arba (per)siuntimas nutolusiam kompiuteriui.



Pav. 5: Syslog dizainas [12]

Iš sistemos (programavimo) lygio architektūra atrodo šiek tiek kitaip. Čia jau galime tiksliai įsivaizduoti, kurios architektūros vietos yra silpniausios, ir kuriose būtų galima sukurti papildomą saugumą.



Pav. 6: Syslog architektūra programavimo lygmenyje [12]

Matome, kad syslog (žr. Pav. 6 „syslogd daemon“) yra tas sistemos taškas, kuris turėtų pasirūpinti sudaromų įvykių žurnalų autentiškumu, autorizavimu.

Syslog architektūra yra paremta paprasto teksto failais, kuriuose įrašų autentiškumas nėra niekaip tikrinamas ir/ar autorizuojamas, tai reiškia, kad bet kuri programa gali prisistatyti kitos programos vardu ir siųsti suklastotą informaciją taip klaidinant sistemą ir sistemos administratorių.

Įvykių registravimo failais galima lengvai manipuliuoti, klastoti, juos galima ištrinti, perrašyti, sukeisti ir t.t. Prieigos kontrolė taip pat neegzistuoja – sistemos naudotojai gauna pilną prieigą prie įvykių registro arba išvis jokios prieigos. Tai reiškia, paprasti sistemos naudotojai gali matyti bet kokią sistemos įvykių registravimo žurnalų informaciją, arba nematyti išvis.

Reikia atkreipti dėmesį, kad be įprasto syslogd įvykių registravimo serviso yra sukurta nemažai alternatyvų, kurios turi daugiau galimybių, lankstesnes įvykių valdymo sistemas ir daug kitų privalumų, tačiau šiuo metu nėra nei vienos syslog principu veikiančios alternatyvos, kuri gebėtų autorizuoti, autentifikuoti ar kitaip apsaugoti įvykių audito žurnalus.

3.1.2. „The Journal“ architektūra ir problemos

Šiuo metu yra aktyviai vystoma Linux sistemos ir servisų valdymo sistema „systemd“. 2011 m. Lapkričio 18 d. kaip viena iš projekto dalių buvo sumanyta visiškai nauja sistemos įvykių registravimo sistema, kuri taptų „systemd“ dalis. Ši įvykiu registravimo sistema pavadinta „The Journal“ [13] sistemos sumanytojų ir kūrėjų tikslais yra visiškai naujo dizaino įvykių registravimo sistema Linux operacijų sistemoms, kuri išspręstų daugumą ar netgi visas „syslogd“ problemas, o tuo pačiu pasiūlytų naujų galimybių.

„The Journal“ dizainas gerokai skiriasi nuo įprasto „syslogd“. Aplikacijos ir servisai negali tiesiogiai sudaryti ir siųsti visos žinutės, kaip tai yra daroma „syslog“ atveju, vietoj to jos gali generuoti įvykių žurnalo įrašus siųsdamos „journald“ servisui įrašų laukelius. Gavęs būsimo įrašo laukelius „journald“ servisas pats surenka papildomą informaciją apie įrašą, įrašo laukelius siunčiančią programą ir prideda šiuos meta-laukus prie gauto įvykio pranešimo. Šie papildomai surinkti meta-laukai yra patikimos reikšmės ir negali būti siunčiamos iš klientų, tai reiškia negali būti ir padirbtos (nebent vėliau redaguojant audito žurnalą). Laukeliai, kurios prideda pats „journald“ servisas prasideda prefiksu „_“, taip pažymint, kad šis laukelis yra patikima reikšmė, o ne pateikta potencialiai piktavališko kliento, nes klientai (servisai, aplikacijos ir t.t.) negali perduoti laukelių pavadinimų prasidedančių pabraukimo simboliu [13,14].

Journald įvyko registravimo pavyzdys:

```
_SERVICE=systemd-logind.service
_EXE=/lib/systemd/systemd-logind
_COMM=systemd-logind
_CMDLINE=/lib/systemd/systemd-logind
_PID=4711
_UID=0
_GID=0
_SYSTEMD_CGROUP=/system/systemd-logind.service
_CGROUPS=cpu:/system/systemd-logind.service
PRIORITY=6
_BOOT_ID=422bc3d271414bc8bc95870f222f24a9
_MACHINE_ID=c686f3b205dd48e0b43ceb6eda479721
_HOSTNAME=waldi
LOGIN_USER=500
MESSAGE=User harald logged in
MESSAGE_ID=422bc3d271414bc8bc9570f222f24a9
```

Taip yra žinoma, kad pranešimą siuntusi programa tikrai yra /lib/systemd/systemd-logind, o ne piktavališko sistemos naudotojo programa /home/systemd-logind, kuri galėtų siųsti klaidinančią informaciją. Tai yra papildomas apsaugos lygis - keli patikimi laukai. Žinoma šioje vietoje audito žurnalo patikimi laukai yra patikimi tol, kol jie patenką į patį audito žurnalą, kurį vėliau vis tiek galima redaguoti pasitelkus dvejetainius redaktorius.

Taip pat viena „journald“ idėjų buvo užtikrinti įvykių registro įrašų vientisumą. Apie „the journal“ sistemoje naudojamą metodą ir kitus metodus įvykių žurnalo vientisumui užtikrinti rašoma sekančiuose skyriuose.

Kitos problemos, kurias sukelia ir/ar neišsprendžia „The Journal“ sistema, nebūtinai susijusi su saugumu yra šios:

- Systemd yra glaudžiai susietas su Linux branduoliu, o "The journal" tiesiogiai priklausomas tiek nuo Linux branduolio, tiek nuo systemd, tai reiškia, kad OS, kurios nenaudoja systemd, negalės pasinaudoti ir "the journal" galimybėmis. Pavyzdžiui, populiarios Ubuntu ir Linux Mint OS vietoje systemd naudoja Upstart sistemą. UNIX sistemos, nenaudojančios Linux branduolio išvis neturės galimybės naudotis „journal“ teikiamais privalumais.
- Systemd žinučių formatas yra ne atviro teksto failai (ASCII), o dvejetainio formato, kuris nėra gerai dokumentuotas. Tai reiškia, kad parašyti papildomas priemones journal serviso žinutėms perskaityti yra gana sunku, ypač kalbant ne apie C ar C++ programuotojus, o apie sistemų administratorius, kurių programavimo žinios dažniausiai apsiriboja kelių interpretuojamų programavimo kalbų žinojimu, tokių kaip Python, Perl, PHP, Rubby, bash.
- Tinklo maršrutizatoriai, komutatoriai ir kita įranga, kuri moka siųsti žinutes būtent syslog (ne systemd) serveriams ir pasitiki šio formato ir standarto stabilumu, negalės komutuoti su kardinaliai pasikeitusia įvykiu registravimo sistema ir pranešimų formatu. Ateityje dėl suderinamumo ši ir panaši tinklo įranga galbūt palaikys abu formatus, tačiau systemd ateitis nėra garantuota, o syslog jau egzistuoja daugybę metu. Net jeigu tinklo įranga ateityje gebės komunikuoti su systemd įvykių registravimo sistema - saugumo reikia jau dabar.

3.2. Atakos ir grėsmės

Norint apsisaugoti iš pradžių būtina žinoti nuo ko, todėl reikia nustatyti galimas grėsmes ir atakas. Įvykių atakos nukreiptos prieš įvykių registravimo sistemas ir audito žurnalus gali būti skirstomos į tokias kategorijas [8]:

- *Trynimas* – vieno įrašo ar viso bloko sunaikinimas dažniausiai naudojamas kompromituojančiai informacijai apie įsilaužimą, ar kitą neteisėtą veiklą, pašalinti.
- *Nukirtimas* – vietoje vieno įrašo ar bloko esančio bet kurioje įvykių žurnalo vietoje yra pašalinami visi įrašai pradedant nuo įsilaužimo momento (ar pasirinkto bet kurio) iki pat audito žurnalo pabaigos.
- *Įvykių žurnalo sukeitimas* – jeigu įmanoma nuskaityti įvykių žurnalą, kurio vientisumas yra tikrinamas pasirašymo raktais, tai galima sukurti naują žurnalą su kitais pasirašymo raktais ir sukeisti vietomis su originaliu žurnalu.
- *Rašymas* – įvykių žurnalo eilučių pakeitimas, įterpimas ar kitoks klastojimas gali būti panaudotas įkalčiams sunaikinti, kompromituoti nekaltą subjektą, suklaidinti analizės įrankius ar ekspertizę atliekantį asmenį.
- *Atsisakymo aptarnauti ataka* – registravimo sistemos resursų išnaudojimas siekiant sutrikdyti normalią darbo eigą tam, kad būtų pristabdomas ir/arba užkertamas kelias įvykių registravimui.
- *Užtvindymas* – įvykių registravimo sistemos užtvindymas įprasta, galiojančia, tačiau beverte informacija, siekiant maskuoti kenkėjiškus veiksmus.
- *Pasitikėjimo išnaudojimas* – patikimas naudotojas gali išnaudoti turimas privilegijas, kad apeitų apsaugos mechanizmus ir gautų neautorizuotą prieigą prie sistemos registravimo žurnalo ir jos turinio.
- *Nepaprastieji įvykiai* – kritinės sistemos ar jos dalies sunaikinimas siekiant padaryti sistemą ar servisą neveiksniu ir taip padaryti ekspertizę neįmanomą.
- *Skaitymas* – neautorizuota prieiga prie jautrios informacijos. Šis atakos tipas sąlygoja konfidencialios informacijos vagystes. Pavyzdžiui, bankinėse sistemose tai gali būti transakcijų informacijos vagystės.
- Eilės perpildymas – tai atakos tipas, tiesiogiai susijęs su Atsisakymo aptarnauti ataka, kai yra išvaloma perpildyta įvykių eilė. Pavyzdžiui, įsibrovėlis, žinodamas, kad sistemoje yra naudojamas lėtas audito žurnalų generavimo metodas, gali pradėti, specialiai, siųsti į sistemą tūkstančius nepavykusių prisijungimų prie sistemos ir taip didinti įvykių eilę, kurie vis dar laukia, kol juos apdoros ir apsaugos audito žurnalų generavimo metodas. Kadangi naudojant lėtą algoritmą susidarys nemaža eilė, tai realūs sistemos sukompromitavimo įkalčiai dar kurį laiką bus neapdoroti ir lauks eilėje sudarydami būsimam įsibrovėliui saugų „atakos langą“, per kurį jis galės patekti į sistemą ir pašalinti eilėje laukiančius realius sistemos atakos įkalčius.

3.3. Esamų metodų, apsaugai nuo duomenų klastojimo, analizė

Audito žurnalų (įvykių registravimo sistemos žurnalų) saugumo tema yra nagrinėjama gana senai. Sprendžiant įvairias, svarbias, kriptografijos problemas, buvo sukurta ir daugiau metodų, kurie vėliau taip pat buvo pritaikyti audito žurnalų apsaugai, pavyzdžiui, „Maišos grandinės“ (angl. „Hash chains“).

Pirmoji, ir, turbūt, svarbiausia savybė, kurią turėtų turėti saugi įvykių registravimo sistemų savybė yra „*klastojimo akivaizdumas*“ (angl. „tamper evidence“)[8]. Tokią savybę turinti sistema turi metodą, kuris patikimai nustato ir atskleidžia bet kokį duomenų klastojimą arba bandymą klastoti duomenis. Žinoma, tokia sistema pati tiesiogiai nesaugo nuo pačio duomenų klastojimo, tačiau saugios įvykių žurnalų audito sistemos atsiduria būtent šioje kategorijoje.

Kita savybė yra „*atsparumas klastojimui*“ (angl. Tamper resistance)[8]. Klastojimui atspari sistema turi mechanizmą(-us), galintį sulaikyti įvairias atakas prieš užregistruotą informaciją. Jeigu „klastojimo akivaizdumas“ yra dvejetainė savybė – savybę turi arba ne, tai „atsparumas klastojimui“

geriausiai apibūdinamas kaip eilė savybių tokių kaip atsparumas neautorizuotam informacijos skaitymui, kelio užkirtimas duomenų ir failų ištrynimui ir t.t. „Atsparumas klastojimui“ gali būti įvairių atsparumo lygių, o „atsparumo klastojimui“ ir „klastojimo akivaizdumo“ ypatybės dažniausiai eina kartu, nes sistema, kuri gali nustatyti klastojimo faktą, iš dalies, psichologiškai veikdama įsilaužėlį, užkerta kelią ir pačiam klastojimui.

Ideali įvykių registravimo ir audito sistema patektų į trečią kategoriją, tai yra „*nesuklastojamumas*“[8]. Šią savybę turinti įvykių registravimo ir audito sistema sistema pasižymi visišku nesuklastojamumu, tai sistema, kuri visiškai nepalieka jokios galimybės suklastoti įvykių audito žurnalų. Tai yra pati siekiamiausia kategorija.

Teoriškai bet kuri sistema, pasižyminti visišku *nesuklastojamumu*, iš esmės yra ir *klastojimui atspari*, tačiau tik teoriškai, nes sistemos puolėjui gavus fizinę prieigą prie sistemos ji visa gali būti sunaikinta, todėl atsparumas nėra šimtaprocentinis.

Pagrindinės šios srities metodų vertinimo charakteristikos yra tokios:

- Resursų sąnaudos struktūrai (eilei, medžiui ir t.t.) sudaryti (angl. „setup cost“);
- Resursų sąnaudos tam tikro struktūros taško pasiekimui (angl. „traversal cost“);
- Resursų sąnaudos reikalingas auditui atlikti (žurnalui patikrinti) (angl. „verify cost“);
- Talpyklos (disko, atminties ar kitos talpyklos) sąnaudos (angl. „storage cost“);
- Komunikacijos resursų sąnaudos (tinklo sistemoms) (angl. „communication cost“).

Resursų sąnaudos struktūrai sudaryti – tai sąnaudos skirtos patiems audito duomenims sudaryti, pavyzdžiui, grandinei ar medžiui. Tai gali būti tarpinių rezultatų paskaičiavimas reikalingas esamos audito žurnalo eilutės apdorojimui, maišos paskaičiavimas, parašo generavimas ar duomenų pasirašymas, visos struktūros kažkokių reikšmių perskaičiavimas ir pan.

Resursų sąnaudos tam tikro struktūros taško pasiekimui – tai sąnaudos, kurių reikia norint audito žurnalo struktūroje pasiekti tam tikrą įrašą (ar „perbėgti“ per visus įrašus), kurį norima audituoti. Į šią charakteristiką nėra įskaičiuojamas pačio įvykio audito sąnaudos. Resursų sąnaudos gali būti įvairios, pavyzdžiui, kaip rašoma metodų analitinėje dalyje, O(n) maišos grandinėms (jeigu nėra naudojama peršokimų), ir $O(\log_2 n)$ merklio ir istorijos medžiams.

Resursų sąnaudos reikalingas auditui atlikti (žurnalui patikrinti) – tai sąnaudos reikalingos patikrinti ar audito žurnale pasiektas įvykis (ankstesnis punktas) yra teisingas ar ne. Tai gali būti įvykio iššifravimas, tarpinių rezultatų paskaičiavimas, parašo patikrinimas ir pan.

Talpyklos (disko, atminties ar kitos talpyklos) sąnaudos – tai sąnaudos skirtos patiems audito duomenims, tarpiniams ar nuolatiniams, saugoti. Tai gali būti privataus-viešo parašo sudaromas papildomas duomenų kiekis, maišos rezultatai, saugomi kartu su pačiais audito duomenimis ir t.t. Tačiau resursų sąnaudos lyginant metodus gali ne tik didėti, tačiau ir mažėti, pavyzdžiui, vietoje paprasto teksto failų naudojant dvejetainius failus, kodavimą ar kompresiją.

Komunikacijos resursų sąnaudos (sistemoms veikiančioms tinklu) – tai sąnaudos reikalingos audito duomenims perduoti tinklu. Tai gali būti įvykio, maišos rezultato, parašo ar visos struktūros perdavimas tinklu. Jeigu metodas yra kompleksiškas ir reikalaujantis didesnės infrastruktūros, tai sąnaudos gali didėti esant poreikiui tuos pačius duomenis perduoti tarp kelių sistemų ir pan.

Saugus įvykių registravimo žurnalų įrašų perdavimas tinklu tarp nutolusių sistemų ir šių įrašų tikrinimas yra plačiai išnagrinėtas įvairiuose šaltiniuose. Protokolą, apsaugantį tinklu perduodamus įvykių žurnalų duomenis tinklu, pirmieji pristatė [15], vėliau šį protokolą kaip standartą patvirtino IETF [11,14]. Tinklu sujungtoms sistemos taip pat reikia garantuoti įvairiapusę tinklo apsaugą, tiek fiziniame, tiek programiniame lygmenyje, o tai kelia papildomus reikalavimus ir papildomas saugumo problemas. Tinkle esančios sistemos gali pasinaudoti galimybe siųsti įvykių žinutes tiek į fiziškai atskirtas, tiek geografiškai nutolusias, centralizuotas įvykių registravimo sistemas, tačiau reikia pasirūpinti ir šių sistemų saugumu. Nors tokios sistemos be įvykius priimančio serviso, dažniausiai, neturi kitų prieinamų servisų, tačiau gali būti lengvai pažeidžiamos dėl, pavyzdžiui, operacinių sistemų lygio nuotolinio pobūdžio spragų.

Net jeigu darytume prielaidą, kad registravimo sistema neturi jokių pažeidžiamumų jokiame (tinklo, aplikacijų, OS) lygmenyje, tai nėra garantija, kad sistema yra 100% saugi. Nors audito žurnalai ir būtų kaupiami tariamai saugioje, nutolusioje, atskiroje sistemoje, tačiau tai negarantuoja ir pačių audito žurnalų tikrumo – *galima tik daryti prielaidą*, kad audito žurnalai nėra praradę vientisumo, kad jie nebuvo pakeisti ir *tikėtis*, kad informacija juose yra saugi.

3.3.1. Maišos grandinės

Iš pradžių maišos grandinių (angl. „Hash chain“) metodas buvo sukurtas kaip įrankis išspręsti įvairių praktinių ir vertingų kriptografijos taikymo sričių, pavyzdžiui [16,17], našumo problemas [18]. Vėliau maišos grandinių (ir išvestiniai metodai) tapo vienu iš pagrindinių metodų sprendžiant audito žurnalų vientisumo problemas ir kitas svarbias saugumo problemas. Vienas anksčiausių maišos grandinių taikymo pavyzdžių yra Lamporto slaptažodžių autentifikavimo schema [19]. Imkime tokį scenarijų – vartotojas nori, naudodamas slaptažodį, prisijungti prie nutolusios sistemos, tačiau slaptažodį reikia perduoti nesaugiu kanalu. Tam, kad apsisaugoti nuo slaptažodžio perėmimo, o vėliau ir jo panaudojimo, vartotojas gali turėti rinkinį tarpusavyje susijusių slaptažodžių {v0,v1,...,vn}, kurie yra saugomi jo kompiuteryje, o vos tik perdavus slaptažodį iš šio rinkinio tinklu jis (slaptažodis) automatiškai tampa nebegaliojantis.

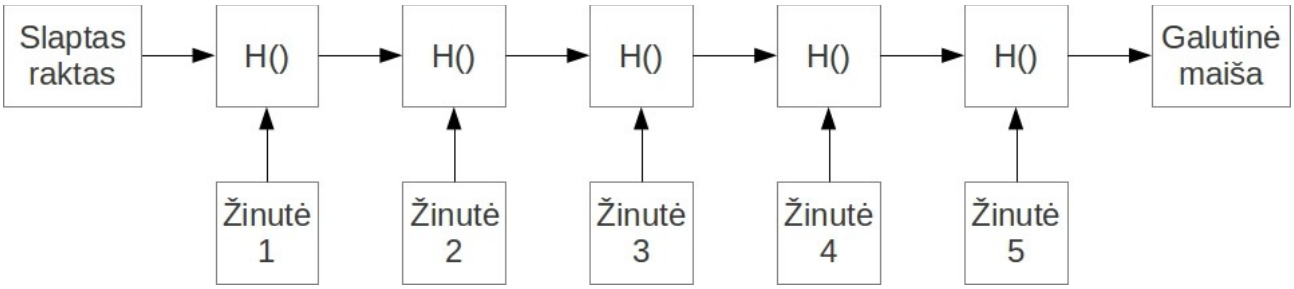
Maišos grandinių metodas naudoja vienkryptes maišos funkcijas (būtent iš čia kyla ir pavadinimas), tokias kaip SHA-256, MD5 ir t.t., kurių paskirtis šiame metode yra garantuoti, kad nebus įmanoma (arba įmanoma tik per labai ilgą laiko tarpą) atrasti ir perskaičiuoti praeities rezultatų, o tuo pačiu nustatyti pirmąją grandinės maišos reikšmę ir perskaičiuoti pilną grandinę arba atlikti kitokias atakas (vėliau, dėl įvairių priežasčių, buvo sukurta ir daugiau maišos grandinių variacijų, kurios yra greitesnės, tačiau kriptografiškai silpnesnės ir atvirkščiai).

Vienkryptė maišos grandinė C yra rinkinys tarpusavyje susijusių reikšmių (v_0 … v_n), kuriame kiekviena V_i reikšmė (išskyrus paskutinę reikšmę V_n) yra šios reikšmės ir prieš tai buvusios reikšmės sąjungos vienkryptė funkcija. Maišos grandinės formulę galima užrašyti taip:

$$i \in [1,n],\ n \in Z,\ V_i = H(V_i\text{-}1)$$

Čia H yra vienkryptė maišos funkcija, $i \in [1,n]$, o v_0 yra pradinė atsitiktinai sugeneruota reikšmė perduodama grandinės pradžiai. Žinoma, v_0 reikšmė turėtų būti saugiai saugoma.

Taigi maišos grandinės generavimo pradžioje yra sugeneruojamas slaptas pradinis raktas v_0, po to kiekviena sekanti grandinės reikšmė yra skaičiuojama išvedant maišos rezultatą iš praėjusios reikšmės, taip sudarant tarpusavyje susijusių įrašų (duomenų) grandinę.



Pav. 7: Maišos grandinės sudarymo schema

Maišos grandinės ilgis yra matuojamas vienkryptės maišos funkcijos vykdymų skaičiumi, taigi maišos grandinės (v_0 … v_n) ilgis yra n.

Maišos grandinės korektiškumas gali būti nustatytas perskaičiuojant visą grandinę nuo v_0 iki v_n ir

sulyginti grandinės galutines reikšmes. Jeigu metode yra naudojami peršokimai, tai grandinę gali reikėti perskaičiuoti $\{v_{i1} \dots v_{i2}\}$ reikšmes ir sulyginti rezultatus.

Viena didžiausių maišos grandinių problemų yra tai, kad grandinės patikrinimo greitis yra O(n), t. y., grandinės patikrinimo greitis yra tiesiogiai proporcingas grandinės ilgiui, o norint patikrinti tam tikrą grandinės reikšmę reikia perskaičiuoti visą grandinę nuo v_0 iki v_i. Dėl šios priežasties, norint sutrumpinti audito laiką, maišos grandinių metodai yra tobulinami, sukuriant įvairias variacijas. Šias variacijas galima skirstyti į tris kategorijas:

- *Be peršokimų (angl. „skipping“)* – tai yra maišos grandinių variacijos, kurios kaip ir standartinė maišos grandinė visoms operacijoms turi O(n) skaičiavimų sąnaudas, o greitesniam tolimesnių grandinės taškų pasiekimui nenaudoja jokių papildomų priemonių. Pavyzdžiui, anksčiau minėta Lamporto slaptažodžių autentifikavimo schema [19];
- *Grandinės kuriose peršokimai naudojami retai* – tai yra maišos grandinės metodo variacijos, kurios naudoja tam tikrus metodus, kad atliekant grandinės auditą būtų galima peršokti kelias grandis ir taip greičiau pasiekti reikiamas grandinės dalis. Tokios variacijos gali naudoti tarpinių reikšmių, sudarant grandinę, išsaugojimą ir pan., pavyzdžiui, [20];
- *Grandinės, kuriose peršokimai yra naudojami dažnai* – tai maišos grandinių variacijos, kuriose peršokimai yra viena pagrindinių savybių. Pavyzdžiui [21] arba [14] minima daugiasluoksnė grandinė (angl. „sandwich chain“);

PRIVALUMAI

Didelis skaičiavimų greitis (priklausomai nuo vienkryptės maišos funkcijos) tiek sudarant grandinę, tiek tikrinant. Tiek struktūriškai, tiek skaičiavimų prasme paprastas metodas, be to gana lankstus (galima naudoti įvairias ypatybes, pavyzdžiui, peršokimus).

TRŪKUMAI

Norint apsisaugoti nuo nukirtimo atakų, reikia kiekvieną maišos reikšmę saugiai laikyti kitoje, saugioje, sistemoje. Kitu atveju prieš maišos grandinę panaudojus nukirtimo ataką jos nepavyks nustatyti. Norint patikrinti audito žurnalo įvykį reikia atlikti tiek maišos operacijų, kiek toli įrašas yra nutolęs nuo grandinės pradžios (nebent yra naudojama grandinė su peršokimais).

3.3.2. Maišos grandinė su peršokimais

Kaip jau buvo minėta, viena didžiausių problemų sprendžiant įvykių žurnalo audito problemą yra audito atlikimo greitis ir efektyvumas. Dar viena problema yra perteklinis duomenų kiekis reikalingas informacijai patikrinti. Įprastų maišos grandinių atveju norit greitai patikrinti *bet kurį įrašą* reikia saugoti visas maišos reikšmes. Čia yra aprašoma viena galimų maišos grandinių variacijų su peršokimais (angl „skipping“). Šiuo atveju generuojant audito duomenis yra saugomi tarpiniai rezultatai, kurie atliekant auditą leidžia pereiti iškart arčiau norimos audituoti informacijos. Norint tai pasiekti galima su audito duomenimis saugoti:

- pirmos ir paskutinės eilutės kontrolines sumas;
- tarpines kontrolines sumas (tarpinių kontrolinių sumų dažnumas gali būti konfigūruojamas), kuriomis naudojantis būtų sudaromos įvykių epochos.

Tarpinių maišos reikšmių išsaugojimas gali būti tiek statinis, pavyzdžiui, kas 10 ar 100 įvykių, tiek dinaminis, priklausomai nuo įvykių kiekio, jų dažnio arba maišos rezultatai gali būti išsaugomi

tam tikru laiko intervalu.

Pavyzdžiui, jeigu failas yra 1000 tūkst. įvykių (eilučių) dydžio, o tarpinių kontrolinių sumų intervalas (epocha) yra 100 įvykių, o maišos rezultato (H) dydis 32 baitai, tai audito duomenų dydis X yra:

$$X = H(\text{grandinės\_pradžia}) + H(\text{grandinės\_pabaiga}) + H()*(\text{įvykiu\_kiekis} / \text{epochos\_intervalas});$$
$$X = 32B + 32B + H()*10 = 384B$$

Čia: H() - maišos funkcija, X – rezultato dydis bitais.

Palyginimui, saugant visų įrašų maišos funkcijų rezultatus, jeigu įrašų kiekis būtų 100tūkst., tai papildomi audito duomenys X užimtų:

$$X = H(\text{įvykių\_kiekis})$$
$$X=H()*1000 = 32000B$$

Čia: H() - maišos funkcija, X – rezultato dydis bitais.

Šiuo atveju meta-duomenų dydžio skirtumas yra ~83.333 . Žinoma, pasirinkus dažnesnį tarpinių sumų intervalą šis skirtumas mažėja, tačiau netgi pasirinkus tarpinius rezultatus saugoti kas 10 įvykių, audito duomenys užimtų beveik 100 kartų mažiau vietos.

Šiuo atveju auditas yra atliekamas skaičiuojant atskirų epochų maišos funkcijų rezultatus ir lyginant su nutolusiame serveryje saugoma (nėra „išankstinio saugumo“ apie kurį rašoma vėliau analizės dalyje) epochos pabaigos maišos funkcijos rezultato reikšme. Epochų eilė $<e_0, e_1, \ldots, e_n>$ leidžia susiaurinti ir pagreitinti paiešką, nes galima greitai patikrinti ne viso failo vientisumą, o tam tikros epochos.

3.3.3. Lengvosios maišos grandinės

Lengvosios grandinės (angl. „light chain“) sprendžia problemą kai reikia pasiekti kuo mažesnes duomenų perdavimo ir duomenų talpyklos sąnaudas. Pavyzdžiui, sensorių tinkluose.

Lengva vienkryptė maišos grandinė C yra rinkinys tarpusavyje susijusių reikšmių ($v_0 \ldots v_n$), kuriame kiekviena v_i reikšmė (išskyrus paskutinę reikšmę v_n) yra šios reikšmės ir prieš tai buvusios reikšmės sąjungos vienkryptės maišos funkcijos rezultatas su trumpa išvestimi: $v_i = H(v_{i-1})$. Tai reiškia, kad lengvoji grandinė yra tiesiog įprastos maišos grandinės variacija. Jeigu įprastame maišos grandinės metode maišos funkcijos rezultatas, dažniausiai, yra nuo 128 iki 160 bitų ilgio (priklausomai nuo maišos funkcijos), tai lengvųjų grandinių rezultatas būna daug trumpesnis, pavyzdžiui 32 arba 64 bitai. Šiame metode H yra „padruskuota“ (angl. „salted“) maišos funkcija, kurios išvestis būna sutrumpinama. Tai yra, gali būti naudojamos tos pačios maišos funkcijos (SHA1, SHA256, MD5), tačiau po rezultato gavimo pats maišos rezultatas yra nukerpamas iki tam tikro bitų kiekio.

PRIVALUMAI

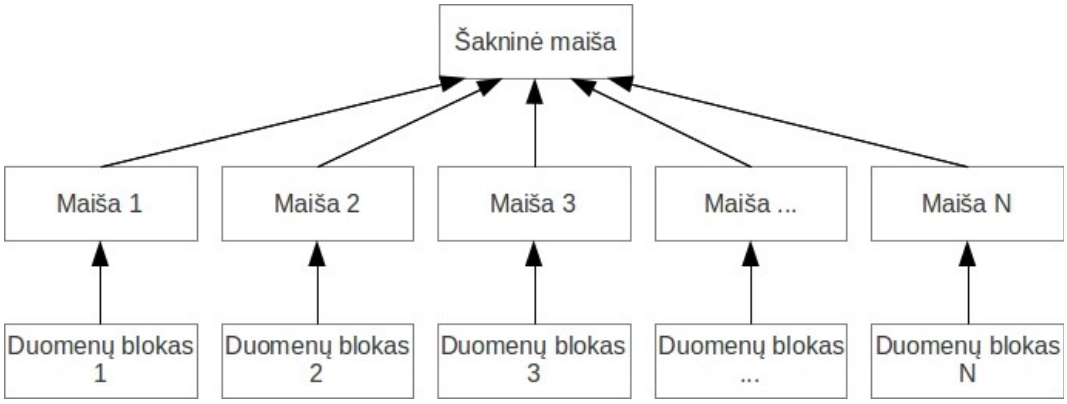
Žymiai mažesni reikalavimai duomenų kiekiui., o jeigu sistema veikia tinklu, tai ir žymiai mažesnės tinklo resursų sąnaudos.

TRŪKUMAI

Mažėjant maišos ilgiui, mažėja ir atsparumas pirmo ir antro tipo piešvaizdžio (angl. „pre-image“) atakoms, bei kolizijoms. Grandinės pradžios atsitiktinį raktą (angl. „seed“), jeigu toks yra naudojamas ir naujausią grandinės pabaigos maišos reikšmę būtina saugoti nutolusiame, saugiame, serveryje.

3.3.4. Merklio medžiai

Poreikis greičiau patikrinti maišos grandines privedė prie metodo, vadinamo maišos medžiu (angl. „hash tree“) arba pagal metodo autoriaus pavardę – Merklio medžiu (angl. „Merkle tree“), sukūrimo [22]. Merklio medžių metodas yra dviejų algoritmų – maišos grandinių ir maišos sąrašų (angl. „hash list“) (žr. Pav. 8), apibendrinimas/susiejimas [8].

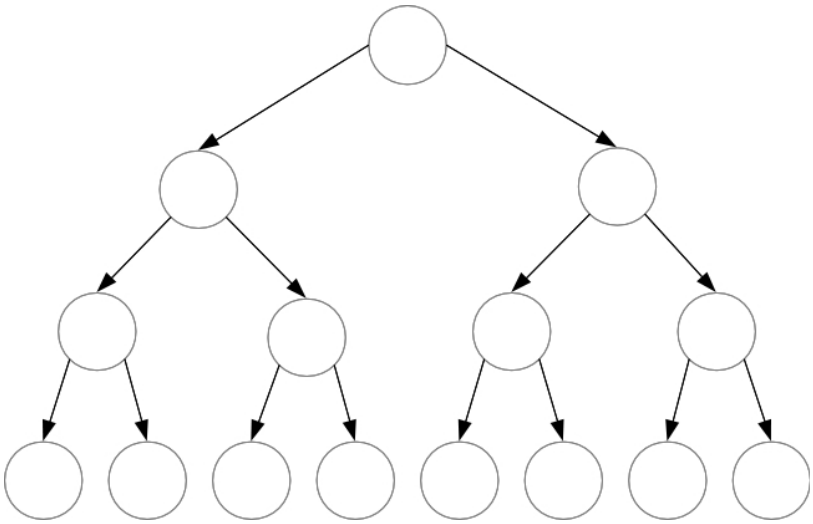


Pav. 8: Maišos sąrašas

Kriptografijoje ir kompiuterių moksle maišos medis arba Merklio medis (žr. Pav. 9) , dažniausiai, yra dvejetainė medžio struktūra t. y. du jaunesnieji mazgai po kiekvienu vyresniu mazgu, tačiau galima naudoti ir didesnį kiekį mazgų, pavyzdžiui tris. Kiekvienas mazgas, išskyrus „lapus“, yra pažymimas jaunesnių mazgų maišos rezultatu.

Maišos medžiai yra naudingi, nes leidžia greitai ir saugiai patikrinti dideles duomenų struktūras. Kelias iš bet kurio taško (lapo) iki šakninio, viešai publikuojamo, maišos rezultato iš tiesinės skaičiavimo priklausomybės O(n), kuri būdinga maišos grandinėms, sumažėja iki logaritmiško ilgio $O(\log_2 n)$.

Dažniausiai merklio medžiams yra naudojamos tokios vienkryptės maišos funkcijos kaip Whirpool, Tiger, SHA-1, SHA-256, tačiau galima naudoti bet kokią funkciją, pavyzdžiui, jeigu reikia ne apsaugoti audito žurnalų vientisumą, o tik apsisaugoti nuo netyčinio sugadinimo (perduodant duomenis tinklu ar kitaip), tai galima naudoti žymiai silpnesnį saugumo atžvilgiu metodą, pavyzdžiui, CRC.



Pav. 9: Merklio (maišos) medis

Pagrindinis merklio medžio ir maišos sąrašo skirtumas yra tai, kad merklio medį galima parsisiųsti ir patikrinti dalimis, t. y. šakomis, dar neturint viso failo. Tuo tarpu norint patikrinti maišos sąrašu apsaugotus duomenis reikia parsisiųsti visus duomenis ir tik po to galima patikrinti.

Norint įrodyti, kad duomenų blokas (medžio lapas), pavyzdžiui, audito žurnalo eilutė, yra nesugadintas ir tikrai priklauso tam tikram duomenų rinkiniui, reikia perskaičiuoti to duomenų bloko maišos rezultatus iki pat šakinės medžio maišos ir sulyginti rezultatus. Deja, tam kad patikrinti ar tam tikras duomenų blokas tikrai yra autentiškas ir yra to medžio dalis auditui reikia pateikti logarimiškai didėjantį kiekį duomenų kiekį kuris yra proporcingas visam medžio mazgų sumos logaritmui [23] (palyginimui, tikrinant maišos eilę šis duomenų kiekis yra proporcingas mazgų skaičiui, kadangi norint patikrinti reikia turėti visus duomenis).

Taigi, naudojant Merklio medį, dvi maišos grandinių problemos – struktūrinis paprastumas ir žemas komunikacijos kompleksiškumas, dingsta ir yra išmainomos į didesnį atsitiktinių įvykių audito greitį [23]. Reikia atkreipti dėmesį, kad Merklio medžio greitis daugumoje randamų šaltinių yra akcentuojamas remiantis skaičiavimų kompleksiškumo formule $O(\log_2 n)$, kas yra tiesa, tačiau beveik niekada nėra minima, kad šis audito logaritmiškumas pasižymi tik tikrinant *atsitiktinai pasirinktus duomenų blokus (lapus)*, tai yra jeigu audito žurnale tikriname įvykius tokia, atsitiktine, tvarka: 3, 8 ,53 ,1239 ,42, 9... arba jeigu žinome kur pažeidimas gali būti arba norime Merklio medyje rasti kažką konkretaus, o nuosekliam visų duomenų blokų patikrinimui tokia skaičiavimų kompleksiškumo formulė, Merklio medžių atveju, nėra teisinga.

Imkime realią, gyvenimišką, situaciją - norime rasti knygoje visas klaidas, tačiau nežinome ar knygoje yra klaidų ar ne, taip pat negalime žinoti ir kurioje knygos vietoje klaida gali būti. Tokiu atveju knygą skaitysime nuo pradžios iki pabaigos, nepraleisdami jokių žodžių ir turėsime patikrinti kiekvieną žodį paeiliui, tai reiškia paieškos kompleksiškumas yra O(n) (kaip ir maišos grandinių) ir Merklio medžio kompleksiškumas tokiu atveju padidėja. Jeigu vieno *atsitiktinio* duomenų bloko (viename bloke du įvykiai) kompleksiškumas yra $O(\log_2 n)$, tai norint patikrinti visus įvykius iš eilės kompleksiškumas tampa $O((\log_2 n)*(n/2))$ t. y. kiekvienai duomenų porai (nes medyje duomenų blokai (medžio lapai) yra poruojami) reikės atlikti $O(\log_2 n)$ operacijų (žr. Pav. 10).

Pav. 10: Merklio medžio atsitiktinio ir nuoseklaus audito operacijų kiekio, nuo įvykių kiekio audito žurnale, palyginimas

PRIVALUMAI

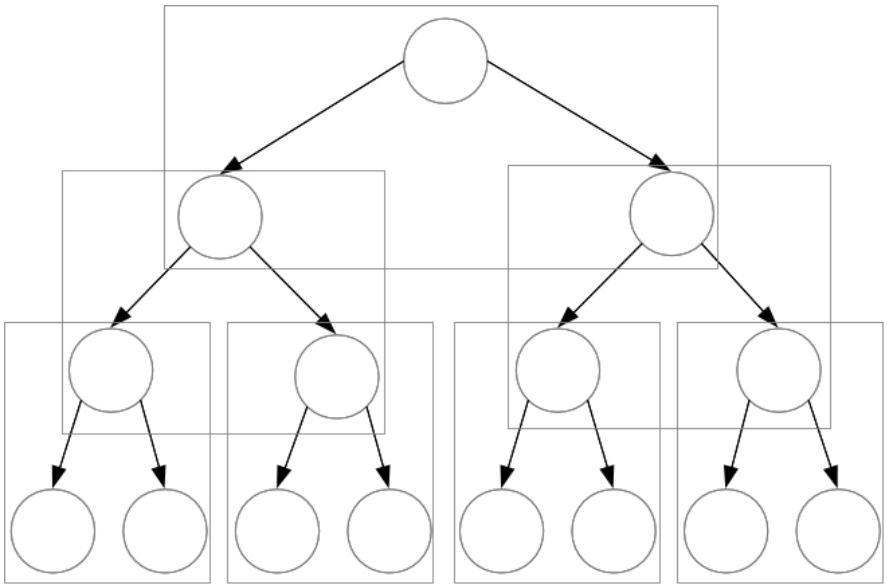
Viso failo auditui atlikti užtenka vos vienos maišos (šakninės). Greitas ($O(\log_2 n)$), bet kurio, atsitiktinio, įvykio patikrinimas.

TRŪKUMAI

Norint patikrinti visą failą nuosekliai (t. y. visus audito žurnalo įvykius, o ne pavienius), audito operacijų kiekis sparčiai didėja - $O((\log_2 n)*(n/2))$.

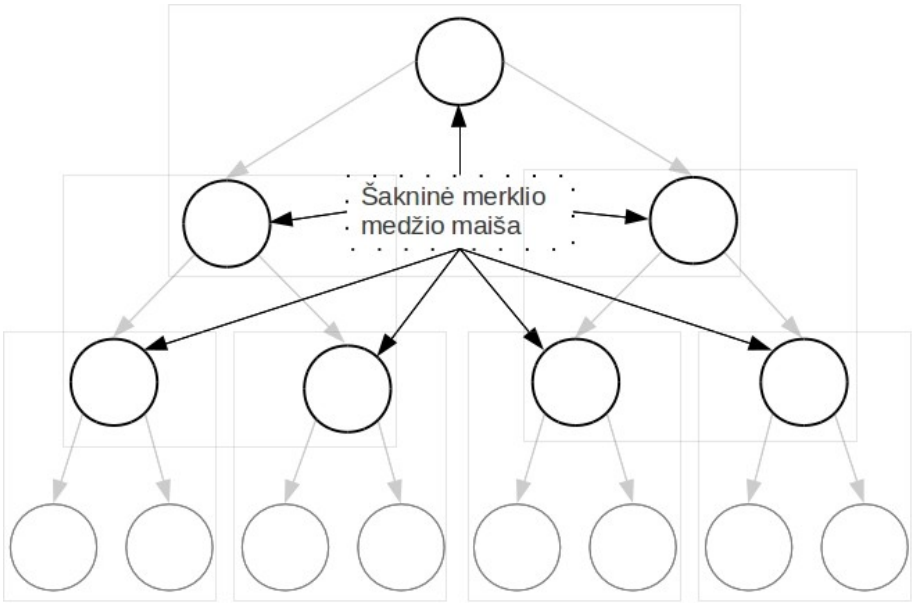
3.3.5. Istorijos medis

Istorijos medis, kaip ir Merklio medis, yra maišos medžio (angl. „hash tree“) struktūra, turinti „klastojimo akivaizdumo“ ypatybę. Istorijos medis yra Merklio medžio variacija. Apie istorijos medžio struktūrą galima galvoti kaip apie medį, sudarytą iš daugybės atskirų Merklio medžių (žr. Pav. 11), todėl, nors tai ir nėra visiškai naujas metodas ar struktūra, tačiau turi įdomių, naujų savybių. Istorijos medis naudoja medžio „versijomis“ pagrįstą maišų skaičiavimą. Toks skaičiavimų metodas gali efektyviai nustatyti ar skirtingos to pačio medžio „versijų“ šakos (angl. „snapshots“), atvaizduotos kaip mažesni Merklio medžiai, ir turinčios *atskiras šaknines maišas*, byloja apie vienodus praeities įvykius.



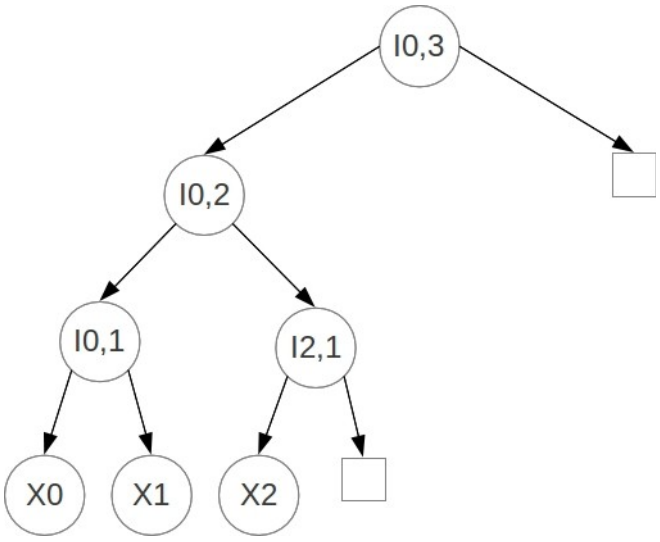
Pav. 11: Istorijos medis sudarytas is mažesnių merklio medžių

Kadangi bet kuris Merklio medis, viso medžio vientisumui patikrinti, turi tik vieną šakninę (angl. „root“) maišos reikšmę medžio apačioje, tai sudėjus visus merklio medžius į vieną bendrą istorijos medį gautume vieną didelį Merklio medį, kuriame kiekvienas vidinis medžio mazgas yra tuo pačiu ir vidinis, ir šakninis (žr. Pav. 12)

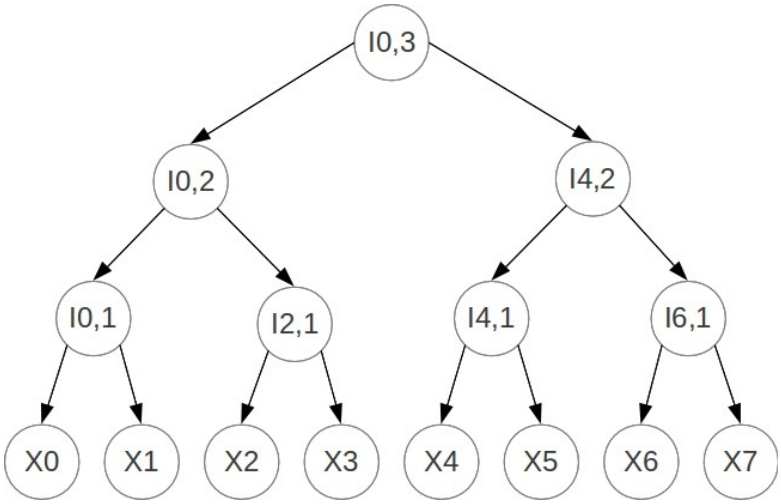


Pav. 12: Kiekvienas istorijos medžio vidinis mazgas yra mažesnio merklio medžio šakninė maiša

Užpildytas *d* gylio istorijos medis, yra dvejetainis Merklio medis, lapuose talpinantis 2^d įvykių. Vidiniai mazgai $I_{i,r}$ yra identifikuojami pagal jų indeksą *i* ir lygį *r*. Kiekvienas lapas $I_{i,0}$ lygyje *0*, saugo įvykį X_i. Vidinis mazgas $I_{i,r}$ turi dvi atžalas: kairę $I_{i,r-1}$ ir dešinę $I_{i+2^{r-1},r-1}$ (šį numeravimo būdą iliustruoja Pav. 13 ir Pav. 14). Jeigu medis nėra pilnas, tai sub-medžiai (angl. „sub-tree“), kurie neturi įvykių yra žymimi kaip □ [14].

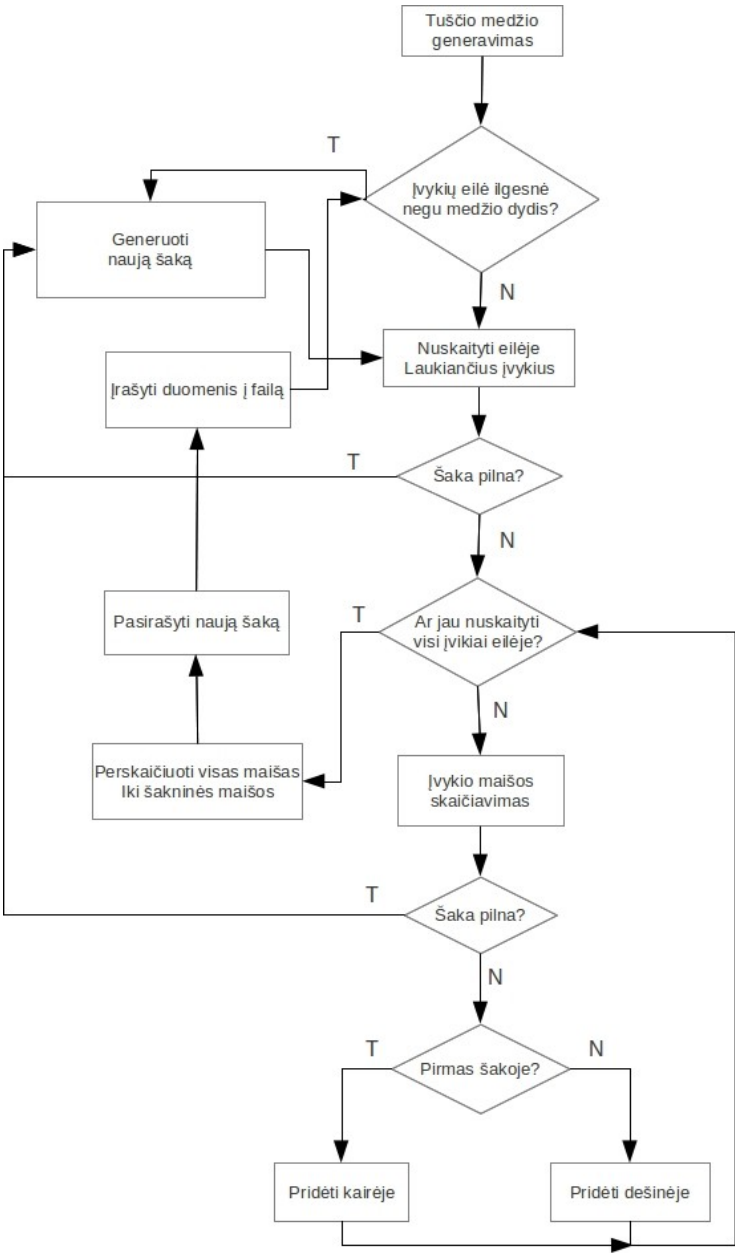


Pav. 13: Nepilnas istorijos medis su žymėjimu. Tuščios šakos pažymėtos tuščiais kvadratais.



Pav. 14: Pilnas istorijos medis. Su lapų ir mazgų žymėjimu.

Kadangi istorijos medį sudaro daug Merklio medžių, tai kiekvienas vidinis istorijos medžio mazgas yra pažymėtas dviejų žemiau esančių mazgų arba lapų maišos suma, t. y., jeigu Merklio medžio atveju auditui turėjome tik vieną maišos reikšmę, apie *160bitų* (priklausomai nuo vienkryptės maišos funkcijos rezultato), tai Istorijos medyje turime tiek audito duomenų, kiek vidinių mazgų yra medyje. Istorijos (ir Merklio) medžio vidinių mazgų skaičius yra lygus N-1, kur N yra įvykių kiekis audito žurnale, todėl audito žurnalas turėtų užimti daugiau talpyklos vietos negu Merklio medžio atveju. Žinoma, būtent dėl šių papildomų duomenų istorijos medis ir tampa lankstesnis. Be to, kadangi visos reikšmės jau yra medyje, tai, teoriškai, turėtų reikėti mažiau skaičiavimų norint atlikti viso įvykių registravimo žurnalo auditą.



Pav. 15: Istorijos medis. Naujų įvykių įterpimas.

Istorijos ir Merklio medžiai struktūriškai turi du skirtumus. Merklio medis yra dvejetainis, todėl visada turėtų turėti dvejetainį įrašų (lapų/duomenų blokų) kiekį (2,4,8,16,32 … n), todėl, jeigu trūksta kažkiek reikšmių iki dvejetainio skaičiaus, tai trūkstamos reikšmės turi būti užpildytos tuščiais laukais. Be to Merklio medis iš esmės yra tinkamas tik statiniams duomenims ir sudarius vieną pilną šaką turi būti „užšaldytas“. Tuo tarpu istorijos medis yra sudarytas iš mažų Merklio medžių (vieno lygio/gylio), todėl įgauna dinamiškumą, nors medis taip pat turėtų būti dvejetainis, tačiau gali turėti ir tuščių šakų (žr. Pav. 13).

Istorijos medžio sudarymas vyksta pagal 15 schemą (žr. Pav. 15). Įvykiai į medį yra įterpinėjami kuriant naujas, tuščias, šakas ir jas palaipsniui pildant duomenimis. Pridėjus kiekvieną naują įvykį turi būti perskaičiuota visos šakos maiša iki pat šakninės maišos, šakninė maiša yra pasirašoma RSA arba DSA viešu raktu (žinoma, galima naudoti ir kitokį viešojo rakto algoritmą, arba nenaudoti išvis). Jeigu viena šaka pilnai užpildoma, tai sukuriama nauja šaka. Ši, nauja, šaka tuo pačiu yra apjungiama su senąja subendrinant šakninę maišą (t. y. du medžiai yra apjungiami viena šaknimi) ir

yra pildoma nauja šaka. Ciklas kartojamas kol įterpimui į medį nebelieka duomenų.

Įvykio patikrinimas vyksta labai panašiai kaip ir Merklio medžio atveju, tik, šioje variacijoje auditoriui perdavus „apgenėtą“ medį ir sulyginus šakninę maišą yra patikrinamas ir RSA/DSA parašas (jeigu toks yra naudojamas).

PRIVALUMAI

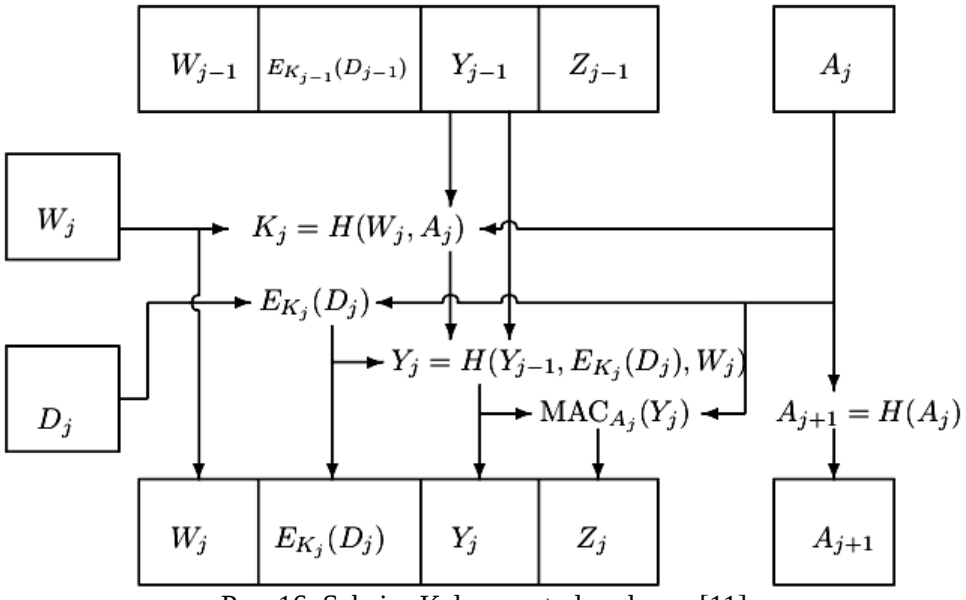
Dinamiškesnis negu Merklio medis. Medyje gali būti tuščių šakų. Medį galima „nugenėti“ skirtingas medžio šakas ir paraleliai perduoti jas skirtingiems auditoriams. Atsitiktinių šakų patikrinimui reikia vos $O(\log_2 n)$ operacijų. Kadangi yra naudojamas viešasis parašas, tai yra apsisaugoma nuo nukirtimo atakų.

TRŪKUMAI

Kitaip negu maišos grandinėse, eilėse ir Merklio medžiuose istorijos medyje yra saugomos visų tarpinių mazgų maišų reikšmės, o tai lemia didesnį audito duomenų kiekį ir didesnes talpyklos sąnaudas. Šias sąnaudas dar labiau padidina parašo naudojimas.

3.3.6. Schneier-Kelsey metodas

Daugumoje realaus pasaulio aplikacijų įvykių žurnalai yra generuojami ir saugomi nepatikimoje įvykių registravimo sistemoje, kuri nėra fiziškai pakankamai apsaugota, kad galėtų būti garantuotas šios sistemos sukompromitavimo neįmanomumas [11]. Būtent šiuo aspektu yra išskirtinis visas Schneier ir Kelsey darbas [11] šia tema. Yra daroma prielaida, kad audito žurnalų kaupimo serveris nėra saugus ir juo negalima pasitikėti. Jų metodas remiasi maišos grandinėmis ir rolėmis paremta saugumo schemą (naudojant prieigos kontrolės sąrašus, leidimų „kaukes“), kurie apriboja šios sistemos naudotojų prieigą tik prie jiems skirtų įrašų. Metode yra naudojami nepatikimi ir pusiau patikimi auditoriai, kurie gali patikrinti tik jiems skirtus duomenis. Tarkime, turime centrinį įvykių žurnalų serverį, jame yra registruojami įvykiai iš skirtingų šaltinių, pagal [11] metodą vienas auditorius gali atlikti vieno šaltinio auditą, kitas auditorius kito šaltinio arba prieiga gali būti skirstoma ir detaliau, pavyzdžiui, pagal įvykių tipus.



Pav. 16: Scheier-Kelsey metodo schema [11]

Saugumas šiame metode yra užtikrinamas tokiais būdais[11]:

1. Nauja autentifikavimo rakto maiša, panaudojant vienkryptę funkciją, yra paskaičiuojama iškart po įvykio įterpimo į failą, o senoji maiša yra neatkuriamai pašalinama.
2. Kiekvieno įrašo užšifravimo raktas yra išvedamas iš to įrašo autentifikavimo rakto, panaudojant vienkryptę funkciją. Tai leidžia išdalinti individualių audito žurnalo įvykių užšifravimo raktus pusiau patikimiems auditoriams, kurie gali iššifruoti ir nuskaityti audito įvykius, tačiau neleidžia šiems auditoriams atlikti jokių nepastebimų pakeitimų.
3. Su kiekvienu audito žurnalo įvykiu yra saugomas maišos elementas, kuris leidžia patikrinti visus prieš tai buvusius įvykius. Būtent ši reikšmė ir yra autentifikuojama audito metu.
4. Kiekvienam audito žurnalo įvykiui yra priskiriama leidimų „kaukė“. Ši leidimų „kaukė“ nustato roles rolėmis paremtoje saugumo schemoje. Pusiau patikimiems auditoriams gali būti suteikiama prieiga tik prie tam tikrų įvykių. Kadangi šifravimo raktai kiekvienam įvykiui yra iš dalies išvedami iš įvykio tipo, tai auditoriui bandant meluoti apie jam suteiktas teises jis tiesiog niekada negaus teisingo šifravimo rakto.

Kaip matyti iš schemos (žr. Pav. 16) šis metodas yra labai sudėtingas ir painus, todėl detalesnės informacijos geriausia ieškoti pačių autorių darbe, kuriame viskas detaliai paaiškinta [11].

PRIVALUMAI

Apsaugo nuo *beveik* visų atakų tipų prieš audito žurnalus. Suteikia galimybę skirtingus įvykius audituoti skirtingiems auditoriams, o tai yra labai naudinga jeigu vienoje sistemoje norime kaupti audito žurnalus iš kelių kitų nepriklausomų šaltinių, pavyzdžiui, skirtingų įmonių, ir auditą atlikti nepriklausomai.

TRŪKUMAI

Scheier ir Kelsey metodas nėra atsparus nukirtimo atakai [7]. Be to metodas yra labai sudėtingas, kompleksiškas ir reikalauja papildomos infrastruktūros – reikia netgi ne vienos nutolusios sistemos auditui, o mažiausiai dviejų – nepatikimos įvykių kaupimo sistemos ir patikimo auditoriaus.

3.3.7. Išankstinis vientisumas

Sistemose, kuriose įvykių registravimo/audito žurnalams saugoti yra naudojami nutolę, patikimi, serveriai turi vieną rimtą problemą. Jeigu nutolęs serveris nėra pasiekiamas, tai visi įvykiai yra laikinai kaupiami ir saugomi lokalioje sistemoje. Tokiu atveju, kai sistema yra sukompromituojama įsibrovėlis, perėmęs dabartinį slaptą raktą, gali nevaržomai keisti lokalioje sistemoje laikinai likusius duomenis. Šioje vietoje svarbi problema tampa „išankstinis vientisumas“ (angl. „forward integrity“): kaip užtikrinti, kad duomenys prieš sistemos sukompromitavimą negalėtų būti klastojami? Tai yra, įsibrovėlis, perėmęs dabartinį slaptą raktą, turi negalėti klastoti audito duomenų, sukauptų prieš sistemos ir esamo rakto sukompromitavimą [7].

Anksčiausios idėjos šioje srityje kyla iš „saugaus laiko štampavimo“ (angl. „secure time stamping“) problemos tyrinėjimų [8]. Šiame metode laiko štampų grandinė buvo nuosekliai sujungiama su kitais laiko štampais, kai tuo tarpu laiko štampo sertifikatas būdavo išvedamas iš prieš tai sugeneruoto laiko štampo [24].

Vėliau šią koncepciją patobulino Bellare ir Yee [1] ir pristatė metodą, kuris, pritaikius jį audito žurnalams, suteikė vientisumo garantijas sistemoje po sukompromitavimo. Autoriai šį metodą

pavadino „Forward Integrity“ (FI). Šis metodas yra patobulinta maišos funkcijos panaudojimo technika, kai kiekvienai įrašo eilutei yra paskaičiuojama maišos funkcija ir saugoma kartu su įrašu. Tik šio metodo atveju yra generuojamas pirminis raktas, kuris yra pritaikomas pirmajai audito eilutei, vėliau šis raktas, kuriuo yra apsaugomi audito įrašai mutuoja ir nauju raktu yra apsaugomi sekantys audito žurnalo įrašai. Struktūriškai įvykių žurnale esantys įrašai yra dalinami į epochas – laiko tarpus, kuriuose naudojami skirtingi, mutuojantys raktai. Būtent apsaugos rakto mutavimas ir lemia šio metodo veiksmingumą – iš esamo rakto galima nustatyti koks bus sekantis raktas, tačiau neįmanoma paskaičiuoti koks raktas buvo prieš tai. Tokio žurnalo vientisumą galima patikrinti tik turint pirminį raktą, iš kurio galima paskaičiuoti bet kurios kitos epochos raktą. Epochos gali mutuoti dėl įvairių veiksnių (priklausomai nuo metodo variacijos), pavyzdžiui, po tam tikro laiko, arba po tam tikro įvykių kiekio. Taigi, net jeigu sistema bus sukompromituota ir sistemos puolėjas turės pilną priėjimą prie sistemos ir audito žurnalų, jis vis tiek negalės pakeisti praėjusių epochų įvykių.

Formaliai metodą galima paaiškinti taip: Iš pradžių yra generuojamas pradinis raktas K0, kuris veikia kaip audito raktas, turi būti *laikomas saugiai* ir prieinamas *tik saugiam auditoriui*. Epochos E_i raktas K_i yra sudaromas iš praėjusios epochos K_{i-1} rakto pritaikant vienkryptę funkciją. Epochos E_i pradžioje raktas K_{i-1} yra saugiai, negrįžtamai ir neatkuriamai (*tai labai svarbu*) pašalinamas. Todėl, jeigu sistema yra sukompromituojama epochos E_i metu, tai jis gali perimti ir raktą K_i, tačiau niekaip negalės atvirkštine tvarka išskaičiuoti praėjusių epochų raktų K_j ($j < i$), taigi sistemos puolėjas negalės suklastoti praeities duomenų.

Bellare and Yee [1] savo darbe nagrinėja ir įvairias šio metodo variacijas, pavyzdžiui, vietoj nuoseklaus rakto mutavimo panaudojant vienkryptę funkciją galima naudoti atsitiktinį mutavimą arba naudoti pseudo-atsitiktinius generatorius.

Atsitiktinis rakto mutavimas

Kai yra naudojamas atsitiktinis rakto mutavimas visas metodas yra analogiškas kaip ir įprasto FI – kiekvienos epochos E_i, …, E_n raktai K_i, …, K_n mutuoja, o prasidedant kiekvienai epochai E_i, yra negrįžtamai pašalinamas raktas K_{i-1}. Tačiau, kadangi raktas mutuoja atsitiktiniu būdu nėra galimybės nustatyti atskirų epochų raktų iš pirminio rakto, todėl kiekvienos epochos atsitiktinį raktą reikia siųsti į atskirą sistemą ir ten juos kaupti, todėl atsitiktinio rakto generavimo išankstinio vientisumo metodas beveik nesiskiria nuo maišos grandinės. Šios variacijos didžiausias trūkumas lyginant su maišos grandine yra tai, kad reikia saugiai kaupti kiekvienos epochos raktus, tuo tarpu maišos grandinėms užtenka kaupti pradinį atsitiktinį raktą (angl. „seed“) ir paskutinį grandinės maišos rezultatą.

Rakto mutavimas panaudojant pseudo-atsitiktines funkcijas

Jeigu atsitiktinio rakto mutavimo variacijoje reikia saugoti kiekvienos epochos raktą, tai rakto mutavimui naudojant pseudo-atsitiktines funkcijas **(**PRF, angl. „pseudo-random function“) užtenka saugoti tik pagrindinį raktą, kuris vėliau gali būti panaudotas paskaičiuoti bet kurios epochos raktą. Pseudo-atsitiktiniai generatoriai šiuo atveju leidžia nuspėjamai generuoti naujus raktus, išvestus iš pradinio rakto, tačiau neįmanoma raktų skaičiuoti atvirkštine tvarka.

Norint apsisaugoti nuo gerai organizuotų ekspertų atakų, FI metodu paremtos audito sistemos turi gebėti kuo greičiau keisti epochas [1], pageidautina po kiekvieno naujo įvykio pridėjimo į žurnalą. Žinoma, tai nėra sunku pasiekti, sunkiausia yra pasiekti, kad pseudo-atsitiktinis generatorius būtų kuo saugesnis ir tuo pačiu perskaičiuotų epochos raktą su kuo mažesnėmis resursų sąnaudomis.

PRIVALUMAI

Įvykių kaupimui nėra reikalinga nutolusi, saugi, sistema jeigu raktų mutavimui yra naudojamas

pseudo-atsitiktinis generatorius (atsitiktinam rakto generavimui netinka). Visai grandinei patikrinti užtenka vos vieno, pagrindinio, rakto, kuris turi būti laikomas saugiai.

TRŪKUMAI

Į sistemą patekęs sistemos puolėjas bandys pašalinti naujausius įrašus, todėl, jeigu epochos keitimo dažnumas yra per mažas, tai puolėjui gali užtekti laiko įsibrauti ir pašalinti įkalčius dar prieš atnaujinant epochos raktą.

Norint patikrinti visą grandinę, reikia atlikti tiek rakto mutacijų iš pradinio rakto, kiek epochų reikia patikrinti. Tai lemia arba didelius skaičiavimo reikalavimus, arba saugumo mažinimą didinant epochų kitimo dažnį. Norint išvengti visų raktų mutacijų perskaičiavimo reikia pseudo generatorių susieti su kažkokia nuolat kintančia reikšme, pavyzdžiui, vidiniu sistemos laikrodžiu, tačiau tai gali pakenkti pseudo generatoriaus patikimumui.

Nėra savaiminio atsparumo nukirtimo atakai – audito žurnalą galima nukirsti ankstesnėje epochoje ir tai nebus pastebėta, nebent paskutinės epochos raktas bus nuolat siunčiamas į nutolusį, saugų, serverį.

3.3.8. Išankstinio saugumo nuoseklus agregavimas

Išankstinio vientisumo (*FI*, angl. „Forward Integrity“) metodo trūkumai pastūmėjo naujo metodo sukūrimą. Daugumoje sistemų ir įrenginių, ypač mažai resursų turinčiuose sensorių tinkluose, mobiliuose įrenginiuose, siekiama taupyti resursus. FI metodas reikalauja kartu su įrašais saugoti ir visų epochų raktus, tuo tarpu Išankstinio saugumo nuoseklaus agregavimo (*FssAgg*, angl. „forward-secure sequential aggregate“) schemoje, įvykiai (žinutės) yra saugomi nepakitę, tačiau autentifikavimo objektas yra tik vienas – agreguotas parašas.

Būtent tokia ir yra šio metodo paskirtis: sumažinti duomenų kiekį reikalingą papildomiems audito žurnalo apsaugos duomenims ir sumažinant komunikacijų kaštus tuo pačiu užkertant kelią slapto rakto sukompromitavimui [25,26].

Pagrindinis šio metodo iššūkis yra efektyvumas: idealiu atveju schema turėtų garantuoti pastovaus dydžio (viešo ir slapto) raktų dydžius, pastovų parašo dydį, taip pat pastovų pasirašymo, patikrinimo ir raktų (viešo ir slapto) atnaujinimo/mutavimo operacijų kiekį [25].

FssAgg autentifikavimo schemoje, išankstinio-saugumo parašai (arba MAC) yra generuojami to pačio pasirašinėtojo ir yra nuosekliai agreguojami į vieną parašą. Sėkmingas agreguoto parašo patikrinimas yra prilyginamas kiekvienos epochos parašo tikrumui. Tuo tarpu nepavykęs agreguoto parašo patikrinimas rodo, kad bent vienas agreguoto parašo komponentas (atskiras parašas) yra neteisingas. Dėl šių priežasčių FssAgg schema yra labai tinkama audituoti žurnalams – atspari nukirtimo atakoms („viskas arba nieko“ audito schema), sąlyginai mažesni talpyklos resursų reikalavimai.

FssAgg schemoje individualūs parašai yra ištrinami vos tik yra apjungiami į parašo agregatą. Visų, iš eilės einančių, individualių, audito žurnalo įvykių tikrumas yra nustatomas perskaičiuojant ir patikrinant visą audito žurnalą su agreguotu parašu. Toks netiesioginis audito žurnalų įrašų patikrinimas yra brangus skaičiavimo resursais tuo atveju, jeigu norima patikrinti vieną, konkretų, audito žurnalo įrašą, o ne visą audito žurnalą.

Toliau metodo analizei yra remiamasi originaliais Ma ir Tsudik [25,26] darbais.

Saugi FssAgg schema turi tenkinti tokias savybes:

1. *Korektiškumas*: Bet kurį parašą agreguotą FssAgg.Asig algoritmu turi suprasti ir priimti FssAgg.Aver (audito) algoritmas.
2. *Nesuklastojamumas*: Nežinant bent vieno (bet kurios epochos) pasirašymo (slapto)

rakto joks sistemos puolėjas negali nustatyti ir perskaičiuoti agreguoto parašo jokiai žinutei ar žinučių rinkiniui.

3. Išankstinis saugumas (angl. „forward-security“): Joks įsibrovėlis sukompromitavęs pasirašymo raktą i negalės sugeneruoti patikimo agreguoto rakto su pasirašyta žinute intervale j < i . Išskyrus einamos epochos agreguotą parašą [25].

Paskutine ypatybe norima pasakyti, kad FssAgg schema yra atspari nukirtimo ir ištrynimo atakoms. Sistemos puolėjas, sukompromitavęs pasirašymo raktą turės dvi galimybes: įterpti sukompromituotą agreguotą parašą į ateities parašus arba ignoruoti sukompromituotą parašą ir pradėti visiškai naujo agreguoto parašo generavimą . Tačiau įsibrovėlis negalės ištrinti pasirinktų įvykių pasirašytų agregavimo raktu sugeneruotu prieš patenkant į sistemą.

Kitaip negu išankstinio vientisumo metodo atveju (žr. „Forward integrity“ analitinėje dalyje), kai įvykių pasirašymas vykdavo epochai keičiantis tam tikru laiko, įvykių ar kitokių vienetų intervalu, FssAgg schemoje yra pabrėžiama, kad parašo mutavimas, pasirašymas, naujo parašo agregavimas turi vykti iškart po naujo įvykio užregistravimo, taip užkertant kelią atakoms pasitelkiant „saugų langą“.

PRIVALUMAI

Užkertamas kelias nukirtimo atakai, kuriai negalėdavo pasipriešinti (be papildomų modifikacijų) kiti metodai, tokie kaip FI ir maišos grandinės.

Išlaikoma „išankstinio vientisumo“ savybė, tačiau sumažinamas audito duomenų kiekis (naudojamas tik vienas, pastovus dydžio parašas).

Sumažinami komunikacijų kaštai sistemose veikiančiose tinklu (lyginant su kitais metodais).

TRŪKUMAI

Norint patikrinti audito žurnalo vientisumą reikia tikrinti visą žurnalą vienu metu. Norint suteikti galimybę patikrinti įvykius mažesniais vienetais, pavyzdžiui, įvykių rinkiniais arba netgi po vieną įvykį, reikėtų kartu su audito žurnalu kaupti ir visus parašus, tačiau tai atvertų kelius įvairioms atakoms, pavyzdžiui, nukirtimo atakai.

3.3.9. Išankstinio saugumo antspaudavimas

Metodas, pavadintas „Išankstinio saugumo antspaudavimas“ (*FSS*, angl. „Forward Secure Sealing“) buvo sukurtas ir pradėtas naudoti 2012 metais Linux OS distribucijoje „Fedora“, yra paremtas FI metodu (savybe) ir naudoja FI metodo variaciją su „išankstinio saugumo pseudo-atsitiktiniais generatoriais“ (FSPRG**,** angl. „forward-secure pseudo-random generators“). Šiame metode naudojamas pseudo-atsitiktinis generatorius yra aprašytas darbe [27]. Vienas iš darbe minimų generatoriaus bendraautorių yra B. Poettering, o programuotojas ir įgyvendintojas Linux OS šeimoje yra šio autoriaus brolis L. Poettering, tačiau, niekur nėra oficialiai dokumentuotas ir paaiškintas pačio metodo veikimas ir algoritmas, išskyrus minėtą pseudo-atsitiktinį generatorių. Dėl šios priežasties ir dėl to, kad tai vienintelis atviro kodo, viešai prieinamas ir oficialioje distribucijoje esantis audito metodas, tolimesniai analizei remiamasi *tik neoficialiais ir viešoje erdvėje prieinamais* autoriaus žodžiais apie patį FSS algoritmą, o informaciją apie FI metodą bei pseudo-atsitiktinių generatorių rolę šiame metode galima paskaityti anksčiau analitinėje dalyje aprašytą FI metodą. Konkrečiai apie FSS metodo naudojamą FSPRG galima rasti jau minėtame darbe [27].

FSS reguliariais laiko intervalais „užantspauduoja“ audito žurnalą [28] (remiantis ankstesne analize, pateikta šiame darbe vadinamasis „antspaudavimas“ yra ne kas kita, kaip įvykių pasirašymas mutavusiu raktu). Todėl, L. Poettering žodžiais, sistemos puolėjas negali klastoti praeities duomenų,

tačiau gali audito žurnalą visiškai pašalinti. Metodas veikia iš pradžių generuojant raktų porą: „antspaudavimo raktas“ ir „audito raktas“. Antspaudavimo raktas yra paliekamas lokalioje sistemoje ir yra reguliariais laiko tarpais atnaujinamas (mutuojamas), o senasis raktas yra saugiai pašalinamas iš sistemos. Tuo tarpu audito raktas turi būti laikomas saugiai nutolusioje mašinoje ar kokiu nors kitu būdu (pavyzdžiui, mobiliajame įrenginyje). Šiuo audito raktu vėliau galima patikrinti visą audito žurnalą. Be to pats audito raktas gali būti panaudotas paskaičiuoti bet kurios epochos antspaudavimo raktą, todėl galima patikrinti bet kurią epochą atskirai.

Remiantis „the journal“ konfigūravimo dokumentacija [29], standartinis rakto mutavimo dažnumas (jeigu generuojant raktus nėra nurodoma kitaip) yra 15 minučių. Tuo tarpu, Bellare ir Yee savo darbe [1] rašo, kad norint apsisaugoti nuo gerai organizuotų atakų, reikia, kad raktas keistųsi kuo dažniau, pageidautina, po kiekvieno įvykio pridėjimo į sistemą. Šiuo (FSS metodo) atveju, jeigu ataka būtų gerai organizuota ir sistemos sukompromitavimas vyktų epochos pradžioje, tai įsibrovėliams galimai pakaktų tokio ilgo laiko tarpo atlikti veiksmus sistemoje ir visiškai pašalinti dar nespėjusios baigtis esamai epochai. Konfigūravimo dokumentacijoje taip pat rašoma, kad intervalą galima ir mažinti, tačiau tokiu didėja CPU apkrovimas. Deja, bet oficialiai skelbiamų matavimų dėl procesoriaus resursų sunaudojimo nėra.

PRIVALUMAI

Autorių teigimu [27] FSS metode naudojamo FSPRG algoritmo saugumas yra didesnis negu kitų.

TRŪKUMAI

Per ilgas rakto mutavimo intervalas. Paveldimi visi FI metodo trūkumai (žr. „forward-integrity“ metodo analizę analitinėje dalyje)

3.4. Skyriaus apibendrinimas

Skyriuje buvo atlikta dviejų šiuo metu plačiausiai GNU/Linux OS šeimoje naudojamų įvykių žurnalų registravimo sistemų – „the journal“ ir „syslog“ analizė, nustatyti šių sistemų privalumai bei trūkumai. „The journal“ dar yra gana jaunas projektas, tačiau jau turi įvykių žurnalų apsaugos ir audito atlikimo galimybę ir naudoja „Forward Secure Sealing“ metodą, kuris yra aprašytas analitinėje dalyje. „Syslog“ ir visos jo atmainos tokios kaip „syslog-ng“ ir „rsyslog“ šio straipsnio rašymo metu neturi jokios įvykių žurnalų vientisumo užtikrinimo ir audito atlikimo galimybės, tačiau dėl GNU/Linux OS lankstumo įvykių registravimo ir audito atlikimo sistema gali tapti bet kuri programa ar servisas, taip pat yra lengva plėsti esamas sistemas, taisyti jų trūkumus, įdiegti saugumo papildinius.

Taip pat šiame skyriuje buvo nustatytos grėsmės, kurios yra arba gali būti nukreiptos prieš audito žurnalus, vėliau apžvelgti metodai, kurie gali apsaugoti nuo šių grėsmių ar jų dalies. Iš analitinės dalies medžiagos, kurioje atliekama įvykių audito ir apsaugos metodų apžvalga, nustatomi metodų privalumai bei trūkumai, galima daryti išvadą, kad nėra vieno universalaus metodo, kuris gebėtų saugoti nuo visų grėsmių. Dažniausiai metodas gali išspręsti tik vieną ar kelias iš gausybės problemų, o kiti metodai, spręsdami ankstesnių metodų trūkumus susiduria su naujomis problemomis, todėl, dažniausiai, yra geriausia įvykių žurnalų apsaugai ir žurnalų auditui naudoti įvairių metodų derinius ir tokių derinių variacijas, tačiau svarbu nepamiršti, kad tokie deriniai gali tapti per daug sudėtingi, kompleksiški ir sunkūs kaip yra su Schneier ir Kelsey metodu, nes vienas svarbiausių charakteristikų žurnalų audito sistemoms yra greitaveika ir paprastumas.

4. PROJEKTO DALIS

4.1. Reikalavimai sistemai

Prieš atliekant įvykių registravimo žurnalų audito tyrimą reikia nustatyti reikalavimus tiriamoms sistemoms (žr. Lentelė 1) ir tyrimo aplinkai (žr. Lentelė 2).

Lentelė 1: Funkciniai ir nefunkciniai reikalavimai tiriamoms audito sistemoms

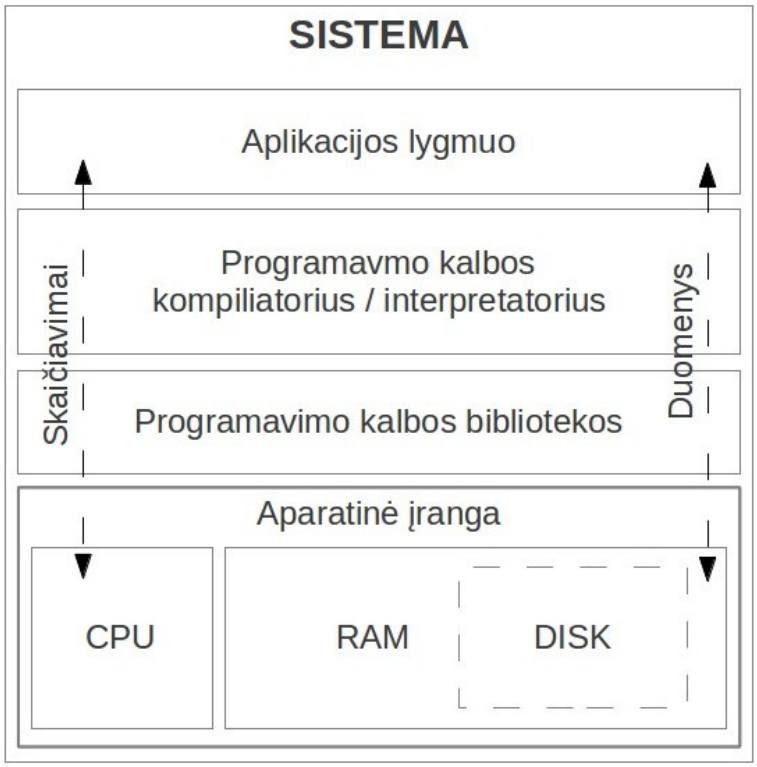
Funkciniai reikalavimai	Nefunkciniai reikalavimai
sugebėti aptikti eilučių įterpimus	Visi audito metodai turi naudoti tą pačią tyrimų aplinką (t. y., ta pati programavimo kalba ir ta pati kodo bazė)
sugebėti aptikti eilučių ištrynimus	
sugebėti aptikti eilučių pakeitimus (vienas simbolis, žodis ar frazė ištrinama arba pridedama arba pakeičiama)	

Lentelė 2: Funkciniai ir nefunkciniai reikalavimai tyrimo aplinkai

Funkciniai reikalavimai	Nefunkciniai reikalavimai
Audito sistemų testavimo ir modeliavimo įrankis turi leisti pasirinkti įvairius parametrus, kuriais dirbs tiriamos audito sistemos	Tyrimo aplinka turi būti sukuriama taip, kad kuo mažiau įtakotų pačių metodų veikimą (pvz., išvengti stabdymo kurioje nors posistemėje)
	Kodo bazė turi būti lengvai perkeliama ir ištestuojama kitoje tyrimo aplinkoje
	Dirbama tik su realiais duomenimis paruoštais būtent šiam tyrimui

4.2. Tyrimo aplinkos modelis

Šiame darbe tiriami metodai yra skaičiavimų tipo, todėl reikia užtikrinti, kad tyrimo metu būtu maksimaliai išnaudotas CPU pajėgumas ir būtų kuo mažesnė pašalinių veiksmų įtaka. Todėl, sėkmingam tyrimui atlikti yra sukurtas darbo aplinkos modelis, kuris, tikėtina, maksimaliai sumažins pašalinių veiksnių įtaką tyrimo rezultatams. Šio modelio schema yra atvaizduota 17 paveikslėlyje.



Pav. 17: Tyrimo aplinkos modelis

Realiomis sąlygomis (ne laboratorijoje) tiriami algoritmai naudos tinklo posistemę įvykių šaltinio ir įvykių žurnalų auditoriaus komunikacijoms. Taip pat, ypač didelėse sistemose, bus daug skaitoma ir rašoma į/iš disko, todėl gali jaustis stipri disko posistemės įtaka. Disko ir tinklo įtaka yra ypač nepageidautina tyrimo metu, nes siekiama išmatuoti būtent skaičiavimų pajėgumus. Tyrimo metu, jeigu tam nebus užkirstas kelias, ypač didelę įtaką darys ir visiškai rezultatus iškreips disko posistemė, nes bus dirbama tiek su mažais, tiek su dideliais failais. Taigi, žinome du pagrindinius šalutinius veiksnius, kurie gali stipriai iškreipti pačio algoritmo rezultatus:

- Disko posistemė
- Tinklo posistemė

Tinklo posistemės įtakai, tyrimo rezultatams, pašalinti, yra naudojama tik viena, lokali, sistema. Tai reiškia jokio tinklo sluoksnio naudojimo.

Disko posistemės įtaka yra pašalinta naudojant vieną ypač patogią šių tyrimų metu Unix/Linux sistemų ypatybę – diską galima simuliuoti operatyvinėje atmintyje (angl. terminas „ramdisk“). Tai reiškia, kad sistema ir aplikacijos dirbs su failais ir direktorijomis tarsi failai būtų diske, o iš tiesų aplikacijos dirba su operatyviąja atmintimi. Kadangi operatyvioji atmintis yra keletą kartų greitesnė už įprastus kietuosius diskus, tai šis veiksnys maksimaliai sumažinamas. Naudodami tokią darbo aplinką rezultatus sukoncentruosime būtent į skaičiavimų pajėgumą, o ne į pašalinius veiksnius.

4.3. Priemonės

Šio darbo modelio realizacijai ir visų pasirinktų algoritmų tyrimams yra panaudotos priemonės pateiktos 3 lentelėje.

Lentelė 3: Tyrimo dalies priemonės

Aparatinė dalis	x86 architektūros PC
Operacinė sistema	Debian 6 (squeeze)
Operacinės sistemos branduolys	Linux 2.6.32
Programavimo kalba	Java 1.6.0
Programavimo aplinka (IDE)	Vi(m)

5. ĮVYKIŲ VIENTISUMO AUDITO METODŲ TYRIMAS

5.1. Tyrimo tipas

Šiame darbe yra naudojamas *taikomojo tyrimo tipas*[2], kuris šio darbo atveju (šio darbo kontekste) yra ir duomenų vientisumo užtikrinimo audito metodų, tyrimų (įskaitant ir įvykių registravimo sistemų) ir žinių bazės panaudojimas, siekiant nustatyti įvykių registravimo sistemos vientisumo audito algoritmų charakteristikas.

Probleminė sritis, teorinė žinių bazė ir pradinė informacija bei audito metodai yra aprašyti įvadinėje ir analitinėje darbo dalyje. Toliau darbe, naudojant specialiai parašytą programinę įrangą, tirsime nustatytas metodų (algoritmų) charakteristikas, pagal kurias lyginsime tiriamus metodus.

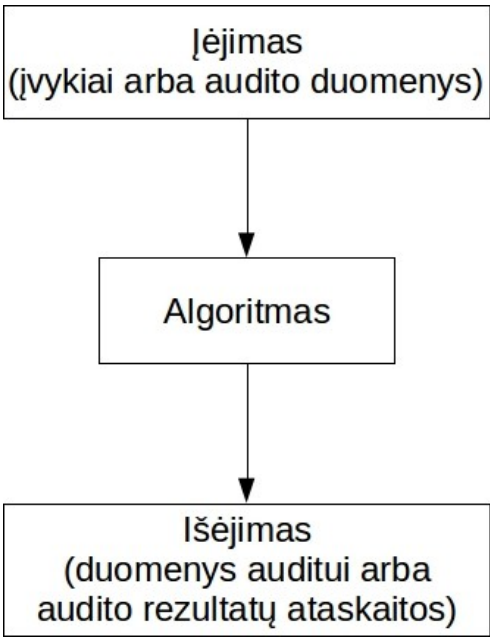
5.2. Tyrimo metodika

Tyrimų metu, taikant šiame skyriuje aprašomą tyrimo metodiką, įvertinamos analitinėje dalyje aprašytų audito metodų charakteristikos:

- *Maišos grandinė* (angl. „Hash chain“) (visos reikšmės yra skaičiuojamos tuo pačiu vienpusiu šifravimo metodu, nepraleidžiant nei vienos reikšmės. Auditas vyksta taip pat, tikrinant kiekvieną narį);
- *Merklio medis* (angl. „Merkle tree“) (be įvykių pasirašymu po kiekvieno įvykio įterpimo);
- *Merklio medis* (su įvykių pasirašymu po kiekvieno įvykio įterpimo);
- *Istorijos medis* (angl. „history tree“) (be įvykių pasirašymu po kiekvieno įvykio įterpimo);
- *Istorijos medis* (su įvykių pasirašymu po kiekvieno įvykio įterpimo);

Šio tyrimo metu yra naudojamas „juodosios dėžės“ principas (žr. Pav. 19) – kiekvieno algoritmo charakteristikos yra vertinamos kaip atskiros sistemos. Kiekvienam algoritmui yra pateikiami įvesties duomenys ir gaunami išvesties duomenys bei tyrime numatytų charakteristikų rezultatai. Po to yra pateikiami modifikuoti įvesties duomenys ir gaunami išvesties duomenys bei nauji darbe nurodytų charakteristikų rezultatai. Metodų rezultatai yra matuojami specialiai parašyta programine įranga įvertinant audito žurnalų apdorojimo charakteristikas.

2 Taikomasis tyrimas (ang. „Applied research“) - tai tyrimas siekiant įgyti naujų žinių apie konkrečią problemą, kurią reikia išspręsti, arba tikslą, kurį reikia pasiekti [30]



Pav. 18: Juodosios dėžės principas

Šio tyrimo kontekste įėjimo duomenys yra iš anksto paruošti, realūs, įmonės įvykių žurnalo duomenys, kuriais bus operuojama, taip pat įėjimo duomenys gali būti audito duomenys, kurie jau aksčiau buvo sugeneruoti tuo pačiu metodu. Išėjimas (galutinis rezultatas) abiem atvejais yra skirtingas – pirmuoju atveju išvestis yra algoritmo sudarytas apsaugotas įvykių audito žurnalas, o antruoju atveju algoritmo audito rezultatas – pakitimas aptiktas arba ne ir metodo audito charakteristikos pateikiamos kaip algoritmo vykdymo statistika. Kadangi tiriami algoritmai turi aptikti pasikeitimus šimto procentų patikimumu, tai yra matuojamas ne aptikimo santykis, o greitaveika, talpyklos sąnaudos. Įvertinimo charakteristikas šiame tyrime galima skirstyti į tris bazines grupes:

- Audito žurnalo sudarymo greitis iš įeinančių duomenų.
- Sudaryto žurnalo vientisumo audito atlikimo greitis.
- Sugeneruotas audito duomenų kiekis (talpyklos sąnaudos).

5.3. Metodų įvertinimo parametrai

Matuojant visus parametrus yra matuojamas tik realių duomenų apdorojimas, o talpyklos duomenų skaitymas, rašymas rezultatuose nėra atspindimas. Kitaip tariant rezultatas yra pradedamas matuoti po įvykio nuskaitymo ir baigiamas matuoti prieš rašant į diską. Audito žurnalo sudarymo greičio parametras (žr. Lentelė 4) priklauso nuo algoritmo efektyvumo, šio parametro rezultatai gali būti tiesiniai, logaritminiai, geometriniai ir įvairios variacijos, priklausančios nuo pačio algoritmo. Audito žurnalo sudarymo greitis nustatomas skaičiuojant skirtumą tarp duomenų (įvykio) pateikimo algoritmui laiko ir įvykio paruošimo rašymui į talpyklą laiko.

Lentelė 4: Algoritmų parametrų vertinimo kriterijai. Audito žurnalo duomenų generavimo greitis.

Žymėjimas	**Aprašymas**	**Matavimo vienetai**
ZGG	Audito žurnalo duomenų generavimo greitis įvykiais per sekundę	ms/įv.

Įvykių žurnalo audito greitis (žr. Lentelė 5) priklauso ne tik nuo audito žurnalą generavusio algoritmo, o taip pat ir nuo pačio žurnalo audito atlikimo algoritmo. Algoritmo greitis matuojamas skaičiuojant skirtumą tarp audito duomenų pateikimo audito algoritmui iki duomenų apdorojimo pabaigos ir nuosprendžio ar duomenys suklastoti ar ne. Šiuo atveju algoritmui kaip duomenys yra perduodami ne būtinai vienas įvykis, tai gali būti duomenų blokas (pavyzdžiui, šaka merklio ir istorijos medžių atveju), tačiau rezultatas yra tik vieno, konkretaus, įvykio. Tai reiškia algoritmui, tam kad atliktų vieno įvykio auditą, gali būti perduotas tiek vienas įvykis, tiek didesnis duomenų rinkinys, priklausomai nuo algoritmo. Taip pat, priklausomai nuo tiriamo metodo (istorijos ir merklio medžio atvejais), algoritmų greitis matuojamas tiek naudojant pasirašymą, tiek nenaudojant.

Lentelė 5: Algoritmų parametrų vertinimo kriterijai. Audito greitis.

Žymėjimas	Aprašymas	Matavimo vienetai
ZAGbp	Žurnalo audito greitis (be duomenų pasirašymo)	ms./įv.
ZAGsp	Žurnalo audito greitis (su duomenų pasirašymo)	ms./įv.

Kiekvienas iš šiame darbe tiriamų metodų generuoja papildomus duomenis (pavyzdžiui, parašai, maišos funkcijų rezultatai), kurie vėliau yra naudojami auditui atlikti. Be to, skirtingi metodai generuoja ir skirtingas duomenų struktūras.

Lentelė 6: Algoritmų parametrų vertinimo kriterijai. Sugeneruotas audito duomenų kiekis.

Žymėjimas	Aprašymas	Matavimo vienetai
ADKb	Vienam įvykiui tenkantis duomenų kiekis po audito žurnalo sudarymo, įskaitant ir papildomus duomenis auditui.	bitai/įv.

5.4. Tiriami scenarijai

Toliau darbe yra tiriami scenarijai, kurie pasitaikys realioje aplinkoje:

- Saugaus audito žurnalo generavimas – įprasta situacija, kurios būsenoje visi algoritmai praleis daugiausia laiko. Tai sistemos, servisų ir programų įvykių rašymas į žurnalą, kurių vientisumą turi apsaugoti numatyti algoritmai.
- Įvykių žurnalo auditas – po audito žurnalų generavimo yra atliekamas žurnalų auditas, kurio metu yra patikrinama ar audito žurnalas yra autentiškas, ar nebuvo suklastotas.

5.5. Tyrimo schema

Tyrimo pradžioje yra surenkami realūs įmonės sistemų ir aplikacijų registruojamai įvykiai (įvykių registravimo žurnalai), kurie yra išsaugomi į vėliau tyrime naudojamus įvairaus dydžio failus. Šie failai vėliau yra naudojamas simuliuoti „/dev/log“ įrenginio darbą. Gavę duomenis algoritmai turi generuoti apsaugotus įvykių audito žurnalus, kurie taip pat yra saugomi failuose. Po to naudojant šiuos įvykių audito žurnalus atitinkamų metodų pagalba yra atliekami įvykių žurnalų auditai.

Tyrimas kartojamas kelis kartus su skirtingais duomenimis ir skirtingo dydžio audito žurnalais, taip siekiant nustatyti ar algoritmo vykdymas yra tiesinis, geometrinis, logaritminis ar neapibrėžtas.

Tiriant metodų priklausomybę nuo įvykių srauto intensyvumo įvykiai (failo turinys) algoritmams yra perduodami su tam tikrais užlaikymas simuliuojant įvairaus intensyvumo įvykių srautą, bei stebima kaip prie skirtingų apkrovų elgiasi algoritmai, skirtingų apkrovų matavimai yra svarbūs

norint įvertinti sistemos atsparumą *atsisakymo aptarnauti* ir *eilės išvalymo atakai* (žr. skyrių 3.2 Atakos ir grėsmės).

Kadangi algoritmo vykdymui šiek tiek įtakos (tikėtina ne daugiau kaip 4-5%) gali turėti kiti sistemos procesai, tai su kiekvienu audito žurnalų rinkiniu kiekvienam algoritmui yra daromi keli matavimai, po to skaičiuojamas trijų geriausių rezultatų vidurkis.

Nei vieno algoritmo matavimams nėra naudojama jokia išorinė programa – visi algoritmai yra įprogramuoti į tą pačią kodo bazę t. y. į specialiai šiems matavimams parašytą programą, kuri turi galimybę pati išmatuoti tam tikras, dominančias, algoritmų charakteristikas. Parašyta programa visus rezultatus išveda į Linux standartinę išvestį, todėl pasinaudojant srautų valdymo operatoriais išvestis (rezultatai) yra išsaugoma į failus. Papildomų algoritmų sukuriamų duomenų kiekis yra nustatomas Linux komandinės eilutės įrankiais skaičiuojant įvykių žurnalo duomenims talpinti sukurtų failų dydį arba skaičiuojant skirtumą tarp pradinio įvykių failo ir galutinio rezultato t. y., saugaus įvykių audito žurnalo. Šie rezultatai taip pat yra išsaugomi failuose. Gauti galutiniai rezultatai yra apdorojami grafikų braižymo priemonėmis.

5.6. Rezultatai

Šioje dalyje matuojamos tiriamų metodų charakteristikos. Visiems algoritmams, išskyrus maišos grandinę, yra daromi du eksperimentų rinkiniai: metodų tyrimas be pasirašymo siekiant išmatuoti ribines greitaveikos reikšmes ir metodų tyrimas su pasirašymu, kaip to reikalauja pačių metodų specifika. Maišos grandinės pasirašymo nenaudoja, todėl yra atliekamas tik tyrimas be pasirašymo. Šie du tyrimų rinkiniai yra išskaidomi į mažesnius: audito žurnalų generavimas ir sugeneruotų žurnalų auditas. Abiem atvejais yra siekiama nustatyti metodų greitaveiką abiejų scenarijų atvejais. Taip pat, atliekant tyrimą su pasirašymu yra matuojamas ir sugeneruotų audito žurnalų dydis. Sekančiuose skyreliuose yra aprašomi tyrimų rezultatai.

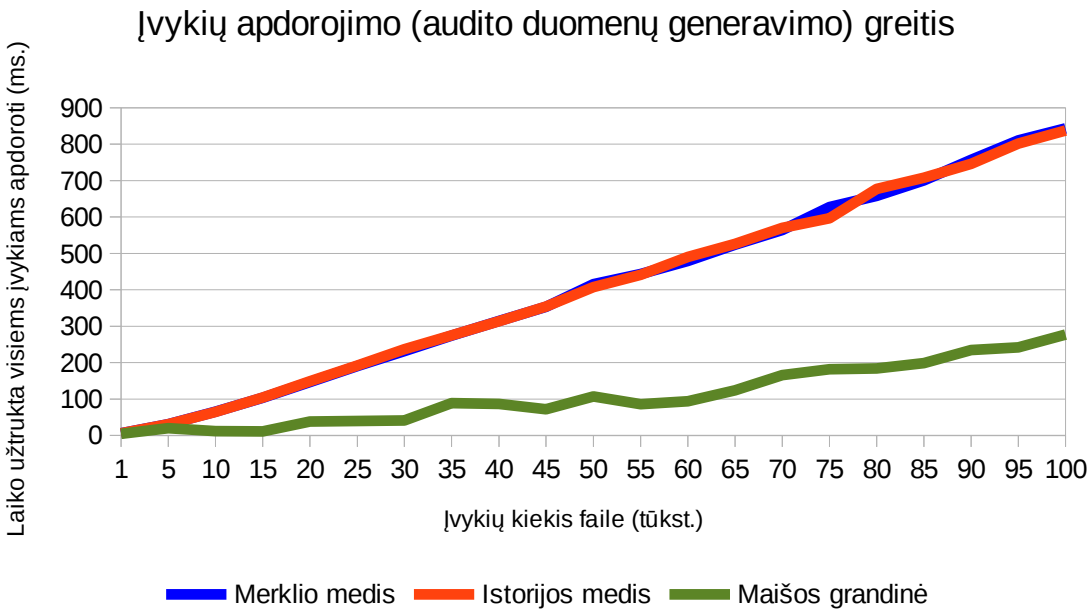
5.6.1. Metodų tyrimas (be pasirašymo)

Šioje dalyje atliekamas istorijos medžio, Merklio medžio iš maišos grandinių tyrimas kai nėra naudojamas įvykių pasirašymas (RSA arba DSA). Nors maišos grandinėms nėra būtinas įvykių pasirašymas, tačiau pasirašymas yra būtinas norint realizuoti pilnavertį istorijos ir medžio metodą (taip pat ir Merklio medžio).

Priežastis kodėl yra atliekamas „nepilnaverčių“ metodų tyrimas yra įvertinti pačių istorijos ir Merklio medžių charakteristikas kai jos nėra veikiamos kitų metodų. Na, o pilnavertis istorijos ir Merklio medžių metodų (kaip ir aprašyta pačių autorių darbuose) tyrimas su įvykių pasirašymu yra tiriamas sekančiame skyriuje.

Audito žurnalų generavimas

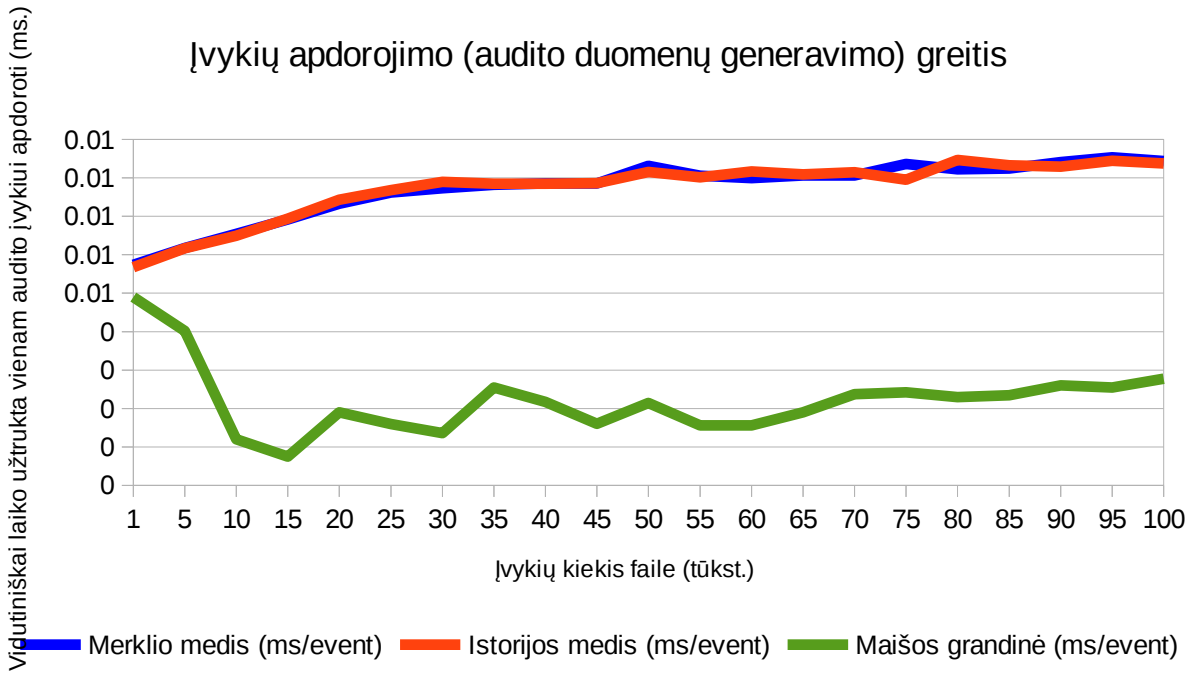
Audito žurnalų generavimo algoritmui buvo perduotas 21 skirtingo dydžio failas. Šiuose failuose buvo iš anksto paruošti įvykių žurnalai atviru tekstu.



Pav. 19: Audito žurnalo duomenų generavimo greitis (be pasirašymo)

Tam, kad išvengti įvairių „butelio kaklelių“ (angl. „bottle-neck“), duomenys buvo perduodami iš atminties (žr. Pav. 17 schemą). Įvykių kiekis skirtinguose failuose pažymėtas visų diagramų X ašyje.

Kaip matome iš Pav. 19, eksperimentiniai rezultatai yra tokie, kokių buvo tikėtasi atlikus merklio medžio analizę (žr. 3.3.4 Merklio medžiai skyriaus Pav. 10) - didėjant įvykių žurnalo dydžiui tuo pačiu, linijiniu greičiu, didėja ir laikas, kurį algoritmas užtrunka atlikdamas visų įvykių apdorojimą. Kadangi vėliau analitinėje dalyje nagrinėtas istorijos medis tiesiogiai remiasi Merklio medžiu, tai Merklio medžio ir istorijos medžio greitaveika beveik nesiskiria. Maišos grandinės algoritmo veikimo laikas taip pat ilgėja su vis didesniu įvykių kiekiu (vis didesniais failais), tačiau ne taip greitai kaip Merklio ar istorijos medžiai. Lyginant su Merklio medžiu maišos grandinė yra nuo 17,01% iki 820% greitesnė, vidutiniškai 305,8%. Lyginant su istorijos medžiu maišos grandinė yra nuo 15,83% iki 822,40% greitesnė, vidutiniškai 306,40%.



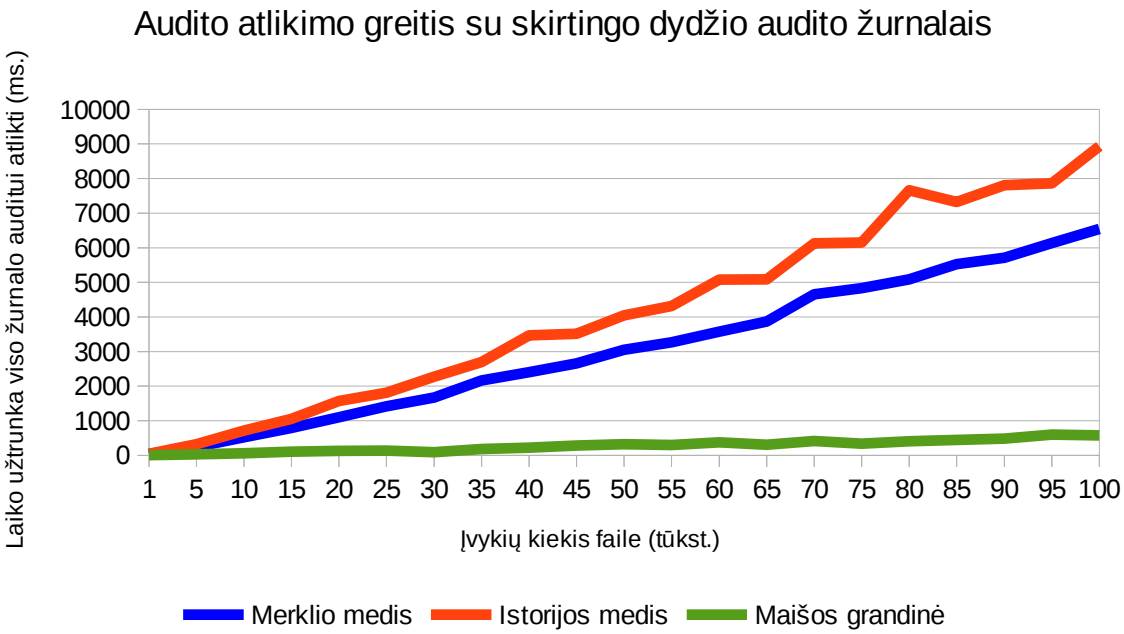
Pav. 20: Audito žurnalo duomenų generavimo greitis vienam įvykiui (be pasirašymo)

Kitą algoritmų pusę atskleidžia tyrimo rezultatai, kuriuose buvo fiksuojamas laikas skirtas vieno įvykio apdorojimui (žr. Pav. 20). Tyrimo rezultatai vėl yra labai panašus į analitinėje dalyje nagrinėto Merklio medžio teorinius rezultatus – didėjant įvykių kiekiui medyje, o tuo pačiu ir pačiam medžiui, didėja ir laikas skirtas apdoroti vieną įvykį. Šis augimas nėra taip stipriai išreikštas kaip analitinėje dalyje, tačiau galime pastebėti tiek Merklio medžio, tiek, tuo pačiu, ir istorijos medžio skaičiavimų kompleksiškumo logaritmiškumą $O(\log_2 N)$, tuo tarpu, nors iš Pav. 20 ir gali pasirodyti kitaip, *vidutinis* maišos grandinės greitis išlieka vienodas, tai yra neturi nei lėtėjimo, nei greitėjimo tendencijos.

Sugeneruotų įvykių žurnalų auditas

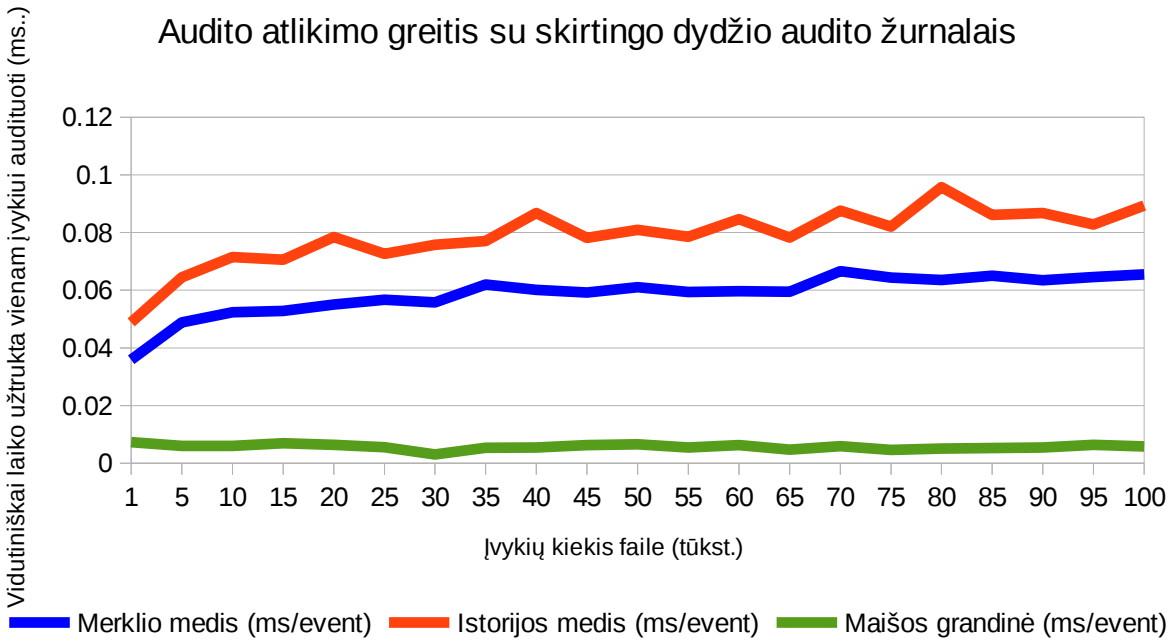
Prieš tai buvusiame skyrelyje buvo atliekamas audito žurnalų generavimo eksperimentas, kurio metu buvo sugeneruoti taip pat 21 skirtingo dydžio failai, turintys skirtingą kiekį įvykių, kurių vientisumas jau yra apsaugotas atitinkamais algoritmais.

Šiame skyrelyje pateikiami šių sudarytų įvykių žurnalų audito atlikimo rezultatai.



Pav. 21: Sugeneruoto žurnalo vientisumo audito greitis (be pasirašymo)

Kaip matome iš Pav. 21 maišos grandinė, atliekant įvykių žurnalo auditą, vėl gi lieka pirma pagal greitį, tačiau atliekant žurnalų auditą naudojant Merklio ir istorijos medžius, buvo gauti šiek tiek kitokie rezultatai negu buvo galima tikėtis atlikus metodų analizę analitinėje dalyje – Pav. 21 diagramoje aiškiai matosi, kad istorijos medis yra gerokai lėtesnis atliekant auditą, nors audito žurnalo generavimo metu parodė beveik tokius pačius rezultatus kaip ir Merklio medis (0.06% skirtumą, generuojant audito žurnalą, galima vertinti kaip paklaidą).



Pav. 22: Sugeneruoto žurnalo vientisumo audito greitis vienam įvykiui (be pasirašymo)

Audito atlikimo metu maišos grandinė buvo nuo 392,92% iki 1730,71% greitesnė už Merklio medį ir nuo 572,07% iki 2386,88% greitesnė už istorijos medį. Vidutiniškai maišos grandinės metodas buvo 968,71% greitesnis negu merklio medis ir 1337,98% greitesnis už istorijos medį. Atliekant auditą Merklio medis yra nuo 124,40% iki 150,60% (vidutiniškai 134,63%) greitesnis už istorijos medį.

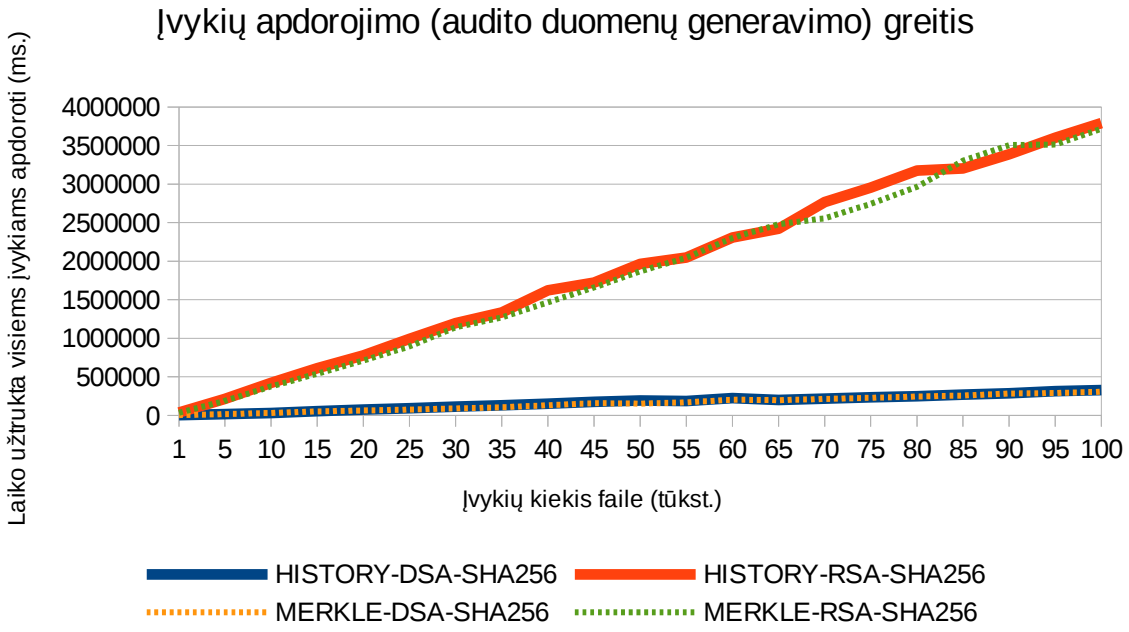
Diagramoje atvaizduotoje Pav. 22 vėl matome teoriškai numatytus rezultatus, kuomet yra matuojamas laikas ne viso žurnalo, o atskirų įvykių audito atlikimo laikas. Kaip ir žurnalo generavimo metu, taip ir audito atlikimo metu, yra pastebimas merklio ir istorijos medžio skaičiavimų kompleksiškumo algoritmiškumas $O(\log_2 N)$, o maišos grandinės vidutinis greitis nuo audito žurnalo dydžio nepriklauso.

5.6.2. Metodų tyrimas (su pasirašymu)

Šiame skyriuje yra daromas „pilnavertis“ istorijos medžio [14] tyrimas, šakninės ir tarpinių mazgų maišų rezultatams pasirašyti naudojant (kaip ir rašoma darbe) DSA arba RSA raktus. Kadangi istorijos medis buvo išvestas iš Merklio medžio ir šie metodai yra gimininginiai, tai toliau skyriuje bus matuojamas ir merklio medžių veikimas su parašais. Skirtingai negu istorijos medyje Merklio medyje kaskart yra pasirašoma vis perskaičiuojama ir atnaujinama šakninė medžio maiša, o tarpinių mazgų rezultatai nėra nei pasirašinėjami, nei išsaugomi audito žurnale.

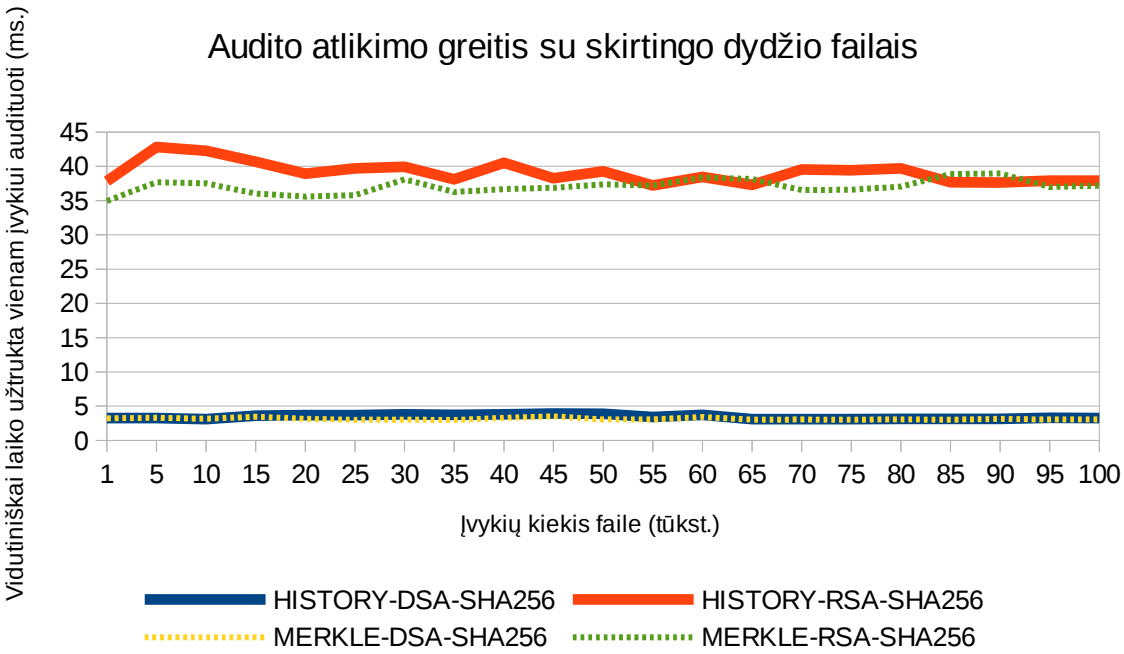
Audito žurnalų generavimas

Šiame skyrelyje jau yra atliekamas „pilnavertis“ metodų tyrimas pagal autorių pateiktą informaciją, t. y. Tam tikru metu generuojant audito žurnalus yra RSA arba DSA raktais pasirašomi nauji duomenys taip siekiant apsisaugoti nuo kai kurių atakų (pavyzdžiui, nukirtimo). Kiekvienam algoritmui tokia pačia tvarka yra perduodami tie patys duomenų rinkiniai, kurie buvo perduodami ir atliekant tyrimą be pasirašymo.



Pav. 23: Audito žurnalo duomenų generavimo greitis (su pasirašymu)

Iš Pav. 23 pateiktos diagramos pastebime, kad rezultatai mažai skiriasi nuo tyrimo atlikto be pasirašymo, išskyrus tai, kad yra ženkliai padidėję generavimo laikai. Tačiau tiek Merklio medžio, tiek istorijos medžio tendencijos lieka panašios. Taip pat reikia pastebėti, kad generuojant audito žurnalus su DSA parašais metodas veikia žymiai lėčiau, negu generuojant tuos pačius duomenis pasirašant RSA raktu. Tačiau tai yra tikėtas rezultatas, nes yra žinoma, kad pasirašinėjant duomenis RSA yra greitesnis algoritmas.



Pav. 24: Audito žurnalo duomenų generavimo greitis vienam įvykiui (su pasirašymu)

Diagramoje atvaizduotoje Pav. 24 yra pateiktas kiekvieno metodo su abiem pasirašymo raktais audito duomenų generavimo greitis vienam įvykiui. Panašus tyrimas buvo atliktas ir ankstesniame skyrelyje, kai buvo daromi bandymai be pasirašymo. Lygindami šiame skyrelyje pateiktus rezultatus su analogišku tyrimu be pasirašymo ankstesniame skyrelyje (žr. Pav. 22) iškart galime pastebėti, kad žymiai pailgėjo vienam įvykiui apdoroti užtruktas laikas, tačiau tai, dėl RSA ir DSA specifikos, yra normalūs ir tikėtini rezultatai.

Kitas pastebimas skirtumas yra tai, kad grafike (žr. Pav. 24) nebesimato skaičiavimų kompleksiškumo logaritmiškumo $O(\log_2 N)$, kuris buvo Pav. 22. Iš tiesų pačių Merklio ir istorijos medžių logaritmiškumas naudojant pasirašymą RSA/DSA raktus neišnyko, tiesiog pačio audito medžio generavimo laikas „paskendo“ laike, kuris yra sugaištamas pasirašinėjant duomenis.

Pasirašant duomenis DSA raktu audito žurnalo generavimo laikas, lyginant istorijos medžius, yra nuo 872,22% iki 1260,03% greitesnis už audito žurnalo generavimą naudojant RSA parašus, vidutiniškai 1045,08% greitesnis. Lyginant tarpusavyje Merklio medžius su DSA ir RSA parašais DSA leidžia generuoti audito žurnalus nuo 943,33% iki 1184,62% greičiau negu RSA, o vidutiniškai 1080,34% greičiau.

Lyginant istorijos medžio ir merklio medžio generavimo laiką kai nėra naudojamas pasirašymas ir kai jis yra naudojamas, skirtumas yra labai žymus. Istorijos medžio atveju, nenaudojant pasirašymo, audito žurnalas yra generuojamas vidutiniškai 450 *kartų* greičiau negu naudojant DSA ir vidutiniškai 5127 *kartus* greičiau negu naudojant RSA.

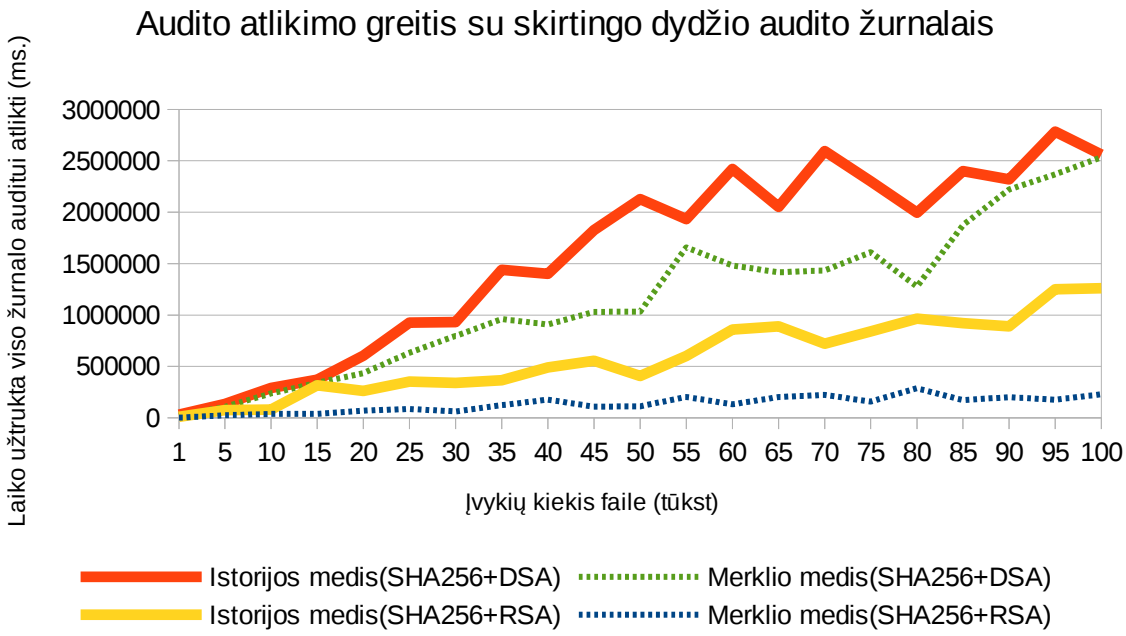
Lyginant Merklio medžio rezultatus su parašų naudojimu ir be jų audito žurnalas yra

generuojamas vidutiniškai 413 *kartų* greičiau negu naudojant DSA ir vidutiniškai 4845 *kartus* greičiau negu naudojant RSA.

Turint šiuos rezultatus galime daryti tarpinę išvadą, kad naudojant istorijos ar merklio medžio metodą su įvykių parašais audito žurnale labai svarbus faktorius yra viešojo rakto algoritmo greitis pasirašant duomenis ir naudojamas parašas viso metodo greitį įtakoja labiau negu pats merklio, istorijos medžio ar maišos grandinių naudojimas.

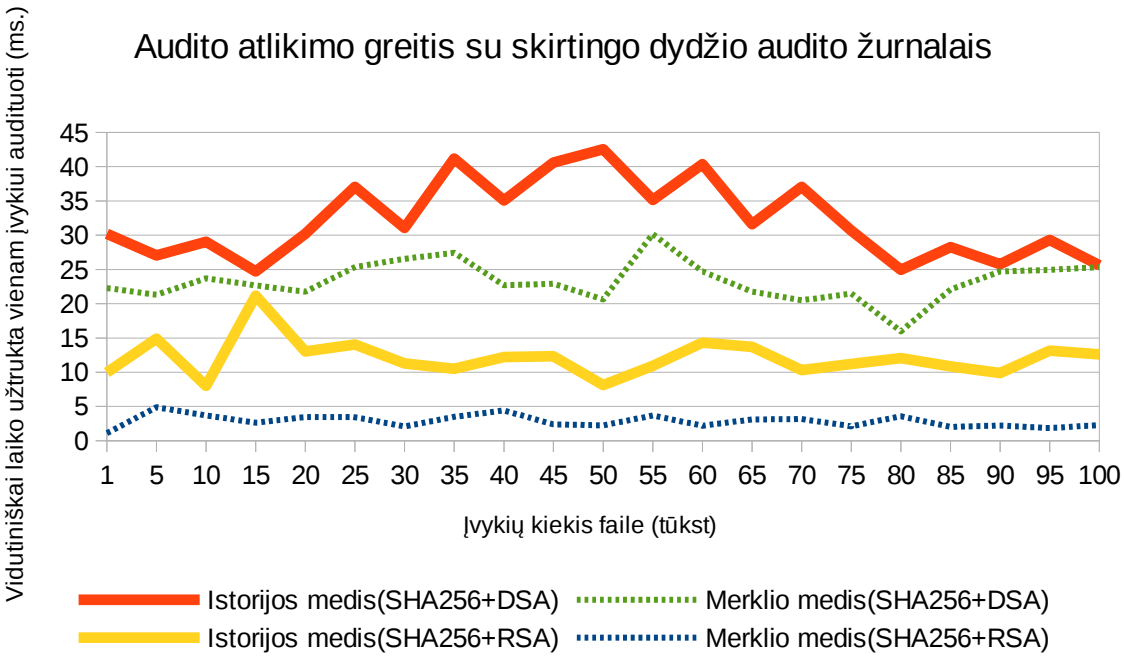
Sugeneruotų įvykių žurnalų auditas

Sugeneruotų žurnalų audito vykdymo laikas visam žurnalui patikrinti yra pateiktas Pav. 25, o audito vykdymo laikas vienam įvykiui žurnale patikrinti yra pateiktas Pav. 26.



Pav. 25: Sugeneruoto žurnalo vientisumo audito greitis (su pasirašymu)

Parašų naudojimas, kaip ir tikimasi, nepakeičia audito vykdymo laiko kreivių, o tik padidina laiką, reikalingą žurnalo auditui atlikti. Lyginant su žurnalo audito atlikimo greičiu be pasirašymo (žr. Pav. 21) istorijos medis, atliekant auditą, yra lėtesnis už merklio medį, tiek naudojant DSA, tiek RSA parašus. Greičiausias metodas yra gana aiškiai išsiskiriantis Merklio medžio ir RSA parašo derinys, o lėčiausias yra istorijos medžio ir DSA derinys.



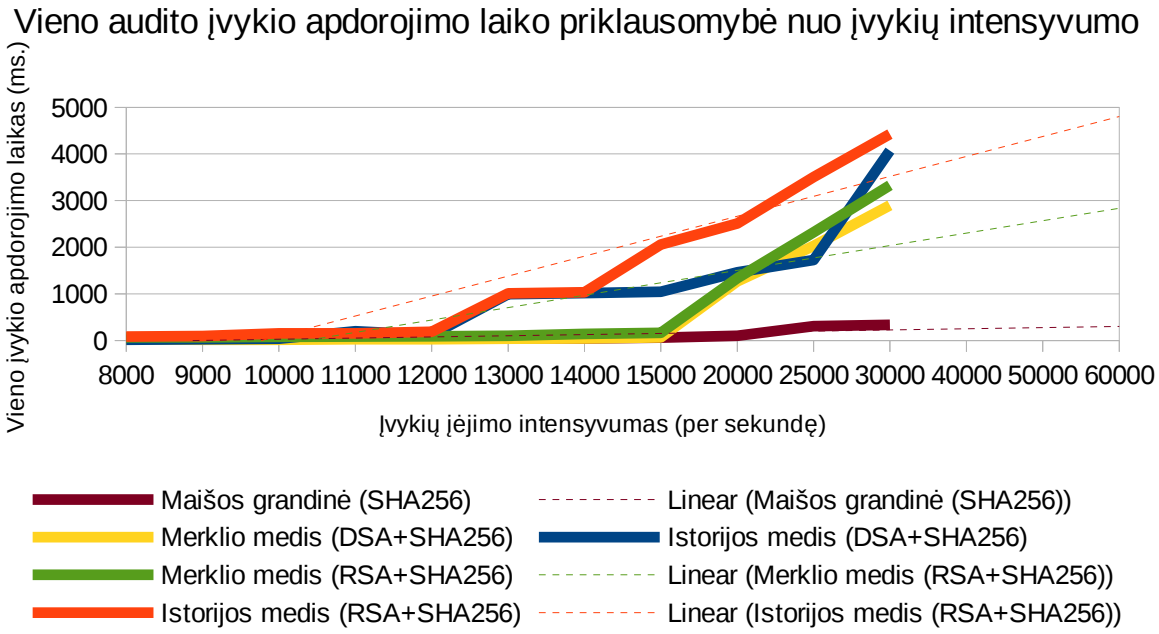
Pav. 26: Sugeneruoto žurnalo vientisumo audito greitis vienam įvykiui (su pasirašymu)

Lyginant algoritmo sugaištą laiką vieno įvykio auditui atlikti su skirtingais duomenų rinkiniais, matome, kad atliekant auditą be pasirašymo (žr. Pav. 22) ir šiuo metu su pasirašymu (žr. Pav. 26) vienas labiausiai į akis krentantis skirtumas, kaip ir generuojant audito žurnalus su pasirašymu (žr. Pav. 24) yra tai, kad logaritmiškumas yra nebepastebimas, tačiau, kaip ir audito žurnalo generavimo su pasirašymu metu, Merklio ar istorijos medžio logaritmiškumas nėra dingęs, tiesiog laikas skirtas DSA ir RSA parašams patikrinti yra pakankamai didelis, kad užgožtų Merklio ir istorijos medžių ypatybes.

Lyginant laiką, sugaištą viso žurnalo auditui atlikti, Merklio medžio su DSA parašu audito laikas yra nuo 101% iki 206% greitesnis negu tokių pačių charakteristikų istorijos medžio, o lyginant Merklio ir istorijos medžius su RSA parašais Merklio medis yra nuo 217% iki 901% greitesnis. Tuo pačiu, lyginant greičiausią (Merklio medis ir RSA) ir lėčiausią (istorijos medis ir DSA) metodus, audito laiko skirtumas yra nuo 553% iki 2736% greitesnis Merklio medžio su RSA parašu naudai.

5.6.3. Įeinančių įvykių srauto intensyvumo modeliavimas

Šioje dalyje yra atliekamas įvykių srauto intensyvumo modeliavimas kuomet įvykiai yra siunčiami į sistemą saugaus audito žurnalo sudarymui. Šioje dalyje įvykiai (failo turinys) algoritmams yra perduodami su tam tikrais užlaikymas simuliuojant įvairaus intensyvumo įvykių srautą, bei stebima kaip jie elgiasi prie skirtingų apkrovų. Skirtingų apkrovų matavimai yra svarbūs norint įvertinti sistemos atsparumą *atsisakymo aptarnauti* ir *eilės išvalymo atakai* (žr. skyrių 3.2 Atakos ir grėsmės). Įvykių srauto intensyvumas yra didinamas nuo 100 iki 30000 įvykių per sekundę kiekvienam algoritmui. Kuo intensyvesnį srautą metodas sugeba apdoroti per kuo trumpesnį eilės užlaikymo laiką tuo yra geriau, nes sistemą puolančiam asmeniui lieka mažesnis „saugus langas“ atakai įvykdyti. Kadangi ši ataka gali būti panaudota tik prieš audito žurnalų generavimą, tai šiame skyrelyje yra tiriamos tik audito žurnalų generavimo charakteristikos, o audito atlikimo charakteristikos netiriamos.



Pav. 27: Metodų greitaveikos priklausomybė nuo įvykių įėjimo intensyvumo

Grafike, atvaizduotame Pav. 27, yra vaizduojamas visų tyrimo dalyje tiriamų metodų greitaveikos priklausomybės nuo įvykių įėjimo intensyvumo, kuris yra pateikiamas kiekvienam iš metodų. Įvykių intensyvumo modeliavimas buvo atliktas didinant intensyvumą nuo 100 iki 30000 įvykių per sekundę. Kadangi (lyginant su 30000 įvykių per sekundę) iki 8000 įvykių per sekundę visų algoritmų greitaveika skiriasi tik tiek, kad grafike matoma viena lygi linija (kaip ir matoma ties 8000 riba), tai diagramoje yra vaizduojamas įvykių intensyvumas tik nuo 8000 įvykių per sekundę. Tyrimas su kiekvienu metodu yra užbaigtas ties 30000 įvykių per sekundę riba, bet norint aiškiau matyti skirtumą tarp metodų greitaveikos grafike X ašis yra išplėsta iki 60000 įvykių per sekundę ribos ir atvaizduotos krypties linijos (angl. „trendline“). Taip pat grafike yra padaryta ir maišos grandinės korekcija – tam kad vietoje maišos grandinės rezultatų būtų matoma greitaveikos tendencija, o ne tiesi linija ties 0 ms., maišos grandinės rezultatai buvo padauginti iš 100.

Iš diagramos (žr. Pav. 27) matome, kad anksčiausiai lėtėti pradeda istorijos medis tiek su RSA, tiek su DSA parašais. Merklio medis su tokiais pačiais tyrimo parametrais yra šiek tiek greitesnis ir lėtėti pradeda tik prie didesnio įvykių srauto intensyvumo. Toks rezultatas yra šiek tiek netikėtas, nes atliekant ankstesnius audito žurnalo generavimo tyrimus be įvykių intensyvumo modeliavimo (žr. Pav. 23), istorijos ir Merklio medžio greitaveika beveik nesiskyrė. Istorijos medžio audito žurnalų generavimas pradeda lėtėti virš 12000 įvykių/sek. ribos, Merklio medis virš 15000 įvykių/sek. ribos, o maišos grandinės sulėtėjimas prasideda tik perkopus 20000 įvykių/sek. ribą. Turint šiuos rezultatus galima teigti, kad labiausiai eilės atakai yra pažeidžiamas Istorijos medžio metodas, ypač naudojant RSA parašus.

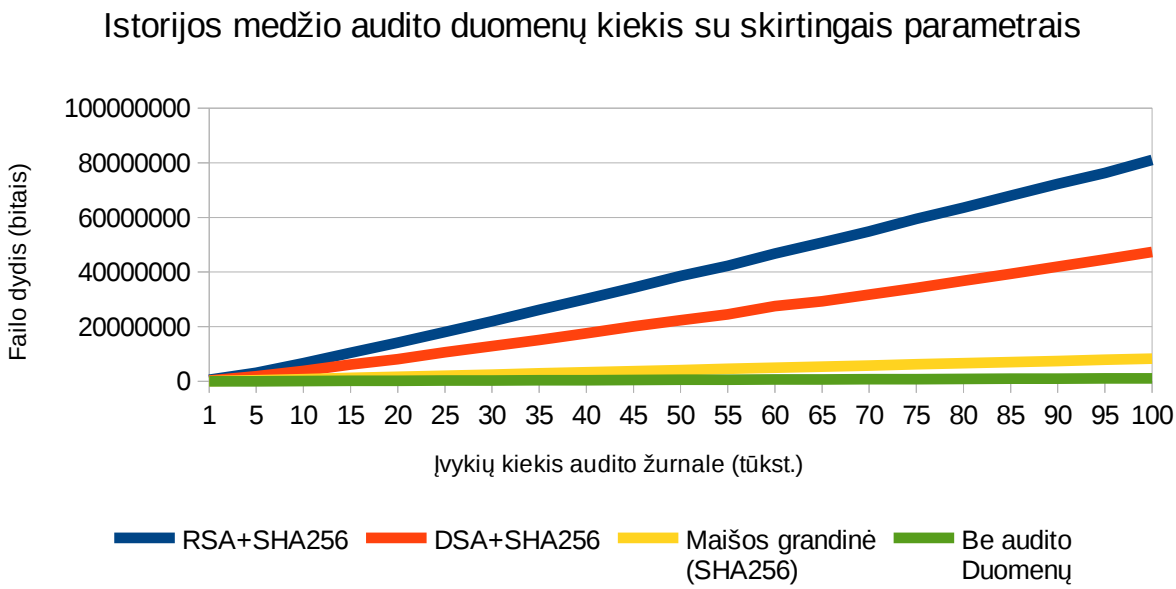
5.6.4. Talpyklos resursų sąnaudos

Talpykla šio tyrimo kontekste yra duomenų saugykla, kurioje yra saugomi ir kaupiami įvairūs duomenys įskaitant ir įvykių audito žurnalus. Talpykla nebūtinai yra kietasis diskas, tai gali būti ir operatyvinė atmintis (laikiniems duomenims kaupti), ir optiniai diskai, ir bet kokia kitokia laikmena. Kadangi Lietuvoje įstatymais yra numatyta prievolė kaupti el. duomenis šešis mėnesius [10], tai

kaupiamas duomenų kiekis gali būti labai didelis ir sudaryti netgi terabaitus talpyklos vietos vien jau skaičiuojant paprasto teksto (arba dvejetainius) failus, kurių vientisumo saugumas nėra užtikrintas, o apsaugotų audito žurnalų struktūros prideda dar keletą procentų nuo šio dydžio. Dėl šių priežasčių talpyklos resursų sąnaudos yra vienas iš svarbių parametrų, kuriuos reikia ištirti su kiekvienu šio darbo tyrime naudojamu metodu.

Istorijos medis

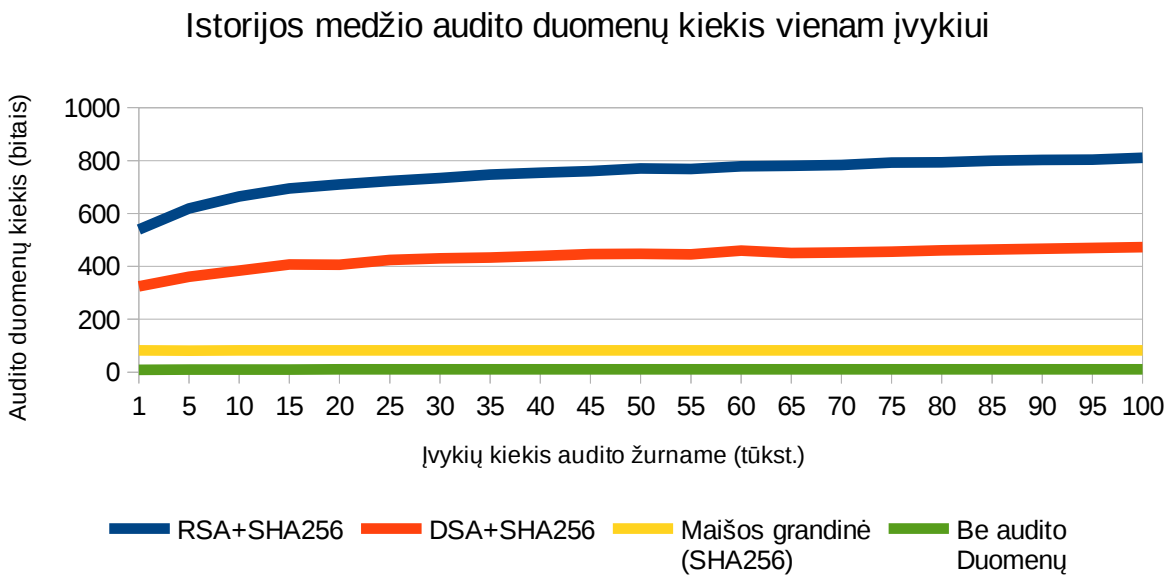
Diagramoje, atvaizduotoje Pav. 28, yra vaizduojamas duomenų kiekis reikalingas visam atitinkamo dydžio audito žurnalui išsaugoti. Audito žurnalo dydis yra įvykių kiekis faile, kuris yra pateiktas X ašyje. Grafike atvaizduotame Pav. 29 yra atvaizduojamas talpyklos duomenų kiekis reikalingas vienam įvykiui į atitinkamą audito žurnalo struktūrą patalpinti.



Pav. 28: Istorijos medžio, su skirtingais parametrais, sugeneruotas audito duomenų kiekis

Siekiant iš talpyklos sąnaudų tyrimo eliminuoti audito žurnalų įvykių dydžių svyravimus, vietoj realių duomenų buvo sugeneruoti skirtingų dydžių failai, bet kuriuose tuo pačiu yra vienodo dydžio įvykių eilutės, po to failai buvo apdoroti šiame tyrime naudojamais algoritmais ir išmatuotas naujų, apsaugotų, audito žurnalų dydis, kuris ir yra atvaizduotas Pav. 28 ir Pav. 29. Palyginimui su kitais metodais grafikuose yra atvaizduotas įvykių žurnalas be jokio apdorojimo kuriuo nors algoritmu, tai yra „Be audito duomenų“ reiškia tik patį atviro teksto įvykių žurnalą dvejetainėje formoje, be maišos rezultatų, parašų ar kitokių struktūrų ar duomenų.

Iš Pav. 28 matome, kad daugiausia duomenų audito žurnalui reikalauja istorijos medis su RSA parašu, šiek tiek mažiau talpyklos vietos reikalaujama audito žurnalui, kuris yra generuojamas DSA parašą. Audito žurnalo dydis su 1000 tūkst. įvykių ir RSA parašu užima 9881,96 kilobaitus, su DSA 5774,63 kilobaitus, o be audito duomenų vos 132,92 kilobaitų. Taigi, generuojant audito žurnalą naudojant istorijos medžio metodą su RSA parašu reikia 7442% (arba 74,42 karto), o naudojant DSA parašus reikia 4344% (arba 43,44 karto) daugiau talpyklos vietos. Tuo tarpu naudojant maišos grandinę reikia 761% (arba 7,61 karto) daugiau duomenų.



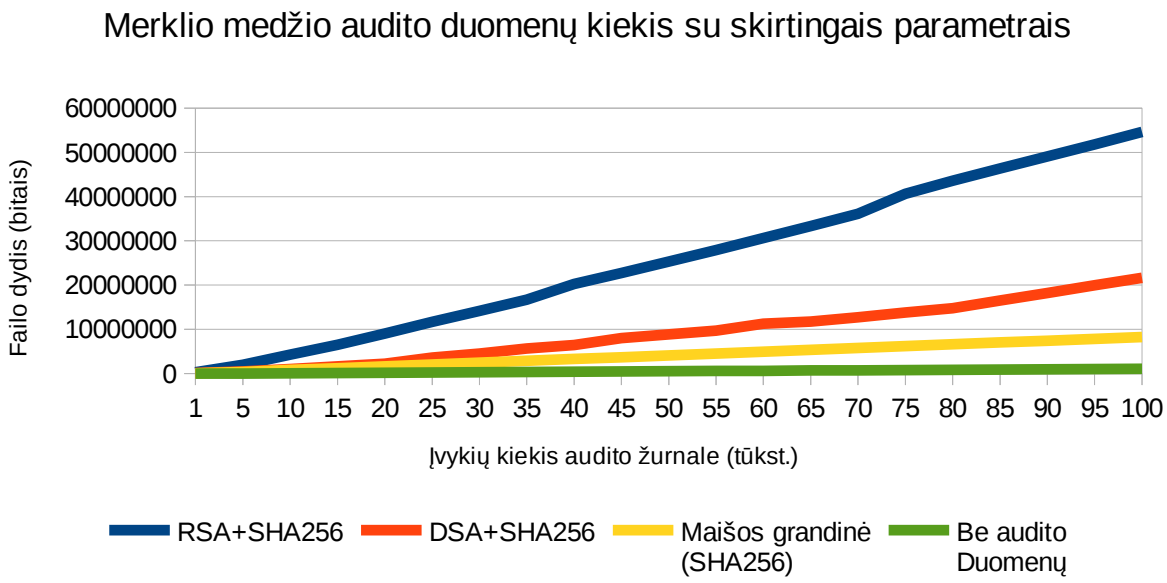
Pav. 29: Istorijos medžio, su skirtingais parametrais, sugeneruotas audito duomenų kiekis vienam įvykiui

Iš Pav. 29 matome, kad naudojant istorijos medžio metodą iš pradžių duomenų kiekis, reikalingas vieno įvykio audito duomenims generuoti yra mažesnis, o po to logaritmiškai didėja. Tokį talpyklos duomenų sąnaudų didėjimą nuo audito žurnalo dydžio lemia istorijos medžio struktūra (žr. Pav. 9) – daugėjant įvykių kiekiui audito žurnale, logaritmiškai didėja ir struktūros dydis (šakos ilgis).

Merklio medis

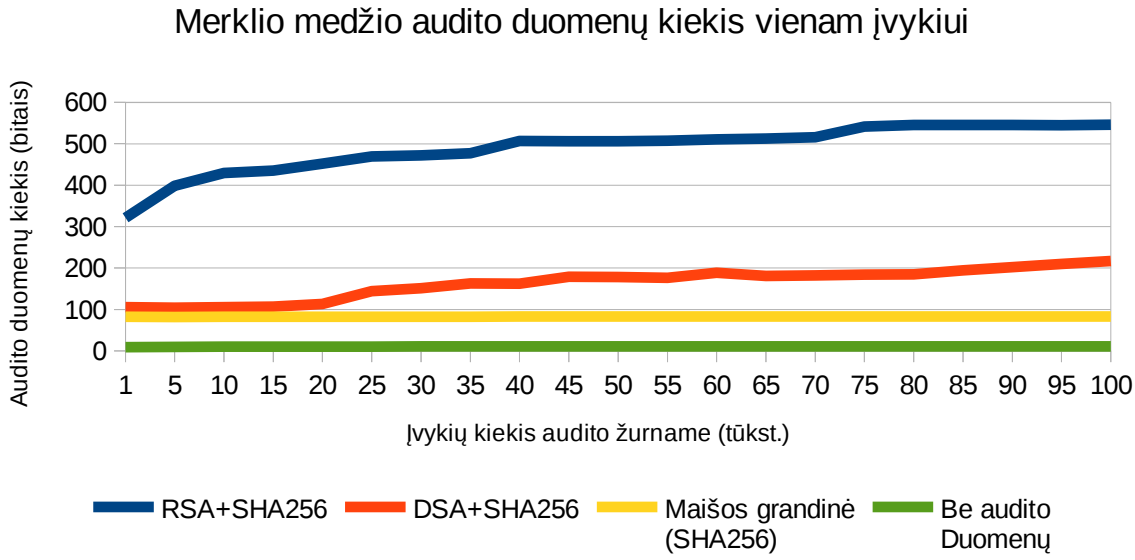
Grafike atvaizduotame Pav. 30 yra vaizduojamas duomenų kiekis reikalingas visam atitinkamo dydžio audito žurnalui išsaugoti. Audito žurnalo dydis yra įvykių kiekis faile, kuris yra pateiktas X ašyje. Grafike atvaizduotame Pav. 31 yra atvaizduojamas talpyklos duomenų kiekis reikalingas vienam įvykiui į atitinkamą audito žurnalo struktūrą patalpinti.

Merklio medis ir istorijos medis yra panašūs algoritmai. Kaip jau buvo rašyta analitinėje dalyje (žr. 3.3.5 Istorijos medis skyrių) istorijos medis yra Merklio medžio algoritmo variacija. Dėl šios priežasties turėtų būti panašūs ir rezultatai, tačiau Merklio medis, skirtingai negu istorijos medis, nesaugo visų vidinių mazgų maišos rezultatų, todėl galutinės viso audito žurnalo talpyklos sąnaudos yra gerokai mažesnės.



Pav. 30: Merklio medžio, su skirtingais parametrais, sugeneruotas audito duomenų kiekis

Kaip ir istorijos medžio atveju taip ir Merklio medžio atveju daugiausia talpyklos resursų šis metodas reikalauja kai yra naudojami RSA parašai ir šiek tiek mažiau talpyklos resursų reikalaujama kai naudojami DSA parašai. Audito žurnalo dydis su 1000 tūkst. įvykių ir RSA parašu užima 6664,83 kilobaitus (3217,13 kilobaitų mažiau negu istorijos medžio atveju), su DSA 2644,62 kilobaitus (3130,01 kilobaitų mažiau negu istorijos medžio atveju), o be audito duomenų vos 132,92 kilobaitus. Taigi, generuojant audito žurnalą naudojant Merklio medžio metodą su RSA parašu reikia 5014% (istorijos medžio atveju reikėjo 7442%), o naudojant DSA parašus reikia 1989% (istorijos medžio atveju reikėjo 4344%) daugiau talpyklos vietos. Tuo tarpu naudojant maišos grandinę reikia 761% (arba 7,61 karto) daugiau talpyklos vietos, kai maišos grandinėje yra saugomi visų tarpinių grandžių maišos rezultatai.



Pav. 31: Merklio medžio, su skirtingais parametrais, sugeneruotas audito duomenų kiekis vienam įvykiui

Duomenų kiekis vieno įvykio audito duomenims sugeneruoti yra atvaizduotas diagramoje esančioje 31 paveikslėlyje. Rezultatai yra panašūs į istorijos medžio. Matosi logaritmiškumas panašus į istorijos medžio, tačiau, skirtingai negu istorijos medžio atveju, Merklio medžio grafike matomas duomenų kiekio laipsniškas didėjimas, kurį lemia pačio metodo ypatybės. Jeigu istorijos medžio atveju su kiekviena nauja šaka būdavo išsaugomas ir tarpinio mazgo maišos rezultatas, tai Merklio medžio atveju yra išsaugomas tik šakos šakninės maišos rezultatas, o tuo pačiu išsaugoma ir mažiau DSA arba RSA parašų.

5.7. Rezultatų ir tyrimų apibendrinimas

Visų tyrimų, atliktų šiame skyriuje, rezultatai yra apibendrinti įvykių žurnalų sudarymo greičio, žurnalų audito atlikimo greičio, atsparumo atsisakymo aptarnauti ir eilės išvalymo atakai ir talpyklos resursų sąnaudų atžvilgiu bei pateikti toliau esančiose lentelėse.

Lentelė 7: Audito žurnalo sudarymo greitis

	Audito žurnalo sudarymo greitis be pasirašymo (ms.)		**Audito žurnalo sudarymo greitis su pasirašymu (ms.)**			
	Visam failui (ties 100tūkst.)	Vienam įvykiui (vidurkis)	Visam failui (su 100tūkst. įvykių)		Vienam įvykiui (vidurkis)	
			RSA	DSA	RSA	DSA
Merklio medis	843,54	0,007727	3790172	326663	39,0926	3,4386
Istorijos medis	837,50	0,007726	3717458	307.370	37,0740	3,1483
Maišos grandinė	277,40	0,00221	-		-	

Lentelė 8: Sudarytų žurnalų audito greitis

	Sudaryto žurnalo audito greitis be parašų (ms.)		**Sudaryto žurnalo audito greitis su parašų tikrinimu (ms.)**			
	Visam failui (ties 100tūkst.)	Vienam įvykiui (vidurkis)	Visam failui (su 100tūkst. įvykių)		Vienam įvykiui (vidurkis)	
			RSA	DSA	RSA	DSA
Merklio medis	6547	0,0586	229290,5	2534518,9	2,8608	23,2765
Istorijos medis	8934	0,0789	1261276,2	2560072,6	12,1203	32,2515
Maišos grandinė	580	0,0056	-		-	

Lentelė 9: Atsparumas atsisakymo aptarnauti ir eilės išvalymo atakai

	Atsparumas atsisakymo aptarnauti ir eilės išvalymo atakai				
	Lėtėjimas prasideda perkopus (įvykiai/sek.)	Greitis ties aukščiausia tirta riba (30000, ms./įvyk.)		Vidutinis greitis visame tirtame ruože (ms./įvyk.)	
		RSA	DSA	RSA	DSA
Merklio medis	15000	3325,5537	2899,5767	350,8164	271,4420
Istorijos medis	12000	4423,1677	40717398	662,8834	451,1410
Maišos grandinė	25000	3,3605		0,5338	

Lentelė 10: Audito žurnalų generuojamas duomenų kiekis

	Audito žurnalų generuojamas duomenų kiekis					
	Visam 100 tūkst. įvykių failui (bitais)		Vidurkis (bitais)		Nuo žurnalo be audito duomenų, ties 100 tūkst. įvykių, (%)	
	RSA	DSA	RSA	DSA	RSA	DSA
Merklio medis	54598367	21664800	26047947	9205527	+5014,11%	+1989,61%
Istorijos medis	81034935	47305788	22681997	39007626	+7441,94%	+4344,38%
Maišos grandinė	5965365		2979961		+548%	

Remiantis tiriamosios darbo dalies rezultatais ir aukščiau pateiktomis lentelėmis, galima formuluoti šias tyrimo išvadas:

1. Sukurta tyrimo aplinka, schema ir metodika maksimaliai sumažina pašalinių veiksnių įtaką tyrimo rezultatams. Visi tyrimai buvo atlikti specialiai šiam tikslui parašyta programa, kuri leidžia modeliuoti įvairių metodų darbą pateikiant įvairius vykdymo parametrus. Ši programa geba modeliuoti audito žurnalų sudarymą, žurnalų auditą ir formuoti tikslias tyrimų ataskaitas. Taip pat šios programos kodą galima panaudoti ne tik tyrimo aplinkoje, tačiau su minimaliais pakeitimais pritaikyti ir realioje aplinkoje – tai ir buvo padaryta (žr. skyrių 6. REKOMENDACIJOS ĮMONĖS ĮVYKIŲ ŽURNALŲ APSAUGAI);
2. Tiriamojoje dalyje daugumoje tyrimų yra naudojami RSA bei DSA parašai, tačiau pačių RSA ir/ar DSA parašų naudojimo tyrimuose nereikia vertinti kaip pačių viešųjų raktų metodų tyrimų. Kadangi RSA ir DSA charakteristikos jau yra plačiai išnagrinėtos, todėl tyrimuose yra tiriamas tik istorijos ir Merklio medžio metodų veikimas papildant juos RSA ir/ar DSA parašais, o RSA bei DSA parašų naudojimą metoduose vertinti tik kaip papildomas priemones istorijos ir Merklio medžių metodams nuo įvairesnių atakų apsaugoti.
3. Kadangi maišos grandinių metodas nenaudoja viešojo rakto parašų, tai tyrimuose maišos grandinės yra tiriamos tik be jų. Dalyse, kuriuose kartu su maišos grandinėmis yra vaizduojami ir Merklio bei istorijos medžiai su parašais, maišos grandinių rezultatai pateikiami tik atskaitos taškas metodų palyginimui.
4. Atlikus tyrimus buvo nustatyta praktinė metodų elgsena įvairiuose scenarijuose: Merklio medžiai tiek audito žurnalų generavimo metu, tiek audito atlikimo metu elgiasi labai panašiai į teorinius paskaičiavimus, istorijos medis tuo tarpu, nors ir yra Merklio medžio variacija, audito atlikimo metu elgiasi kitaip negu buvo galima tikėtis ir visuose audito atlikimo bandymuose buvo gerokai lėtesnis už Merklio medį, tačiau audito žurnalų generavimo metu charakteristikos yra beveik identiškos.

5. Audito žurnalų generavimo metu labai aiškaus lyderio nėra (nevertinant maišos grandinių), tačiau greičiausias yra istorijos medis be pasirašymo arba su DSA parašu, o lėčiausias metodas Merklio medis su RSA parašu.
6. Sugeneruotų žurnalų audito atlikimo metu aiškus lyderis yra (nevertinant maišos grandinių) Merklio medis su RSA parašu, o lėčiausias yra istorijos medžio metodas be pasirašymo arba su DSA parašu.
7. Didžiausią atsparumą atsisakymo aptarnauti atakoms, atsparumo didėjimo tvarka, demonstravo metodai: istorijos medis, merklio medis ir maišos medis. Akivaizdus algoritmo greitaveikos degradavimas istorijos medžio atžvilgiu pastebimas perkopus 12 tūkst. įvykių per sekundę įvykių srauto intensyvumą, Merklio medžio – perkopus 15 tūkst., o maišos grandinės – 25 tūkst. įvykių per sekundę intensyvumą. Metodų greičio degradavimas pastebimas nepaisant naudojamo RSA ar DSA pasirašymo rakto, todėl galima daryti išvadą, kad tokią greičio lėtėjimo tendenciją lemia ne parašo naudojimas, o pačių merklio, istorijos medžių arba maišos grandinės ypatybės. Na, o naudojant pasirašymą labiausiai atsparus atsisakymo aptarnauti atakai yra Merklio medžio metodas su DSA parašu, o mažiausiai atsparus istorijos medžio metodas su RSA parašu.
8. Talpyklos duomenų sąnaudų atžvilgiu geriausias metodas (generuojantis mažiausiai duomenų), žinoma, yra maišos grandinė. Nevertinant maišos grandinės geriausias metodas yra Merklio medis su DSA parašu, generuojantis 1989.61% daugiau duomenų negu toks pats įvykių žurnalas be audito duomenų. O prasčiausias (generuojantis didžiausią duomenų perteklių) yra istorijos medžio metodas su RSA parašu, generuojantis 7441.94% daugiau duomenų, negu tas pats įvykių žurnalo failas be audito duomenų.
9. Daugumoje šaltinių minima, kad Merklio medis (o tuo pačiu, teoriškai, turėtų ir istorijos medis) turi logaritmišką greitį atsitiktiniams įvykiams patikrinti, tačiau beveik niekur nėra tiesiogiai užsimenama, kad atliekant ne atsitiktinį, o nuoseklų medžio auditą, šių metodų greitis stipriai mažėja. Šiame darbe, analitinėje dalyje, pastebėtą nuoseklaus audito greičio ypatybę. patvirtina ir tiriamosios dalies rezultatai. Būtent dėl šios priežasties maišos grandinių metodas visais tyrimų atvejais buvo nepralenkiamas ne tik lyginant su Merklio ir istorijos medžiais, kuomet buvo naudojamas pasirašytas, bet ir kai pasirašymas nebuvo naudojamas nei vieno įvykio atveju. Žinoma, maišos grandinės be papildomų priemonių išvis neturi galimybės atlikti žurnalo audito atsitiktine tvarka.

6. REKOMENDACIJOS ĮMONĖS ĮVYKIŲ ŽURNALŲ APSAUGAI

Įmonė, kurioje yra pritaikyti šio darbo rezultatai, iki šiol neturėjo veikiančio audito žurnalų apsaugos įrankio, skirto GNU/Linux šeimos operacijų sistemoms, tačiau turi kelis svarbius ir saugumui jautrius klientus. Kiekvienam šių klientų buvo pritaikyta modifikuota, tyrimų dalyje naudota programinė įranga. Nors ši programinė įranga gali veikti ir lokalioje sistemoje, tačiau, dėl papildomo saugumo ir įvairių kitų priežasčių, tokių kaip rolių atskyrimas, administravimo paprastumas, patikimumas ir kt., klientų įvykiams kaupti buvo pasirinktos atskiros, virtualios, sistemos. Taip pat, kadangi minėtų klientų įvykiai jau buvo kaupiami atskirose sistemose, tai šio darbo rezultatai ir programinė įranga, kol kas eksperimentinėje stadijoje, buvo pritaikyta dirbti lygiagrečiai su jau veikiančia įvykių kaupimo sistema.

Minėtų klientų poreikius galima išskirti į tris scenarijus:

1. Naujoje sistemoje kaupti visus įvykius, kurie yra kaupiami ir esamoje įvykių kaupimo sistemoje. Duomenų kiekis nėra svarbus, taip pat nėra svarbus ir audito greitis, kadangi auditas bus atliekamas tik esant reikalui;
2. Naujoje sistemoje kaupti tik su saugumu susijusius įvykius, tokius kaip vartotojų prisijungimas ir atsijungimas nuo sistemos, vartotojo pakeitimas, komandos vykdytos sistemoje ir kt. Kadangi įvykių žurnalai kaupiami apie metus laiko ir duomenų kiekis labai didelis netgi saugant tik su saugumu susijusius duomenis, tai papildomas duomenų kiekis, kurį generuoja nauja sistema, pageidautina, yra kuo mažesnis;
3. Scenarijus yra analogiškas kaip ir antruoju atveju, tačiau duomenų kiekis yra šiek tiek mažesnis ir klientui nėra svarbu papildomas duomenų kiekis, svarbiausia yra nuolatinis, pastoviai kartojamas, įvykių žurnalo auditas. Todėl pageidautina įvykių žurnalų auditą atlikti kuo greičiau, su kuo mažesnėmis resursų sąnaudomis.

Remiantis tiriamosios dalies rezultatais, išvadomis bei išvadose pateiktomis lentelėmis (žr. nuo Lentelė 7 iki Lentelė 11), galime parinkti visiems klientams optimalius (arba artimus jiems) parametrus, kuriais turi veikti nauja įvykių žurnalo kaupimo ir audito sistema, kad tenkintų įmonės klientų pageidavimus.

Pirmojo scenarijaus atveju, kadangi yra kaupiami visi duomenys, kurių srautas yra ganėtinai intensyvus, o talpyklos sąnaudos nėra svarbios, buvo pasirinktas Istorijos medžio ir DSA parašo derinys. Tai garantuos greitesnį už kitus tirtus metodus audito žurnalų generavimo greitį.

Antrojo scenarijaus atveju įvykių registravimo ir audito sistemai buvo pasirinktas Merklio medžio ir DSA parašo derinys, kadangi garantuoja mažiausias iš tirtų metodų talpyklos sąnaudas, be to, kadangi įvykių srautas yra pakankamai didelis, tai Merklio medžio ir DSA parašo derinys pasižymi ir geriausiu atsparumu atsisakymo aptarnauti atakai, kurią dėl įvykių intensyvumo, gali sukelti pačios klientų sistemos.

Trečiojo metodo atveju, kadangi pagrindinis kliento pageidavimas buvo nuolatinis žurnalų auditas, buvo rekomenduotas ir yra naudojamas Merklio medžio ir RSA parašo derinys, kadangi jis pasižymi didžiausiu audito atlikimo greičiu.

7. GALUTINĖS DARBO IŠVADOS

1 Įvykių registravimo žurnalai dėl juose kaupiamų jautrių įmonės duomenų ir svarbių sistemos sukompromitavimo įkalčių, po sistemos sukompromitavimo tampa vienu pirminiu taikinių. Siekiant pašalinti sistemos sukompromitavimo įkalčius įvykių registravimo žurnalai yra redaguojami, klastojami, ištrinami arba kitaip sunaikinami;

2 Remiantis 2009-2013m. Lietuvos Respublikos nacionalinio elektroninių ryšių tinklų ir informacijos saugumo incidentų tyrimo padalinio statistikos duomenimis sistemų užvaldymo ir duomenų klastojimo incidentų kiekis nuolat auga, todėl saugi įvykių registravimo ir įvykių žurnalų audito sistema, kurios sukauptų duomenų auditu galime pasitikėti, yra vis svarbesnė užduotis;

3 Remiantis šiame darbe, analitinėje dalyje, atlikta analize galime daryti tokias išvadas:

3.1 Šiuo metu įvykių registravimo žurnalų duomenims saugiai kaupti dažniausiai naudojamas būdas yra jų siuntimas į nutolusį kompiuterį, o dažniausias audito atlikimo būdas yra tiesiog fizinis šių žurnalų skaitymas ir tam tikros informacijos paieška, tačiau jokiais vientisumo apsaugos būdais neapsaugotais žurnalais negalima pasitikėti, nes nors duomenys ir kaupiami nutolusioje sistemoje, tačiau tai negarantuoja visiško duomenų saugumo visuose lygmenyse (fiziniame, programiniame);

3.2 IETF yra patvirtinusi standartus, kurie turi garantuoti saugų įvykių pranešimų perdavimą tinklu, tačiau nėra jokio standartizuoto, plačiai ir atvirai naudojamo metodo, apsaugoti ne duomenų perdavimą tinklu, o pačių įvykių audito žurnalų vientisumą, saugumą ir garantuojančio tokių žurnalų klastojimo akivaizdumo nustatymą;

3.3 Šiuo metu GNU/Linux šeimos operacijų sistemoje naudojami du pagrindiniai įvykių registravimo, kaupimo ir apdorojimo servisai – „syslog“ ir „the journal“, kurie yra labai skirtingi, tačiau atlieka tą pačią užduotį. Syslog neturi jokio automatinio įvykių žurnalų apsaugos ir audito mechanizmo. „The journal“ tokį mechanizmą turi, tačiau jis dokumentuotas minimaliai, nėra visiškai naujas ir yra tik eksperimentinėje stadijoje;

3.4 Tiek prieš apsaugotus, tiek prieš neapsaugotus įvykių žurnalus bei šių žurnalų sudarymo ir audito sistemas gali būti panaudotos įvairios atakos: trynimas, nukirtimas, įvykių žurnalo sukeitimas, rašymas, atsisakymo aptarnauti ataka, užtvindymas, pasitikėjimo išnaudojimas, nepaprastieji įvykiai, skaitymas, eilės perpildymas ir išvalymas

3.5 Yra sukurta keletas pagrindinių metodų, galinčių apsaugoti įvykių audito žurnalus, bei suteikti galimybę patikimai atlikti sudarytų saugių įvykių žurnalų auditą. Dalis metodų buvo kurta būtent šiam tikslui, kita dalis buvo kurta siekiant išspręsti svarbias kriptografijos problemas, o vėliau buvo pritaikyti ir įvykių žurnalų apsaugai ir auditui. Visi analitinėje dalyje analizuoti metodai turi tiek privalumų, tiek trūkumų ir sprendžia tam tikras problemas, tačiau nėra nei vieno universalaus metodo, kuris pats savaime užkirstų kelią visoms 6 išvadų punkte minėtoms atakoms, todėl norint apsaugoti įvykių žurnalus ir sukurti visapusišką įvykių žurnalų audito sistemą minėti metodai yra naudojami ne pavieniui, o tam tikri jų deriniai ar modifikacijos.

3.6 Vienas labai svarbių tokių metodų kriterijų yra jų greitaveika ir resursų sąnaudos.

4 Šiame darbe, tiriamojoje dalyje, yra atliktas trijų metodų (Merklio medžio, istorijos medžio ir maišos grandinių), kurių analizė buvo atlikta analitinėje darbo dalyje, tyrimas greitaveikos ir resursų sąnaudų atžvilgiu, atliktas metodų palyginimas, nustatytos metodų

silpnybės bei stiprybės, kurios yra apibendrintos žemiau esančioje lentelėje (žr. Lentelė 11). Skirtingi metodai yra labiau/mažiau tinkami vienam ar kitam scenarijui negu kiti, todėl vieno, universalaus ir geriausio metodo šiame darbe rasta nebuvo:

4.1 Istorijos medis su DSA parašu, pagal šio darbo tyrimų rezultatus, yra labiausiai tinkamas, kai reikia, kad įvykių žurnalai būtų sudaromi greitai ir su kuo mažesnėmis resursų sąnaudomis;

4.2 Jeigu reikia kuo greičiau ir su kuo mažesnėmis procesoriaus resursų sąnaudomis atlikti įvykių žurnalų auditą – labiausiai tinkamas metodas, pagal šio darbo rezultatus, yra Merklio medžio ir RSA parašo derinys;

4.3 Merklio medžio ir DSA parašo derinys geriausiai tinka sistemoms, kuriose yra intensyvus įvykių srautas, nes šių metodų derinys, pagal tyrimų rezultatus, yra labiausiai atsparus atsisakymo aptarnauti atakai;

4.4 Taip pat Merklio medžio ir DSA parašo derinys reikalauja mažiausiai talpyklos resursų (laisvos vietos) sąnaudų, todėl yra tinkamas ir sistemose, kuriose yra sąlyginai mažai vietos arba pageidaujama, kad įvykių žurnalai sunaudotų kuo mažiau talpyklos resursų;

5 Visų tyrimų modeliavimas ir tyrimai buvo atlikti specialiai parašyta programa, kuri, šiek tiek ją modifikavus, kartu su darbo išvadomis ir rezultatais, buvo pritaikyta įmonės veikloje pasiūlant įmonės klientams papildomas paslaugas. Rekomendacijos įvykių audito žurnalams yra pateiktos 11 lentelėje.

Lentelė 11: Tyrimų rezultatų santrauka

Scenarijus	Tinkamiausias metodas	Netinkamiausias metodas
Generavimas	Istorijos medis + DSA (SHA256)	Merklio medis + RSA (SHA256)
Auditas	Merklio medis + RSA (SHA256)	Istorijos medis + DSA (SHA256)
Atsparumas DoS	Merklio medis + DSA (SHA256)	Istorijos medis + RSA (SHA256)
Talpyklos sąnaudos	Merklio medis + DSA (SHA256)	Istorijos medis + RSA (SHA256)

LITERATŪROS ŠALTINIAI

1. **Mihir Bellare, Bennet S. Yee**. Forward Integrity For Secure Audit Logs. Technical Report, Computer Science and Engineering Department, University of San Diego, 1997.
2. **Don Coppersmith[1] and Markus Jakobsson[2]**. Almost Optimal Hash Sequence Traversal. FC'02 Proceedings of the 6th international conference on Financial cryptography, Pages 102-119, 2003.
3. Lietuvos respublikos baudžiamojo proceso kodeksas, 154 straipsnis: Elektroninių ryšių tinklais perduodamos informacijos kontrolė, jos fiksavimas ir kaupimas. – [žiūrėta 2013-02-01]. Prieiga per internetą: http://www3.lrs.lt/pls/inter3/dokpaieska.showdoc_l?p_id=438015
4. Trusted Computer System Evaluation Criteria (ORANGE BOOK), U.S., Department of Defense, 1985.
5. RFC: Signed Syslog Messages – [žiūrėta 2013-02-01]. Prieiga per internetą: http://tools.ietf.org/html/rfc5424
6. RFC: Reliable Delivery for syslog – [žiūrėta 2013-02-01]. Prieiga per internetą: http://tools.ietf.org/html/rfc3195
7. **Di Ma and Gene Tsudik**. A New Approach to Secure Logging. ACM Transactions on Storage (TOS), Volume 5 Issue 1, Article No. 2, 2009.
8. **Aryapetov, D., Ganapathi, A., Leung, L.** Improving the Protection of Logging Systems. Technical Report, UC Berkeley, 2002.
9. **Bruce Schneier, John Kelsey**. Secure Audit Logs to Support Computer Forensics ,. ACM Transactions on Information and System Security (TISSEC), Volume 2 Issue 2, Pages 159-176, 1999.
10. **Steena D. S. Monteiro, Robert F. Erbacher**. An Authentication and Validation Mechanism for Analyzing Syslogs Forensically. ACM SIGOPS Operating Systems Review, Volume 42 Issue 3, Pages 41-50, 2008.
11. **Bruce Schneier, John Kelsey**. Cryptographic Support for Secure Logs on Untrusted Machines. SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium - Volume 7, Pages 4 – 4, 1998.
12. **John Fenwick**. syslog: The UNIX System Logger. Enterprise Systems Division, Hewlett-Packard Company, 2003.
13. „The Journal“ įvykių registravimo sistema – [žiūrėta 2013-02-01]. Prieiga per internetą: http://0pointer.de/blog/projects/the-journal.html
14. **Scott A. Crosby, Dan S. Wallach**. Efficient Data Structures for Tamper-Evident Logging. SSYM'09 Proceedings of the 18th conference on USENIX security symposium, Pages 317-334, 2009.
15. **J. Kelsey, J.Callas**. Syslog-Sign Protocol DRAFT. Network Working Group, 2002. – [žiūrėta 2013-03-04]. Prieiga per internetą: http://tools.ietf.org/html/draft-ietf-syslog-sign-02
16. **S. Micali**. Efficient Certificate Revocation. In Proceedings 1997 RSA Data Security Conference, *1997*.
17. **A. Perrig, R. Cametti, D. Song, D. Tygar**. TESLA: Efficient and Secure Source Authentication for Multicast. Proceedings of Network and Distributed System Security Symposium NDSS, 2011.
18. **Gene Itkis**. Forward Security, Adaptive Cryptography: Time Evolution. Computer Science Department, Boston University, 2004.
19. **L. Lamport**. Password authentication with insecure communication. Communications of the ACM, Volume 24 Issue 11, Pages 770-772, 1981.

20. **Steven Cheung**. An efficient message authentication scheme for link state routing. ACSAC '97 Proceedings of the 13th Annual Computer Security Applications Conference, Page 90, IEEE Computer Society Washington, 1997.
21. **Adrian Perrig, Ran Canetti, J.D. Tygar, Dawn Xiaodong Song**. Efficient authentication and signing of multicast streams over lossy channels. SP '00 Proceedings of the 2000 IEEE Symposium on Security and Privacy, Page 56, IEEE Computer Society Washington, 2000.
22. **Ralph. C. Merkle**. A Digital Signature Based on a Conventional Encryption Function. CRYPTO '87 A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, Pages 369-378, Springer-Verlag, 1988.
23. **Marc Fischlin**. Fast Verification Of Hash Chains. RSA Security Cryptographer's Track 2004, Lecture Notes in Computer Science, Volume 2964, pp. 339-352, Springer-Verlag, 2004.
24. **S. Haber, W. Scott Stornetta**. How to Time Stamp a Digital Document. CRYPTO '90 Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, Pages 437-455, Springer-Verlag, 1991.
25. **D. Ma and G. Tsudik**. Forward-Secure Sequential Aggregate Authentication. SP '07 Proceedings of the 2007 IEEE Symposium on Security and Privacy, Pages 86-91, IEEE Computer Society Washington, 2007.
26. **D. Ma**. Practical Forward Secure Sequential Aggregate Signatures. ASIACCS '08 Proceedings of the 2008 ACM symposium on Information, computer and communications security, Pages 341-352, ACM New York, 2008.
27. **Freire,E.S.V., Paterson,K.G., Poettering,B.** Simple, efficient and strongly KI-secure hierarchical key assignment schemes. CT-RSA'13 Proceedings of the 13th international conference on Topics in Cryptology, Pages 101-114, Springer-Verlag Berlin, Heidelberg, 2013.
28. **Lennart Poettering.** Forward Secure Sealing (FSS) – [žiūrėta 2013-03-29]. Prieiga per internetą: https://plus.google.com/115547683951727699051/posts/g1E6AxVKtyc
29. journalctl — Query the systemd journal – [žiūrėta 2013-03-29]. Prieiga per internetą: http://www.freedesktop.org/software/systemd/man/journalctl.html
30. Applied Research Definition – [žiūrėta 2013-04-02]. Prieiga per internetą: http://businessvocab.com/applied-research-definition

PRIEDAI

Su darbu yra pateikta kompaktinė plokštelė (CD), kurioje yra:

1. Šio darbo elektroninė versija ODT ir PDF formatais;
2. Tyrimuose naudotos programinės įrangos kodas;
3. Patvirtinimas iš įmonės apie šio darbo rezultatų pritaikymą ir naudingumą įmonės veikloje;