KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Deividas Inkratas

Signalizacijos įvykių apdorojimo sistemos programinės įrangos ir duomenų apsauga

Baigiamasis magistro darbas

Vadovas
Prof. Egidijus Kazanavičius

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

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

Signalizacijos įvykių apdorojimo sistemos programinės įrangos ir duomenų apsauga

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas

(parašas) Prof. Egidijus Kazanavičius
(data)

Recenzentas

(parašas) Doc. Agnius Liutkevičius

(data)

Projektą atliko
(parašas) Deividas Inkratas
(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Deividas Inkratas

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

„Baigiamojo projekto pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ____ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Deivido Inkrato**, baigiamasis projektas tema „Signalizacijos įvykių apdorojimo sistemos programinės įrangos ir duomenų apsauga“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

_____ _____

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

Inkratas, D. Signalizacijos įvykių apdorojimo sistemos programinės įrangos ir duomenų apsauga. Magistro baigiamasis projektas / vadovas Prof. Egidijus Kazanavičius; Kauno technologijos universitetas, informatikos fakultetas, Kompiuteriu katedra. Kaunas, 2017. 50 psl.

SANTRAUKA

Signalizacijos įvykių apdorojimo sistema – tai signalizacijos priedo sistema, kuri perima duomenis apie jutiklių būsenas iš apsaugos sistemos ir juos atvaizduoja vartotojui internetinėje svetainėje. Kad šios sistemos veikimo principas nebūtų pažeistas, kuriami saugos modeliai, kurie apsaugos nuo programišių. Kuriant šiuos saugos modelius išanalizuojamas sistemos veikimo principas, sistema suskirstoma į saugumo lygius: fizinis lygmuo, operacinės sistemos lygmuo, duomenų bazių lygmuo, tinklo lygmuo ir vartotojo lygmuo. Saugumo lygmenims išanalizuojami saugumo modeliai. Pagal pasirinktus kriterijus pritaikomas geriausias saugumo modelis. Fizinis lygmuo saugomas konfigūruojant programinį kodą ir prijungiant papildomą daviklį. Operacinės sistemos lygmeniui kuriama programa, kuri tikrins visus aktyvius procesus. Duomenų bazės lygmenyje koreguojamos funkcijos, kad parametrizuotų užklausas. Tinklo srautą apsaugai pasitelkiama *CloudFlare* tarnyba. Vartotojo lygmeniui apsaugoti sukuriamas papildomas kodas, kuris bus sugeneruotas ir išsiųstas vartotojui į elektroninį paštą. Kiekvienas saugumo modelis suprojektuojamas. Pateikiamos sekų ir veiksmų diagramos. Saugumo modeliai suprojektuojami, pateikiamos duomenų bazės diagramos. Kiekvieno saugumo modelio testavimui sudaromas testavimo modelis pagal kurį jie ištestuojami. Išvadose pateikiamas darbo rezultatas, dėsningumas ir pastabos.

Inkratas Deividas. Security event processing system software and data security. *Master's* thesis / supervisor assoc. Prof. Egidijus Kazanavičius; Department of Computer Science. Faculty of Informatics, Kaunas University of Technology. - Kaunas, 2017. 50 p.

SUMMARY

Security event processing system – is a security system additional device (system), which take data from security system, analyse and represent it for user in website. For this principle of operation of the system is not damaged, will be create security model that protects against hackers. The development of these models system principle of operation is analysed and divided system by layers of security: physical layer, operating system layer, database layer, network layer and user layer. Security layers are analysed and by chosen criteria the best security model for each layer is chosen. Physical layer is secured with additional sensor and programming software. Created program for operating system layer will monitor all active processes. Change database layer functions will parameter query. *CloudFlare* controls network traffic of network layer. In user layer we be generated additional code, which will be send to user email each time user will try to login to website. Provide action sequences and diagrams. Security models were designed, provided database diagrams. Testing models created for each security model. System was tested by those models. The conclusion describe work results, patterns and additional notes.

Turinys

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

SEPS - Signalizacijos įvykių apdorojimo sistema (angl. *Security event processing system*).

DDOS – Paskirstytas atsisakymas aptarnauti (angl. *Distributed denial of service*).

C-Bus – Protokolas skirtas bendrauti tarp signalizacijos pulto ir signalizacijos centralės.

OS – operacinė sistema.

ĮVADAS

Apsaugos sistemos nėra tobulos, todėl norint jas pagerinti yra kuriami ir integruojami moduliai. Norint pagerinti *Paradox* apsaugos sistemą sukurtas modulis, kuris fiksuoja apsaugos sistemos ir jos zonų būsenas, bei pateikia duomenis internetu, kadangi tokios galimybės *Paradox* apsaugos sistema neturi.

Signalizacijos įvykių apdorojimo sistema savyje kaupia konfidencialią informaciją. Pakliuvus šiai informacijai į piktavalių rankas vartotojams gali būti padaryta daug žalos, todėl reikia užtikrinti sistemos saugumą. Prie šios sistemos galima prisijungti tiek fiziniais veiksniais (jungiantis tiesiai prie pačios sistemos), tiek programiškai - per atstumą. Dėl to saugumo priemonių reikia imtis kelių tipų: fiziniais veiksniais, programiniais veiksniais bei trečių šalių programų pagalba. Saugojant šią sistemą bus orientuojamasi į keturis esminius dalykus: *Signalizacijos įvykių apdorojimo sistemos* kodo apsaugą, bendravimą tarp dviejų sistemų (*signalizacijos įvykių apdorojimo sistemos* ir signalizacijos sistemos), vartotojo duomenų saugumą bei sistemos pažeidžiamumą.

Pagrindiniai šio darbo tikslai:

- Išanalizuoti *Signalizacijos įvykių apdorojimo sistemą;*
- Sukurti sprendimo metodą, kuris apsaugotų atminties kortelėje esančius failus (sistemos programinį kodą);
- Sukurti sprendimo metodą, kuris apsaugotų operacinę sistemą, kad joje neveiktų pašaliniai procesai;
- Sukurti sprendimo metodą, kuris apsaugotų *Signalizacijos įvykių apdorojimo sistemos* tinklą (apsaugoti nuo atakų einančių per tinklą tokių kaip DDOS);
- Sukurti sprendimo metodą, kuris sustiprintų vartotojo prisijungimą prie internetinės svetainės;
- Sukurti sprendimo metodą, kuris apsaugotų sistemą nuo prisijungimo fiziškais veiksniais;
- Realizuoti ir ištestuoti visus sprendimo metodus.

Darbo struktūra. Šis dokumentas sudarytas iš aštuonių pagrindinių dalių. Pirmoje dalyje pateiktas įvadas, kuriame supažindinama su *Signalizacijos įvykių apdorojimo sistema*. Antroje dalyje pateiktos dvi analizės. Vienoje išanalizuota *Signalizacijos įvykių apdorojimo sistema* jos saugumo spragos, o antroje pateikiamos tų spragų sprendimo būdai. Trečioje dalyje pateiktas saugumo spagų realizavimo kūrimas, atvaizduotos veiksmų ir sekų diagramos. Detalizuoti sprendimo metodai, kuriuos pasirinkome analizės metu. Ketvirtoje dalyje pateiktas sistemos vaizdas ir realizuoti sistemos saugumo metodai. Penktoje dalyje testuojami realizuoti metodai ir pateikiamos testavimo išvados. Šeštoje dalyje

pateikiama informacija apie gautų dokumentacijoje rezultatų apibendrinimą ir suformuotos išvados. Septintoje dalyje pateiktas literatūros sąrašas. Aštuntoje pateikti priedai.

1. SIGNALIZACIJOS ĮVYKIŲ APDOROJIMO SISTEMOS PROGRAMINĖS ĮRANGOS IR DUOMENŲ APSAUGOS ANALIZĖ

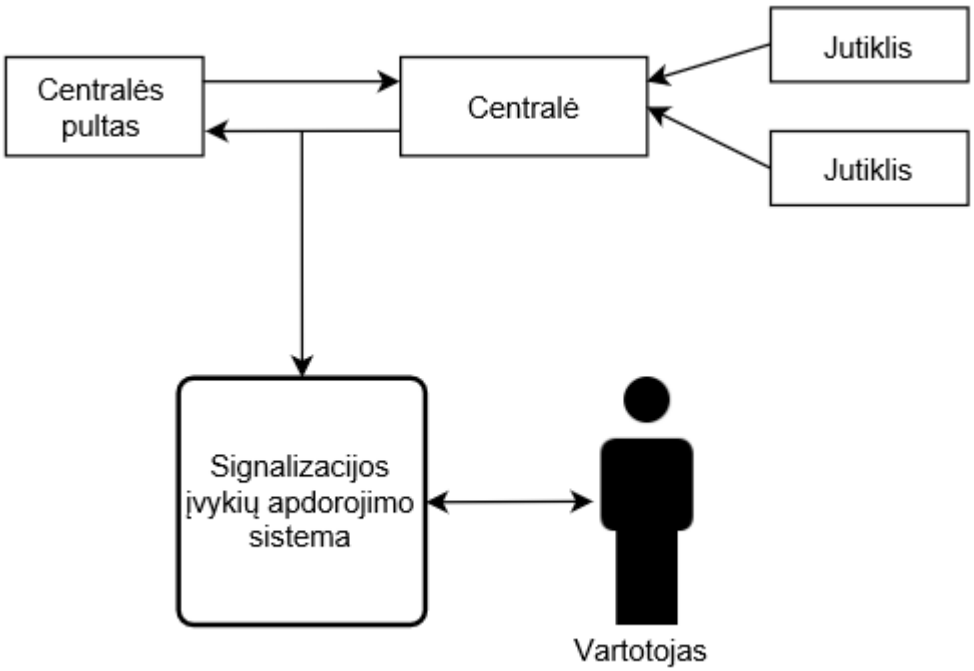
Šiame skyriuje bus pateiktas *Signalizacijos įvykių apdorojimo sistemos* veikimo principas ir analizė. Analizė susideda iš dviejų poskyrių:

- sistemos analizė, kurioje bus analizuojamos spragos esančios sistemoje ir galinčios pakenkti duomenims ar sistemos darbui;
- sprendimo metodų analizė.

1.1. *Signalizacijos įvykių apdorojimo sistemos* analizė

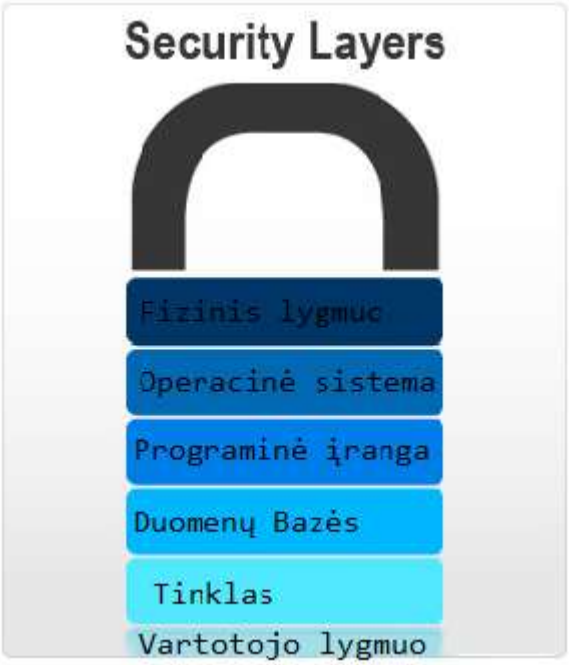
Signalizacijos įvykių apdorojimo sistemos, struktūra pateikta 1.1 pav, (angl. *Security even processing system,* toliau bus vartojama kaip SEPS) susideda iš kelių komponentų: mikrokompiuterio, SD kortelės ir pačios signalizacijos sistemos (į kurią įeina centralė, pultas ir davikliai). Norint sužinoti saugumo spragas pačioje sistemoje reikia išnagrinėti veikimo principą ir visus saugumo lygmenis.

Sistemos veikimas prasideda kai suveikia kuris nors signalizacijos daviklis. Tada jutiklis siunčia duomenis į centralę tam tikru užkoduotu protokolu. Po to centralė siunčia duomenis į klaviatūrą ir į SEPS, per neapsaugotą *C-Bus* protokolą. Patekę į SEPS duomenys yra išanalizuojami ir įkeliami į SD kortelę, kurioje yra visa duomenų bazė, analizavimo programa ir internetinė svetainė, kurioje atvaizduojami duomenys vartotojui. Vartotojas prisijungęs prie internetinės svetainės gali pamatyti signalizacijos sistemos veikimo istoriją ir dabartinę būseną. Visas sistemos darbas vyksta realiu laiku ir svarbu, kad niekas šios sistemos neapkrautų pašaliniais veiksniais, nes tai gali sutrukdyti sistemos darbą. Paveikus sistemą iš šalies gali nukentėti veikimo kokybė. Dėl to reikia apsaugoti ne tik pačius duomenis, bet ir pačią sistemą.



1.1 pav. *Signalizacijos įvykių apdorojimo sistema*

Norint sustiprinti sistemos saugumą reikia atkreipti dėmesį į visus saugumo lygmenis, kurie pavaizduoti 1.2 pav. Operacinės sistemos lygmenyje reikia apsaugoti, kad programišiai neįdiegtų pašalinių programų, kurios galėtų pažeisti sistemos darbui. Visi vartotojų duomenys yra saugojami duomenų bazėje. Norint užtikrinti konfidencialumą reikia apsaugoti pačią duomenų bazę ir vykstančias transakcijas. Tinklo lygmuo yra menkai apsaugotas, todėl prie svetainės prisijungti gali bet kas, kas turi prisijungimo duomenis (tai nėra labai saugu, kai naudojami konfidencialūs duomenys, susiję su vartotojų turtu). Tuo labai patogu pasinaudoti kenkėjams ir pasisavinti vartotojų informaciją. Taip pat labai svarbu apsaugoti programinę įrangą, kad kenkėjai nekopijuotų sistemos, nes išanalizavę sistemos veikimo principą jie gali pasisavinti SEPS sistemos duomenis. Vartotojo lygmenį reikia apsaugoti tam, kad vartotojas neatliktų neleistinų veiksmų, kurie galėtų pakenkti sistemos veikimui. Fizinis lygmuo gali sistemai atnešti didžiausių nuostolių, dėl to jam reikia skirti daugiausia dėmesio.



1.2 pav. Saugumo lygmenys

1.1.1. Operacinės sistemos lygmuo

Naudojama *Raspbian* operacinė sistema (toliau bus naudojama kaip *OS*). Ši OS turi nemažai apsaugų, bet prasibrovus įmanoma įdiegti savo piktavališką programą, kuri gali pakenkti sistemos darbui. Jeigu įdiegta programa turės vartotojo teises - daug veiksmų atlikti negalės. Viskas kuo gali pakenkti yra sistemos resursų atėmimas, dėl kurių sistema gali nebeveikti. Jeigu piktavališka sistema bus įdiegta su administratoriaus teisėm - visos „durys“ tampa atviros ir taip gali ne tik stipriai pakenkti sistemai, bet ir pasisavinti vartotojų konfidencialius duomenis.

1.1.2. Duomenų bazių ir transakcijų lygmuo

Visa vartotojų informacija saugoma duomenų bazėje. Duomenų bazė patalpinta atminties kortelėje. Atminties kortelė nėra apsaugota ir visi joje esantys duomenys yra atviri. Dėl šios priežasties SD kortelės yra nesaugios ir bet kas gali nusikopijuoti ir pasisavinti duomenis. Geriausias būdas apsaugoti SD kortelės duomenis - juos užšifruoti. Yra daug šifravimo programų, kurių kiekviena turi savo privalumų ir trūkumų.

Duomenų bazių lygmenį reikia apsaugoti nuo neleistinų vartotojo veiksmų. Vartotojas ar įsibrovėlis pasinaudojęs prieiga prie internetinės svetainės gali pakenkti sistemai tokiais būdais kaip SQL injekcijos atakomis.

SQL injekcijos atakos – tai tokios kodo atakos technikos, kai puolamos duomenų saugyklos su papildoma kodo eilute, kuri būna įrašyta į įvedimo laukelį [16][17]. Pavyzdžiui - surandama silpna vieta interneto svetainėje, kurioje kreipiamasi į duomenų bazę (pavyzdžiui „Paieškos laukelis“,

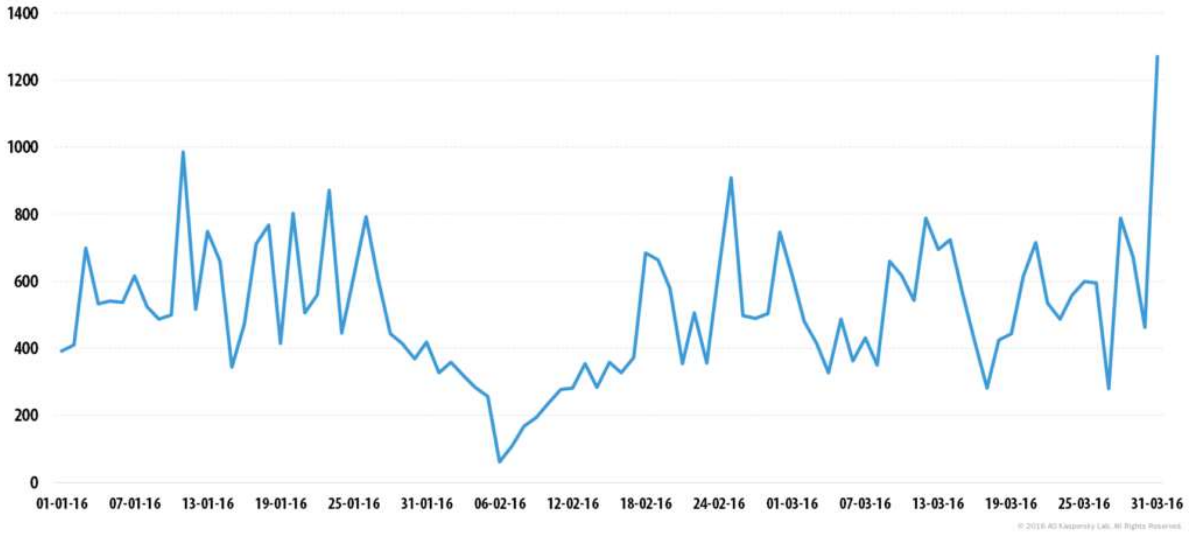
„Prisijungimo langas“ ir kt.), kurioje pasitaiko paprastos „Execution“ ar „Select“ funkcijos prie kurių pridėjus papildomą kodo eilutę galima ištraukti naudingus duomenis. Pavyzdžiui, prie prisijungimo lauko pridedamos kabutės, kad uždarytų vedamą teksto (angl. *String*) laukelį ir pridedame „or 1=1“, kad prijungtų mus prie sistemos, nors esame neregistruoti vartotojai, tokiu būdu prisijungsime prie sistemos. Čia vienas iš daugelio būdų, kaip galima pasinaudojus SQL injekcijos atakomis įsibrauti į sistemą arba pasisavinti informaciją[20][21].

Norint apsisaugoti nuo injekcijos atakų reikia imtis kelių veiksmų, kurie sustiprins apsaugą nuo įsilaužėlių:

1. Visada reikia suteikti vartotojams, kuo mažiau privilegijų. Vartotojas iš duomenų bazės turi galėti išsitraukti tik jam būtinas reikšmes, kad nematytų tarpinių veiksmų ir informacijos.
2. Visada tikrinti informaciją gaunamą į serverį. Ji gali būti iš serverio arba iš paties vartotojo, kai bandoma pateikti tam tikrą informaciją serveriui.
3. Negrąžinti vartotojui SQL klaidų žinučių. Pagal klaidų informaciją įsilaužėliai gali gauti detalią informaciją apie SQL lentas, užklausas ar pačią informaciją. Tai paprastai išsprendžiama su „Catch“ funkcija.
4. Būtina nurodyti visų įvedamų reikšmių tipus. Nurodant tipą vartotojas negali įrašyti nepageidautinų simbolių ir reikšmių, kurios gali pakenkti serverio greitaveikai. Tačiau reikia įvertinti, kad buferis nebūtų perpildytas
5. Užkoduoti tekstinius įvesties laukelius, kuriuose gali būti ženklų naudojant funkciją *Base64*
6. Filtruoti įvedamą informaciją dviem žingsniais. Pirmiausia svetainėje ties vartotojo įvesties laukais pritaikyti baltojo sąrašo (angl. *white-list*) filtravimą. Tai leidžia vartotojui įvesti tekstą tik tam tikrais formatais (*data, String* formatas, *integer* ir kt.) ir tik tam tikrą ženklų skaičių. Antras žingsnis – juodojo sąrašo (angl. *black-list*) filtravimas jis taikomas SQL serveryje duomenų gavimo lygmenyje prieš vykdant SQL užklausą. Šis filtras neleis naudoti dvigubų užklausų ir nepageidaujamų ženklų
7. Naudoti dinamiškai sugeneruotus duomenų bazės objektus su baltojo sąrašo filtru.
8. Vengti naudoti identifikatorius, kurie galėtų komplikuoti baltojo ir juodojo sąrašo filtrus. (pvz., vartotojų ID susikurti leisti tik raides ir skaičius, nes leidžiant naudoti tam tikrus ženklus negalima bus jų nufiltruoti, nes jie yra būtini)

1.1.3. Tinklo lygmuo

Pastaruoju metu vis daugėja įsilaužimų per tinklą. Atsižvelgiant į statistiką DDOS (angl. *Distributed Denial of Service,* liet. paskirstytas atsisakymas aptarnauti) atakos vis populiarėja ir jos pasikartoja vis dažniau. DOS atakų skaičius laiko grafike atvaizduotas 1.3 pav.



1.3 pav. DDOS atakų kiekis per 2016m pirmą ketvirtį

DDOS - tai būdas kenkti serveriui, kad jis nepajėgtų aptarnauti vartotojų užklausų. Serveriui yra pasiunčiama daug užklausų iš skirtingų kompiuterių kurių serveris nebespėja apdoroti ir išsiųsti atsakymų.

Visiškai apsisaugoti nuo DDOS atakų neįmanoma. Tačiau kuo daugiau resursų investuojama į apsaugą nuo DDOS, tuo mažesnė tikimybė, kad serveriui bus pakenkta. Vienas iš būdų investuoti kuo mažiau resursų į sistemos apsaugą, bet užtikrinti jos saugumą, yra valdyti ateinantį srautą. Tai yra sustabdyti didelius sinchroninius, *TCP, UDP, ICMP* srautus, neleisti lėtų *HTTP* užklausų (*GET&POST* atakos). Yra keletas jau sukurtų nemokamų programų, kurios filtruoja tokius srautus.

1.1.4. **Vartotojo lygmuo**

Dažnai pasitaikanti problema, dėl kurios kenkėjai prasibrauna į vidų - gauti prisijungimo duomenys. Dėl to norint apsaugoti vartotojo duomenis reikia apsaugoti prisijungimą. Apsaugoti prisijungimą geriausia gavus laikinus prisijungimo duomenis tada jie galiotų tik tam tikrą laiko tarpą ir būtų galima prisijungti tik vieną kartą. Kad laikinas slaptažodis būtų sugeneruotas reikia į sistemą įkelti slaptažodžių generatorių, kuris reaguotų į trumpąją žinutę arba siųstų į vartotojo paštą.

Taip pat sistemą vartotojo lygmenyje reikia apsaugoti nuo neleistinų vartotojo veiksmų. Vartotojas ar įsibrovėlis pasinaudojęs prieiga prie internetinės svetainės gali pasisavinti kito vartotojo sesiją tokiais būdais kaip sesijos užgrobimo (angl. *Session Hijack*) atakomis.

Sesijos užgrobimo atakos – tai tokios atakos, kai įsilaužėlis pasinaudojęs kito vartotojo sesija, prisijungia į sistemą. Tokiu atveju įsibrovėlis turi gauti slapuko (angl. *cookie*) informaciją

(dažniausiai ją gauna priversdamas vartotoją prisijungti prie nepatikimos svetainės), kurią įsilaužėlis gali panaudoti ir prisijungti to vartotojo vardu [18]. Pavyzdys:

1) vartotojas turi paskyrą prie sistemos http://www.sistematestas.lt;
2) įsilaužėlis nusiunčia laiškelį vartotojui, kad patikrintų naujus galimus veiksmus sistemos svetainėje http://www.sistematestas.lt/?SID=žinomas_id
   Įsilaužėlis išsisaugoja sesijos ID;
3) vartotojas susidomėjęs nueina į įsilaužėlio atsiųstą adresą, jam iššoka prisijungimo langas, jis prisijungia. Ir taip įsilaužėlis gauna vartotojo prisijungimo sesiją;
4) įsilaužėlis prisijungia prie http://www.sistematestas.lt/?SID=žinomas_id ir taip turi visą prieigą prie vartotojo paskyros.

1.1.5. Programinės įrangos lygmuo

Programinę įrangą svarbu apsaugoti, kad niekas nekopijuotų ir neplatintų sistemos. Ją apsaugoti užteks pasinaudoti atminties kortelės šifravimo metodu.

1.1.6. Fizinis lygmuo

Mikrokompiuteris su centrale bendrauja per *C-Bus* protokolą. *C-Bus* protokolas bendrauja per 5 laidus. Dažniausiai naudojamas šviesos reguliavimui namuose arba apsaugos sistemose bendrauti tarp jos komponentų (pavyzdžiui tarp centralės ir pulto). Šiuo atveju *C-Bus* protokolu bendrauja centralės pultas ir *Signalizacijos įvykių apdorojimo sistema*.

Šis bendravimas yra visiškai neapsaugotas ir bet kas prijungęs kenkėjišką prietaisą gali nuskaityti bendravimą tarp abiejų sistemų. Galimi keli šios spragos sprendimo būdai: konfigūruojant kodą arba fiziniais veiksniais apsaugojus priėjimą prie pačios sistemos.

1.2. Sprendimo metodų analizė

1.2.1. Atminties kortelės šifravimas

Atminties kortelė, kartu ir visi duomenys esantys viduje, šifruojami su algoritmu. Algoritmo sprendimo variantų yra daug. Tam, kad išsirinkti geriausią bus išanalizuoti keletas rastų algoritmo variantų.

O***penSSL*** – tai atvirojo kodo projektas, kuris paremtas *SSL* ir *TLS* protokolais. Pagrindinės bibliotekos parašytos C programavimo kalba. *OpenSSL* turi pagrindines kriptografijos funkcijas ir turi daug kitų naudingų funkcijų. *OpenSSL* turi didelį privalumą, nes palaiko daug programinių kalbų dėl to gali būti naudojamas daugelyje operacinių sistemų (*Windows*, *Solaris*, *Linux*, *Mac OSX*, *BSD*, *OpenVMS*, *System I* (OS/400)). Jis palaiko daugumą šifravimo algoritmų tokių kaip *AES128*, *AES256*, *DES3* ir t.t. *OpenSSL* yra saugomas dviejų licencijų *OpenSSL* licencijos ir *Ssleay* licencijos. *OpenSSL*

licencija yra *Apache 1.0* licencija. Abi licencijos leidžia pilnai naudoti programą ir ne komerciniais ir komerciniais tikslais.[1][2]

Beringso – tai atvirojo kodo projektas, paremtas *OpenSSL* programa. Labiau skirtas naudoti „Google“ kompanijos naudotojams bei „Google“ produkcijos kūrėjams. Pašaliniams žmonėms nerekomenduojama naudotis, API (angl. *application programming interface*) ar ABI (angl. *application binary interface*) stabilumo garantijos. Taip pat *Beringso* savininkai „Google“ kompanija leidžia bet kada koreguoti kodą pagal jų poreikius, be jokių įspėjimų.[3]. Ši programinė įranga palaiko tokius pat šifravimo algoritmus kaip ir *OpenSSL*.

LibreSSL – tai atvirojo kodo projektas paremtas *OpenSSL* programa. Šis projektas skirtas ištaisyti klaidas kurias paliko *OpenSSL* kūrėjų kompanija, modernizuoti kodo struktūrą, patobulinti apsaugą ir optimizuoti procesus. Visas kitas principas ir programinių kalbų palaikymas toks pat kaip ir *OpenSSL*[4]. Šis projektas leidžia naudoti tokius pat šifravimo algoritmus kaip ir *OpenSSL*.

eCryptfs – tai kriptografinių failų sistema skirta *Linux* Operaciniai sistemai. Ši programa saugo kriptografinius duomenis kiekvieno failo antraštėje, dėl to užšifruoti failai gali būti kopijuojami tarp savininkų. Failas atšifruojamas su tam tikru raktu, kuris yra *Linux Kernel* raktų rinkinyje [24]. Todėl nereikia sekti papildomos informacijos, užtenka to kas yra užšifruotame faile. Ši sistema šifruoja *AES128* algoritmu 16 ženklų raktu. C*ryptfs* labiausiai paplitęs „Ubuntu“ kai norint užšifruoti namų (angl. *home*) direktoriją.[5][6][7]

TrueCrypt – tai programa skirta užšifruoti diską ar jo dalį. Ši programa nebėra pardavinėjama ar tobulinama, bet ją, vis dar veikiančią galima parsisiųsti iš oficialios svetainės. Ji pritaikyta dirbti *Windows* aplinkoje, bet taip pat palaiko ir *Linux* bei *Mac OS X*. *TrueCrypt* nėra atviro kodo programa ją saugo *TrueCrypt* licencijos. Ji turi dauguma šifravimo algoritmų kaip ir *OpenSSL*. Jos veikimo principas - sukurti naują užšifruotą particiją (disko dalį), kurioje galima talpinti duomenis, kurie bus šifruojami.[8][9]

EncryptedHome – tai funkcija kuria leidžia naudotis *Ubuntu*. Ši funkcija leidžia užšifruoti tam tikrą aplanką (namų direktoriją). Norėdami prisijungti prie užšifruotos direktorijos vartotojui reikia tiesiog prisijungiant įrašyti atšifravimo slaptažodį. Ši funkcija neleidžia užšifruoti visos SD kortelės.[10] Ši funkcija naudoja *AES 128* algoritmą.

Cryptsetup – tai *Raspbian* funkcija, kuri leidžia užšifruoti particija, Šiai funkcijai patartina naudoti atskirą kompiuterį (ar kitą sistema su *Linux* operacine sistema) su kuriuo reiktų užšifruoti dalį disko. Užšifruota particija negali būti užkeliama kaip atvaizdas (angl. *Mount image*). Atšifravimas nėra trumpas, tai gali užtrukti šiek tiek ilgiau nei naudojantis kitomis priemonėmis.[11][12]. *Cryptsetup* palaiko dauguma šifravimo algoritmų *AES, DES, twofish* ir kt.

Luks – tai *Raspbian cryptsetup* papildomas įrankis, kuris orientuojasi į disko ar dalies jo šifravimą. Su šiuo įrankiu lengviau užšifruojamas diskas. Norinti atšifruoti reikia remtis slaptažodžiu

kurį naudojome šifruodami. Trūkumas - pirmiau reikia užšifruoti tam tikrą diską ar jo dalį ir tik paskui jame talpinti duomenis. Šis įrankis ištrina buvusią informaciją. Šifravimo algoritmai tokie patys kaip ir *cryptsetup*. [13][14]

1.2.2. Apsauga nuo DDOS atakų

Vis labiau populiarėja DDOS atakų ir vis su galingesniais kompiuteriais jos organizuojamos, dėl to visiškai nuo jų apsisaugoti nėra įmanoma, bet kadangi mūsų sistema nėra didelė ir prie sistemos jungiasi mažas vartotojų kiekis, įsivedėme kriterijų, kad sistema galime laikyti saugia jeigu sistema sugeba susidoroti su 30 vartotojų, kurie jungiasi vienu metu.

Guardian - Viena iš tokių programinių įrangų yra *Guardian*, tai mokama programa, kuri kontroliuoja srautą, taip tikrindama ar nevyksta *DDOS* atakos. Ši programa apsaugo nuo sinchroninių srautų, lėtų užklausų, brutalios jėgos atakų (kai spėliojami slaptažodžiai), galimi individualūs *IP* adresų draudimai (ar net visos šalies) ir viskas vyksta realiu laiku. Gamintojai teigia, kad ji atlaiko iki dvidešimties tūkstančių (20 000) atakuotųjų vienu metu. Vienos licencijos kaina 100 JAV dolerių.

Snort - tai programinė įranga, kuri skenuoja tinklo srautą (pasitelkdama WinPcap) ir pagal taisykles (kurias galima parsiųsti iš internetinės svetainės arba susikurti pačiam) priima veiksmus. Pagal tas taisykles yra atpažįstama kokia ataka yra puolamas serveris ir imamasi reikiamų priemonių. *Snort* turi dinamines taisykles, jos leidžia, kad viena taisyklė iškviestų kitą taisyklę tam tikrais atvejais. Tai labai patogu, nes galima pagal tam tikrus kriterijus parinkti kurią taisyklę taikyti. Ši programinė įranga taip pat gali kaupti istorija, kuri padeda išanalizuoti, kaip ateityje apsisaugoti nuo tokių atakų. Ji veikia realiu laiku ir neapkrauna sistemos. Pati programinė įranga yra nemokama taip pat ir taisyklės, bet kaip žinome, kad įsilaužėliai vis naudoja įmantresnius būdus prasibrauti į sistema ir taisyklių sąrašą vis reikia papildyti. Dėl šios priežasties *Snort* programinės įrangos gamintojai apmokestina naujausias taisykles, kurios kaina 29.99 JAV dolerių per metus.

Suricata – tai nemokama programinė įranga, kuri skenuoja tinklo srautą pagal taisyklių sąrašą ir atitinkamai į jas reaguoja. Fiksuoja visus pažeidimus ir įrašo pasirinktą failą. Veikimo principas panašus į *Snort* programinę įranga.

Mod_Evasive - tai papildomas modulis *Apache* serveriui. Jis tikrina ateinantį srautą ir pagal aprašytus kriterijus blokuoja IP adresus tam tikram laiko tarpui. Priklausomai nuo serverio lankomumo dydžio galima nurodyti kiek laiko turi praeiti tarp vartotojo kreipimosi į serverį, kad jis galėtų kreiptis antrą kartą. Taip pat galima prisitaikyti, pagal serverio pajėgumą, kiek laiko gali užtrukti viena užklausa. Atitikęs, bent į vieną iš šių kriterijų, IP adresas yra blokuojamas tam tikrą laiko tarpą, kuris nurodytas nustatymuose (juos galima keisti) [15].

CloudFlare teikia tinklo saugos paslaugas vartotojams. Ši tarnyba yra tarpinis kelias tarp mūsų svetainės ir vartotojo, kuris nori tą svetainė pasiekti. Kai vartotojas bando prisijungti prie svetainės jį nukreipia į tarptautinį serverių tinklą. *Cloudfalare* turi didelį paketą taisyklių, kuriose aprašytos visos dažniausiai pasitaikančios atakos [22]. Pagal tas taisykles yra skenuojamas ateinantis tinklo srautas. Šis srautas neatitinkantis taisyklių reikalavimų, yra blokuojamas [23]. Jų teikiamos paslaugos pagrindiniai pliusai:

- infrastruktūra yra pakankamai galinga, teigiama , kad gali apsaugoti nuo didžiausios DDOS atakos, kuri buvo pasirodžiusi iki šiol;
- jeigu aptinkama nauja ataka bent vienoje svetainėje, nuo šios atakos saugojamasi ir visose kitose svetainėse (taisyklės ištisai atnaujinamos);
- lengvas konfigūravimas SSL;
- lengvas įdiegimas;
- kaina priklauso nuo paketo dydžio (mažiausias paketas nemokamas).

Kadangi šis serveris neturi didelio vartotojų kiekio, tai mažiausias paketas, kurį siūlo *CloudFlare* yra nemokamas ir patenkina mūsų lūkesčius.

1.2.3. Apsauga nuo įsibrovimo į sistemą

Apsisaugoti nuo įsibrovimo į SEPS sistemą per patį mikrokompiuterį galima dviem variantais: uždrausti priėjimą prie pačios sistemos (fiziniais veiksniais) ir į pačią sistemą įdiegti tikrinimą (kodo konfigūravimas).

1.2.3.1. Kodo konfigūravimas

Šiuo metu kodas neturi jokios apsaugos nuo įsilaužėlių. Kodas skirtas tik gauti duomenims juos išanalizuoti ir įkelti į duomenų bazę. Norinti išvengti įsilaužimo į bendravimą reikia konfigūruoti kodą, įterpti kodo dalį, kuri tikrintų gaunamus duomenis ir, jei būtų bent akimirkos nutrūkimas, nutrauktų sistemos darbą. Kadangi sistema veikia realiu laiku ir kiekviena akimirka siunčiami tam tikri duomenys (bitai), tai bent vieno bito neatitikimas reiškia, kad sistemoje įvyko trikdis (gali būti pažeidžiama sistema arba prie laidų prijungtas kenkėjiškas daiktas)

1.2.3.2. Fiziniais veiksniais

Bendravimą galima apsaugoti per *C-BUS* spragą apsaugojant priėjimą prie sistemos. Jeigu neleisime prie sistemos prieiti kenkėjui - į ją įsilaužti negalės. Kai *Signalizacijos įvykių apdorojimo sistema* veikia ją saugoją jutiklis, kuris praneša jeigu kas nors atidaro dėžę, kurioje yra sistema bei centralė.

1.2.4. Operacinės sistemos lygmens apsauga

Iš atliktų grėsmių analizės matome, kad *Raspbian* operacinei sistemai papildomos įdiegtos programos yra pavojingos. Jos savarankiškai atlieka papildomus veiksmus, kurie kenkia sistemos darbui. Dažniausiai piktavaliai paleidžia programas su automatiško įsijungimo funkcija, todėl netgi išjungtos po kurio laiko jos vėl įsijungia. Tokias piktaviliškas programas reikia pašalinti su visomis šaknimis t.y. surasti kaip ji atsirado, užkirsti tam kelią, kad daugiau neatsirastų ir ištrinti pačia programa ir visus susijusius failus su ta programa.

1.3. Analizės išvados

Pagal atliktą analizę matome 1 lent., kad visiems saugumo lygmenims reikės skirti papildomą dėmesį.

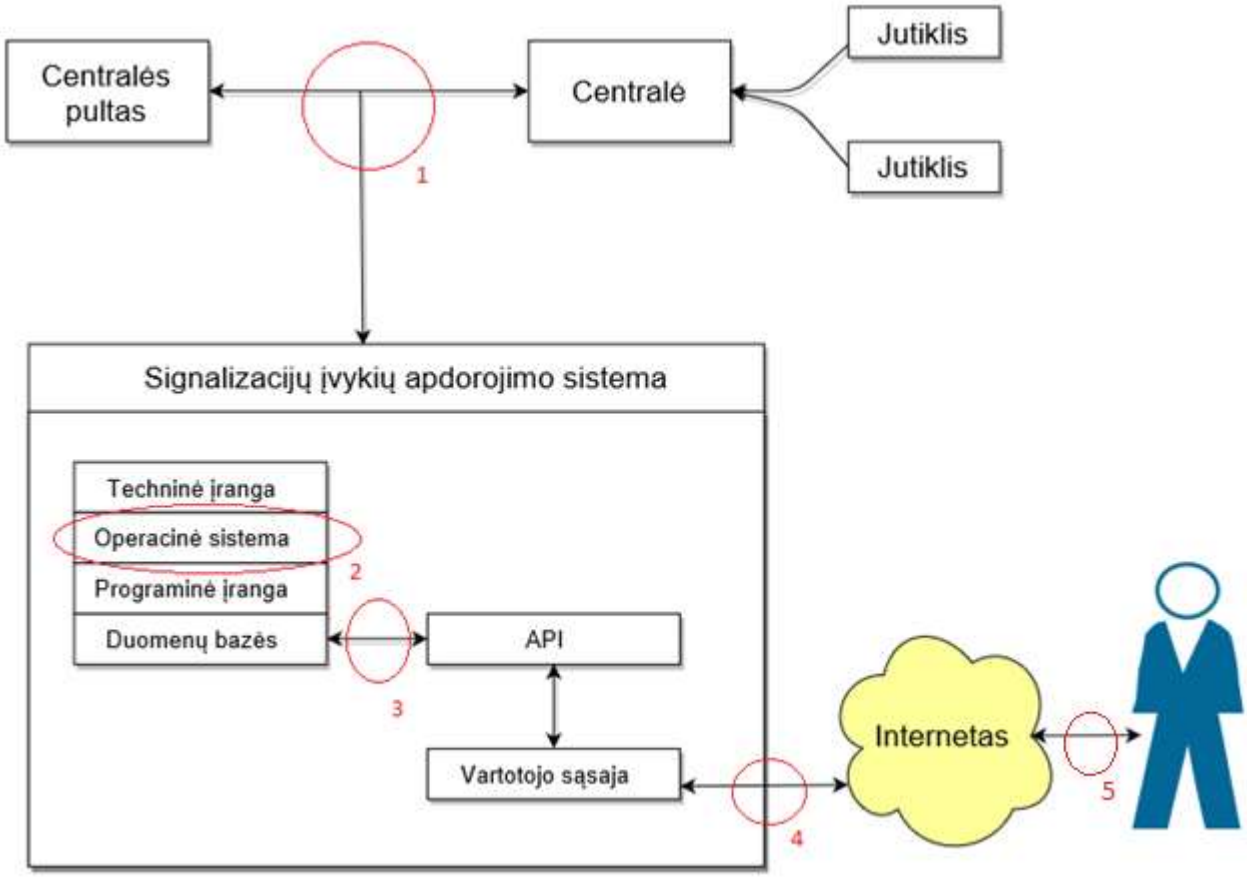
1.1 lentelė Analizės išvados

Saugumo lygmuo	Saugumo spraga	Sprendimo būdas
Operacinės sistemos lygmuo	Įdiegta kenkėjiška programinė įranga	Stebėti veikiančias programas ir kenkėjiškas išjungti (naikinti)
Duomenų bazės ir transakcijų lygmuo	Priėjimas prie duomenų bazės ir pačių duomenų atviri	Šifruoti visą atminties kortelę. Riboti vartotojo vedamus ženklus, pakeisti kreipimosi į SQL DB metodą.
Tinklo lygmuo	Jokios apsaugos prieš DDOS atakas	Naudosime CloudFlare
Programinės įrangos lygmuo	Programinės įrangos kodas atviras ir pati programa atvira	Šifruoti visą atminties kortelę
Vartotojo lygmuo	Vartotojo prisijungimas galioja visam laikui, nekeičiamas	Sukurti laikino kodo generatorių,
Fizinis lygmuo	Neapsaugotas fiziniais veiksmais kodo nuskaitymas	Konfigūruoti kodą. Pridėti papildomą daviklį

Operacinės sistemos lygmeniui apsaugoti sukursime papildomą programą, kuri susidoros su tokiomis programomis. Atminties kortelės šifravimui, pagal iškeltus kriterijus, šifravimo įrankiu išsirinkome *eCryptfs*. Tinklo apsaugojimui naudosime programinę įrangą *CloudFlare.* Vartotojo lygmeniui apsaugoti naudosime laikino slaptažodžio generatorių, kuris galios tik tam tikrą laiko tarpą ir tik vieną kartą. Apsaugoti sistemą fiziniame lygmenyje bus naudojamas kodo patobulinimas ir papildomas jutiklis, kurie praneš vartotojui apie kėtinimus įsilaužti.

2. SIGNALIZACIJOS ĮVYKIŲ APDOROJIMO SISTEMOS PROGRAMINĖS ĮRANGOS IR DUOMENŲ APSAUGOS REALIZAVIMAS

Išanalizavus sistemos trūkumus, nustatyti geriausi būdai šalinti spragas. Pirma spraga - įsibrovimas į bendravimą tarp *Signalizacijos įvykių apdorojimo sistemos* ir signalizacijos centralės. Ją galima išspręsti dvejais būdais: pasinaudojant signalizacijos davikliu ir konfigūruojant *SEPS* programinį kodą. Antra spraga - kenkėjiškos programos, įdiegtos į operacinę sistemą, ši problema sprendžiama sukuriant programą, kuri neleis veikti pašalinėms programoms. Trečia spraga atminties kortelės duomenų nesaugumas, kuri šalinama naudojant šifravimą su *eCryptfs* įrankiu. Ketvirta spraga tai visas tinklo srautas einantis į internetinę svetainę. Tai išsprendžiama su *CloudFlare* tarnyba. Penkta spraga, kur jai reikia papildomos apsaugos, tai vartotojo prisijungimas prie internetinės svetainės, kur talpinami duomenys. Šią problemą galima išspęsti su papildomu kodų generavimu. Visų šių saugumo spragų vieta atvaizduota 2.1 pav.



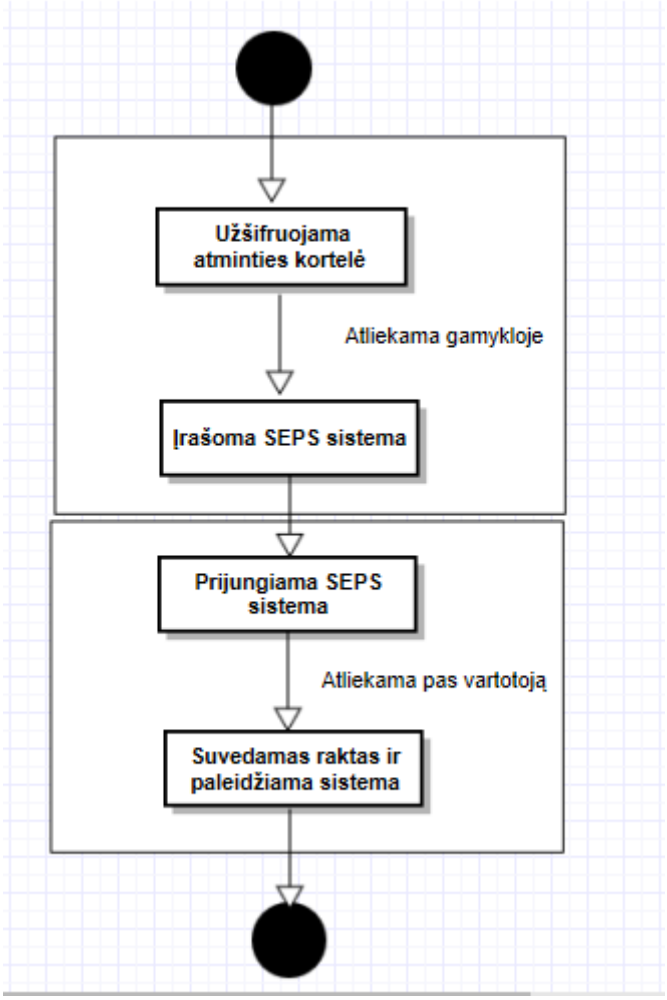
2.1 pav. *Signalizacijos įvykių apdorojimo sistemos* saugumo metodų modelis

2.1. Duomenų laikmenos šifravimas

Pati svarbiausia vieta, kurią reikia apsaugoti *SEPS* sistemoje, yra duomenų laikmenos (SD atminties kortelės) apsauga . Ten laikomas visas programinis kodas. Šiuo metu jis yra prieinamas visiems kas turi kaip nuskaityti atminties kortelę. Norinti apsaugoti šią vietą pasitelkiamas pagalbinis *Linux* įrankis *eCryptfs* ir jo pagalba užšifruojami duomenys atminties kortelėje.

Su *eCryptfs* užšifruojama vieno vartotojo visa namų direktorija. Su šiuo vartotojo bus paleidžiamos visos programos (signalizacijos įvykių apdorojimo programa, procesų tikrinimo programos ir kt.). Šifravimas vyks su slaptažodžio pagalba. Slaptažodis bus suteikiamas SEPS sistemos administratoriui, tai gali būti pats vartotojas arba už saugumą atsakingas žmogus. Slaptažodis pateikiamas ne elektroniniu variantu (taip saugiau).

Administratorius sudiegia ir užtikrina *Signalizacijų įvykių apdorojimo sistemos* palaikymą, todėl šifravimo raktą bus galima susigražinti tik jam. Slaptažodis yra saugomas atskiroje duomenų bazėje. Pačios sistemos veiksmų diagrama šiek tiek pasikeis (pavaizduota 2.2 pav.).



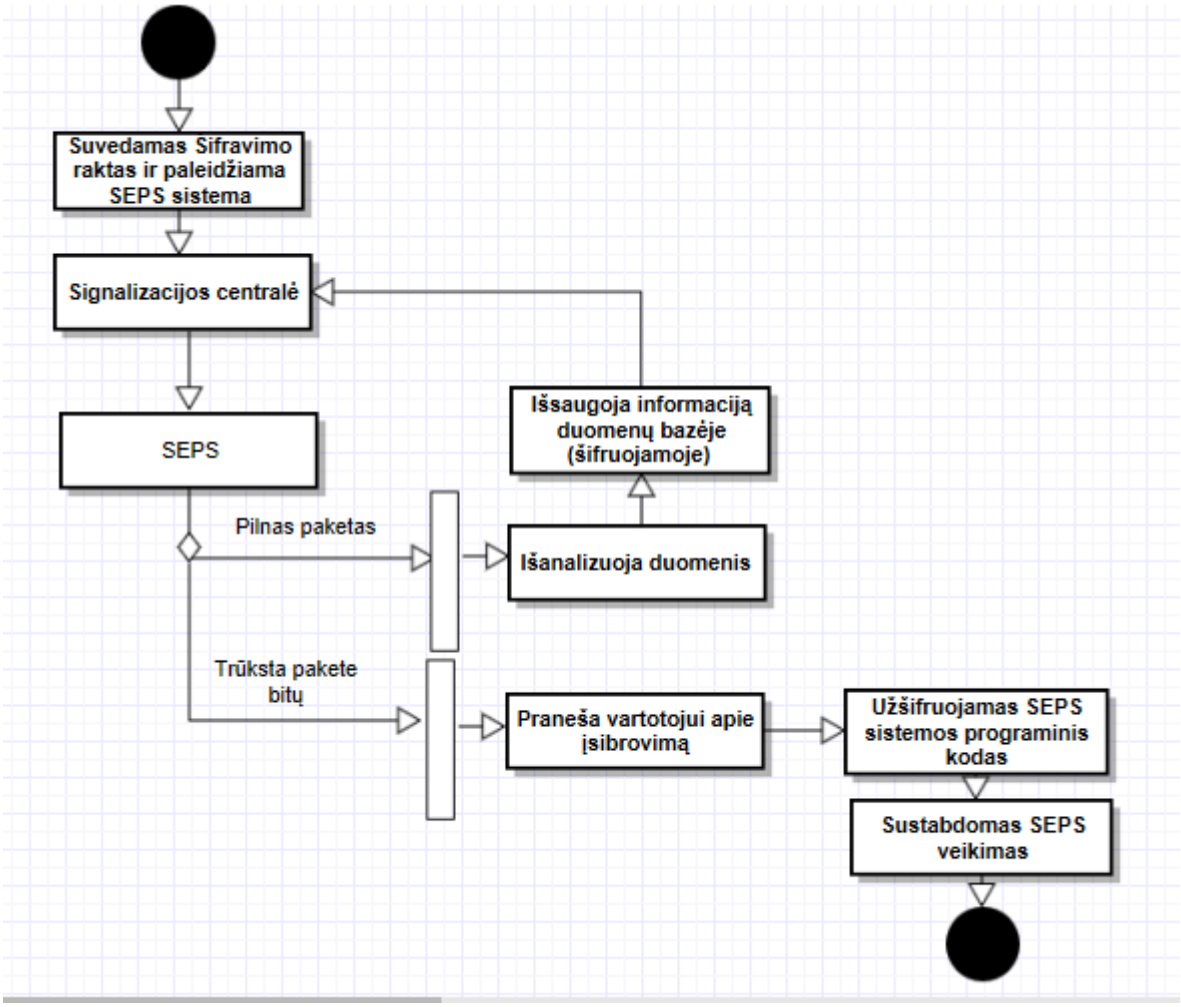
2.2 pav. SEPS sistemos įdiegimas su užšifruota kortele

2.2. Komunikavimo tarp SEPS ir signalizacijos centralės apsauga

SEPS sistemos pagrindinis darbas analizuoti signalizacijos duomenis. Duomenų perdavimas vyksta per centralę į *SEPS* sistemą, per *C-BUS* protokolą. Pagrindinė problema šitoje vietoje - lengvas įsibrovimas tarp šių dviejų prietaisų (*SEPS* ir centralės). Norint apsaugoti ir padaryti saugų bendravimą reikia imtis saugumo priemonių. Išanalizavę matome, kad geriausia imtis papildomo daviklio, bei kodo konfigūravimo.

SEPS sistema veikia realiu laiku, todėl neatsargus kodo pakeitimas gali kainuoti sistemos darbo kokybę. Jei sistema praleis vieną bitą (*SEPS* sistema ištisai gauna duomenis iš centralės) galime teigti. kad sistema praleido svarbų įvykį, o tai laikoma šios sistemos neveikimu. Dėl to, konfigūruojant kodą stengsimės naudoti kuo mažiau simbolių ir nenaudoti ilgų ciklų, kurie sulėtintų *SEPS* darbą. Taikysime paprastą tikrinimą (ar bitų masyvas atitinka jo tikrąjį ilgį - prisijungusi kenkėjiška sistema sulėtina duomenų perdavimą ar visai ji nutraukia trumpam laiko tarpui ir sistema būtų nuskaito mažiau ar išvis nenuskaito). Jeigu jis neatitinka tikrojo masyvo ilgio, sistema nutrauks darbą ir neleis įsijungti. Įjungimas bus galimas tik administratoriui.

Kaip matome iš veiksmų diagramos 2.3 pav, modifikavus programinį kodą atsirandą naujas tikrinimas (IF) ar bitų paketas yra pilnas. Jei šios sąlygos neatitinka, sistema nutraukia darbą, prieš tai informavusi vartotoją apie įsilaužimą. Kitu atveju, jeigu tikrinimo sąlyga atitiko sistema analizuoja duomenis ir saugoja duomenų bazėje, kuri yra šifruotoje aplinkoje. Po saugojimo vėl tikrinama gauto paketo bitų suma ir taip procesas nenutrūkstamai sukasi.



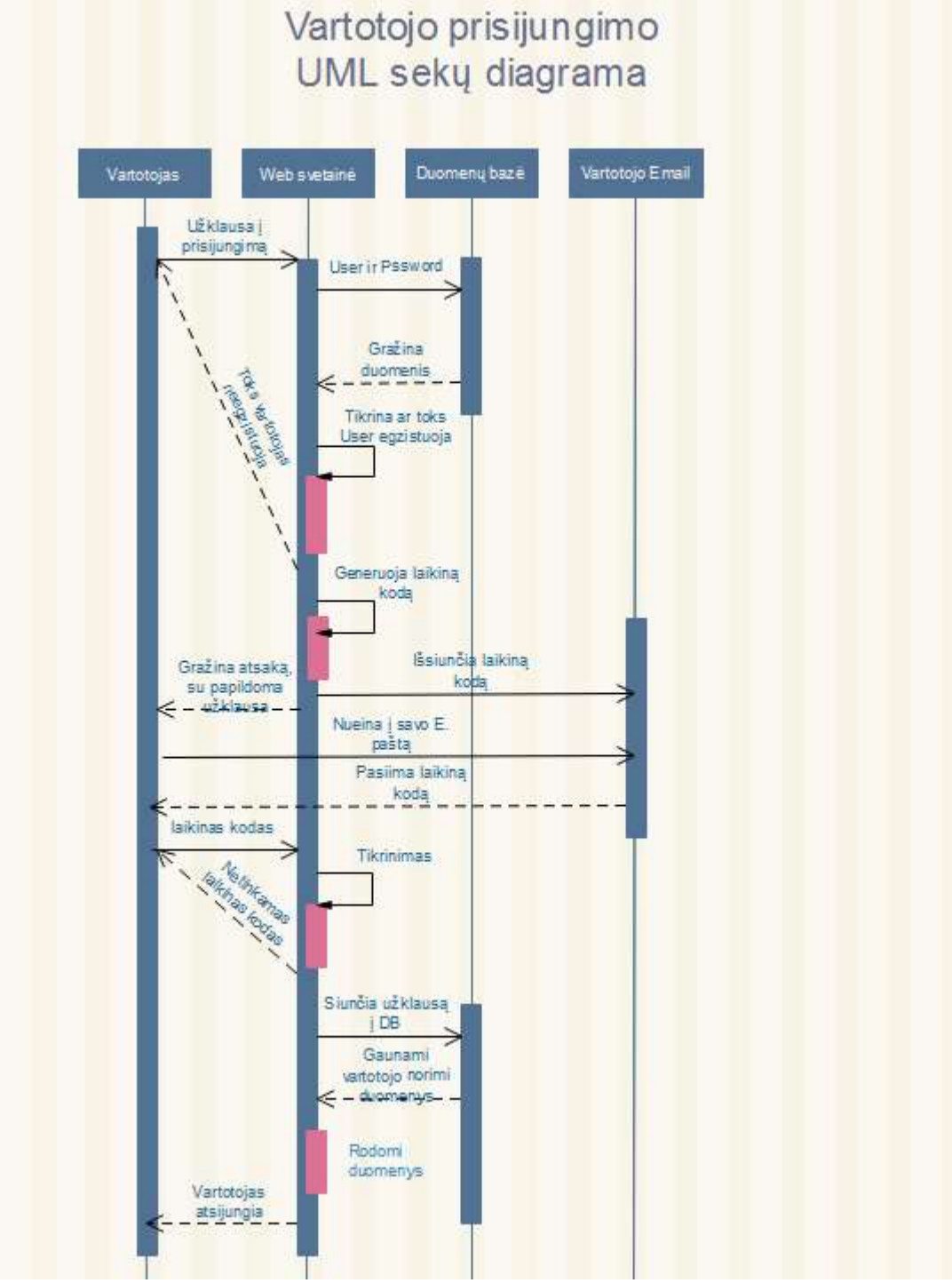
2.3 pav. Duomenų analizavimo veiksmų diagrama

2.3. Papildomo kodo generavimas

Didesnė dalis įsilaužimų vyksta per internetines svetaines, kai vadinami „hakeriai“ pasinaudoja prisijungimo duomenimis ir slaptažodžiu. Norint apsaugoti vartotojo duomenis rodomus internetinėje svetainėje nuspręsta naudoti papildomą kodų generatorių. Prisijungęs vartotojas į savo elektroninį paštą gauna (kuris buvo panaudotas registruojantis) automatiškai sugeneruotą 8 ženklų kodą ir yra nukreipiamas į kita puslapį, kuriame reikės jį įrašyti. Junginys iš raidžių ir skaičių yra laikomas tik tam

tikrą laiko tarpą ir tik vieną prisijungimą. Kai vartotojas atsijungs kodas bus ištrinamas iš atminties ir vartotojas antrą kartą su tuo pačiu kodu prisijungti negalės.

Vartotojo prisijungimo sekų diagramą matome 2.4 pav. (su papildomo kodo generavimu vartotojas turės atlikti papildomų veiksmų norint prisijungti prie svetainės). Vartotojas turės prisijungti su jau turimu prisijungimo vardu ir slaptažodžiu. Tada jam sugeneruotas papildomas kodas, kuris išsiųstas į elektroninį paštą. Vartotojas pasiėmęs laikiną kodą turės patalpinti jį į svetainę ir jeigu jis tinkamas, jam suteikta prisijungimo galimybė. Ten vartotojas galės matyti jam norimą (ir galimą) informaciją.



2.4 pav. Vartotojo prisijungimo UML sekų diagrama

2.4. Duomenų bazės lygmuo

Duomenų bazės lygmeniui apsaugoti reikia dvejų etapų:

1) Konfigūruojant internetinę svetainę
2) Konfigūruojant SQL serverį

2.4.1. Konfigūruojant SQL serverį

Norint apsaugoti SQL serverį nuo SQL injekcijų reikia atlikti keletą pakeitimų:

1) Sukurti vartotoją, kuris galės tik skaityti jam priklausančia informaciją, daryti SQL *Select* užklausas tik į tam tikras lentas.
2) Uždrausti klaidų gražinimą vartotojui. Tam puikiai tinka „*Catch*“ funkcija, kuri iškviečiama, kai įvyksta klaida.
3) Apsirašyti visus kintamuosius ir pritaikyti jiems tipą, bei kintamojo ilgį, kuris neleis vartotojui pateikti per ilgos reikšmes ar netinkamų ženklų.
4) Koduoti įvedamas reikšmes duomenų bazėje su funkcija *Base64*.
5) dinaminiuose užklausose naudoti „*sp_executesql*“ funkciją. Ši funkcija pirmiausia apdoroja paduotus laukus, o paskui tiktais įvykdo užklausą. Taip įsilaužėlis negali panaudoti netinkamų įvesties laukų, kurie galėtų pakenkti duomenų saugumui[25].

2.4.2. Konfigūruojant internetinę svetainę

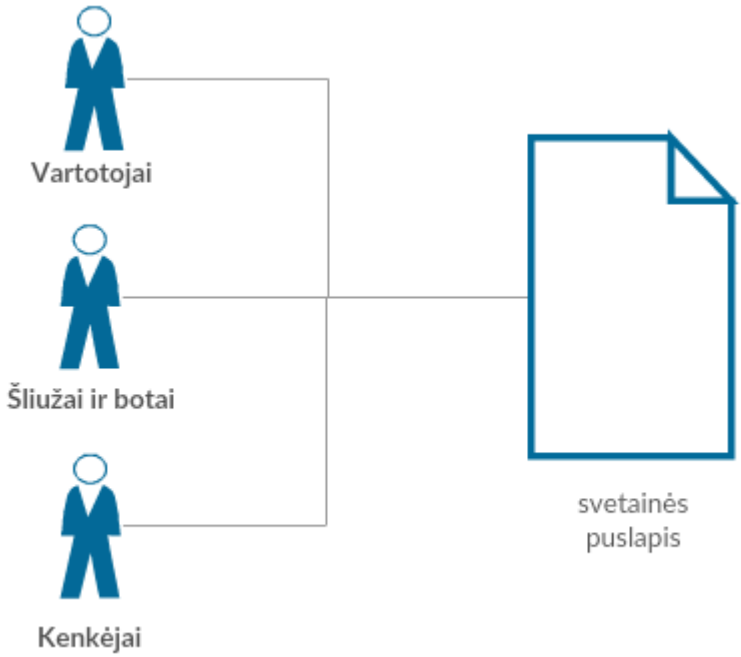
Reikia pakoreguoti vartotojui leidžiamus veiksmus internetinėje svetainėje, kad jis negalėtų pakenkti serverio darbui. Reikės atlikti šiuos pakeitimus:

1) Filtruoti vartotojo įvedamas reikšmes.
2) Įvedimo laukuose leisti vartotojui įvesti tik tam tikro ilgio ir formato ženklus.

2.5. Tinklo apsauga

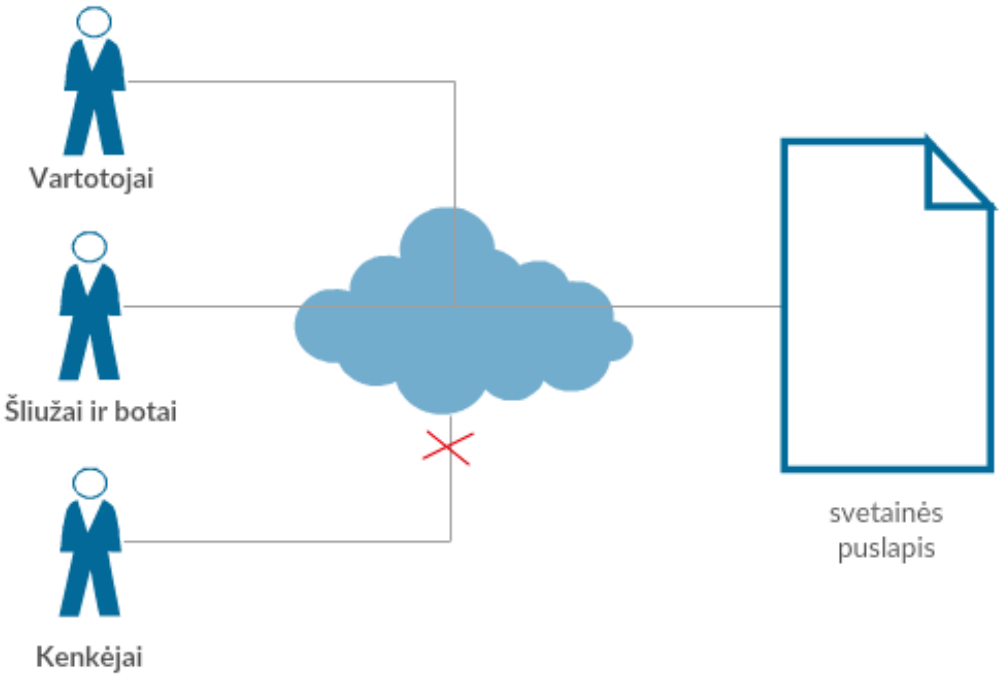
Tinklo apsaugai naudojama papildoma tarnyba CloudFlare. Ši tarnyba visą ateinantį srautą leidžia per savo debesį, kuriame, pagal aprašytas taisykles, vyksta einančio tinklo analizavimas ir priimami tam tikri sprendimai.

Serverio paleidimas per CloudFlare nėra sudėtingas, tereikia nurodyti keletą parametrų. Be *Cloudflare* visas tinklo srautas keliauja tiesiai į serverį, kuris bando atlikti visas užklausas kaip pavaizduota 2.5 pav.



2.5 pav. Tinklo srautas be *CloudFlare*

Taip sistema stengiasi visiems atsakyti kuo greičiau ir nežiūri, nei kokios užklausos jam paduotos, nei kas jas atsiuntė. Su *CloudFlare* pirmiausia paduotas užklausas apdoroja ši tarnyba, tik paskui jis gali patekti į svetainės puslapį. Ši tarnyba blokuoja visus jiems žinomus kenkėjus (pagal IP adresus ir puolimo tipus) ir kai kuriuos šliužus, ar botus, kurie gali pakenkti. Kaip pavaizduota 2.6 pav.



2.6 pav. Tinklo srautas su *CloudFlare*

2.6. SEPS programinės įrangos ir duomenų analizės projektavimo išvados

Kiekvienam *Signalizacijos įvykių apdorojimo sistemos* saugumo lygmeniui pavyko sukurti ir suprojektuoti sprendimo metodą. Išanalizuotas kiekvienas metodas ir jo poveikis sistemos darbui. Užšifravus kortelę prisidės dar du žingsniai prie analizavimo programos: tai šifravimas ir atšifravimas. Pridėjus papildomo kodo generavimą, vartotojas prisijungdamas turės atlikti papildomą veiksmų seką ir įvesti papildomą kodą į įvesties lauką svetainėje. Programa, kuri saugos operacinę sistemą veiks fone ir netrukdys sistemos darbui. Suprojektavus galima realizuoti ir ištestuoti metodus.

3. SIGNALIZACIJOS ĮVYKIŲ APDOROJIMO SISTEMOS PROGRAMINĖS ĮRANGOS IR DUOMENŲ APSAUGOS SPRENDIMO REALIZAVIMAS

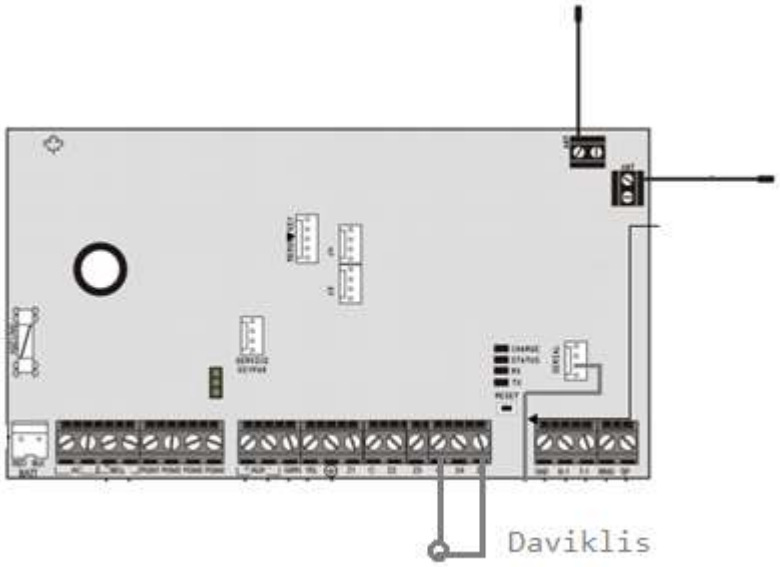
Atlikus analizę ir suradus sprendimo metodą prieinama prie realizacijos. Pagal pateiktus aprašymus toliau bus pateiktas *SEPS* programinės įrangos ir duomenų apsaugojimo projekto realizavimas.

3.1. Fizinio lygmens saugumo sprendimo realizacija

Fizinis lygmuo saugomas dvejais būdais: pridedant papildomą daviklį ir pakoreguojant programinį kodą.

3.1.1. Papildomo daviklio prijungimas

Kadangi *SEPS* sistema sujungta su signalizacijos sistema, tai papildomą daviklį galima prijungti tiesiai prie centralės, kaip parodyta 3.1 pav. Jutiklis prijungiamas į nepriklausomą zoną ir pačioje signalizacijos klaviatūroje užprogramuojamas nepriklausomoje zonoje. Pats jutiklis pritvirtinamas prie dėžės į kurią įdėta pati sistema. Norint prieiti prie *SEPS* reikės atidaryti dėžę, o atidarius dėžę signalizacija įsijungs ir pati *Signalizacijos apdorojimo sistema* nutrauks darbą.

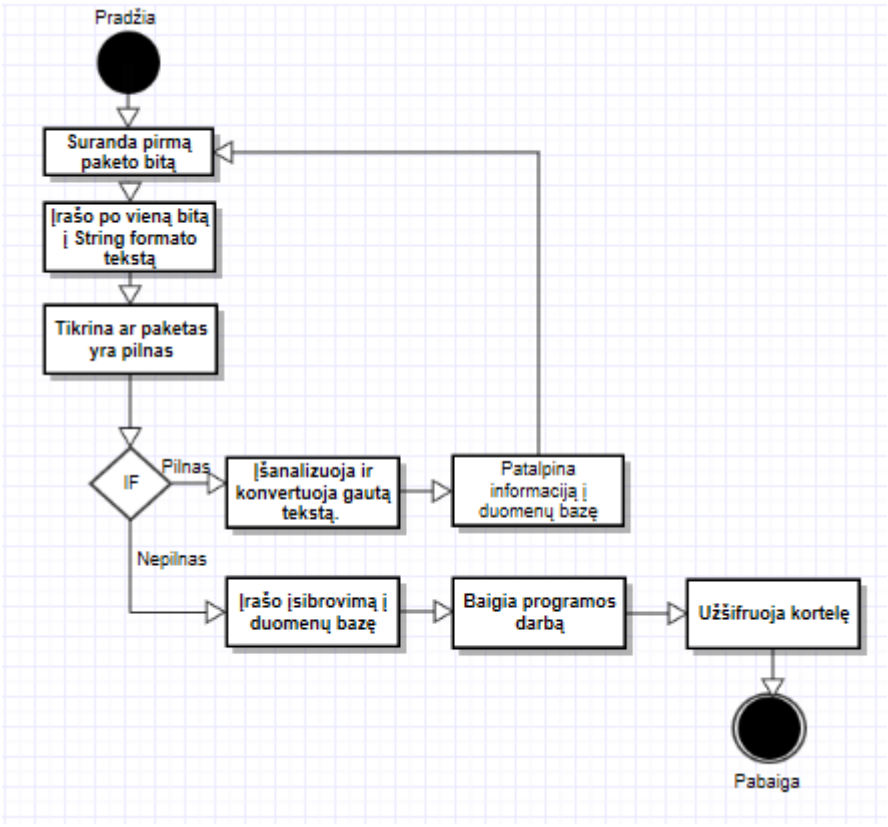


3.1 pav. Daviklio prijungimas prie signalizacijos centralės

3.1.2. Kodo konfigūracija

Prijungiant pašalinį prietaisą prie *SEPS* sistemos gali būti nuskaityti duomenys. Norint tai apsaugoti reikia konfigūruoti kodą. Kadangi sistema veikia realiu laiku kiekviena mili sekundė yra svarbi. Prijungiant kenkėjišką įrangą duomenų nuskaitymas gali užtrukti labai trumpa laiko tarpą (iki 1s).

Todėl reikia patalpinti paketo ilgio skaičiavimą, kuris aptiktų nepilną duomenų paketą ir nutraukti programos darbą.



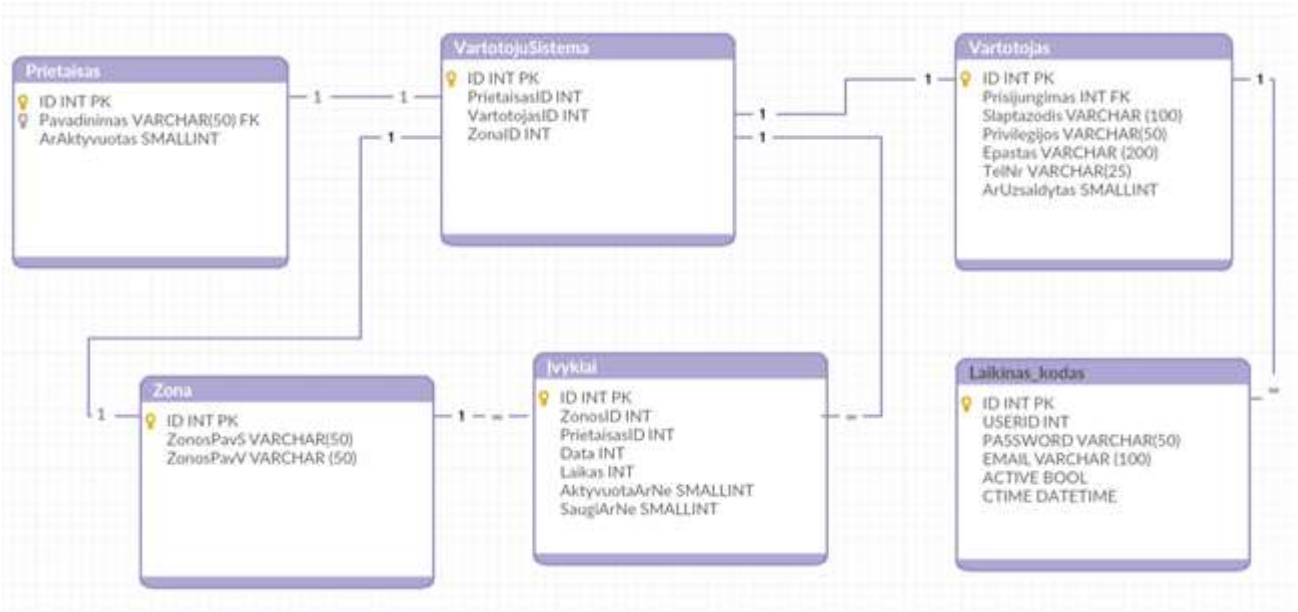
3.2 pav. Signalizacijos paketų analizavimo programa

Kaip matome iš 3.2 pav. sistema pradžioje susiranda pirmą paketo bitą, nuo kurio pradedama skenuoti ir įrašinėti bitus. Į šią vietą patalpinamas papildomas kodas, kuris, kai surinkamas visas paketas, patikrina ar paketas susideda iš 80 bitų ir ar paskutinis paketo bitas nėra sekančio paketo pirmas bitas. Taip sužinome ar programos veikimui nepakenkė pašalinai veiksniai. Jeigu aptinkame, kad į sistemą buvo bandyta įsilaužti, sistema įrašo paskutinį įvykį (kad buvo įsilaužta), pabaigia savo programos darbą ir užšifruoja atminties kortelę. Dalis programinio kodo atvaizduota pirmame priede.

3.2. Papildomo kodo generavimas

Norint apsaugoti vartotojo prisijungimą, generuojamas papildomas kodas, kuris galios tik tam tikrą laiko tarpą ir tik vieną kartą. Kodo generatorių reikia įkelti į internetinę svetainę. Kad, kai vartotojas bando prisijungti prie svetainės ir suveda teisingus prisijungimo duomenis jam sugeneruotų naują papildomą kodą ir išsiųstų jį elektroninio pašto adresu.

Visus sugeneruotus laikinus slaptažodžiu saugojame atskiroje duomenų bazės lentelėje (3.3 pav.), kuri bus sujungta su „Vartotojas“ lentele per UserID (vienas su daug ryšiu).



3.3 pav. Duomenų bazių schema

3.1 lent., pateikiami „Laikinas_kodas“ lentelės atributai

3.1 Lentelė Duomenų bazės lentelės „Laikinas_kodas“ atributų aprašymai

Atributas	Tipas	Aprašymas
ID	Int	Laikino kodo identifikacijos numeris (pirminis raktas).
USERID	Varchar(50)	Vartotojo ID, kuriam priklauso laikinas kodas.
PASSWORD	Varchar(50)	Laikinas slaptažodis.
EMAIL	Varchar(50)	Vartotojo elektroninis paštas.
ACTIVE	BOOL	Nurodo ar laikinas slaptažodis dar nebuvo panaudotas.
CTIME	Datetime	Sugeneruoto laikino slaptažodžio data.

Internetinėje svetainėje patalpinama funkcija, kuri generuos atsitiktinį kodą. Funkcija generuoja bet kokį raidžių ir skaičių junginį iš 8 ženklų (laikino kodo generavimo funkcija pateikta 2 priede).

Vartotojas norėdamas prisijungti prie svetainės turi suvesti savo vartotojo prisijungimo vardą ir slaptažodį. Tada sistema sugeneruoja laikiną kodą, kuri patalpina į duomenų bazės lentelę “Laikinas_kodas” ir išsiunčia elektroniniu pašto adresu. Elektroninį pašto adresą sistema pasiima iš lentelės “Vartotojas” pagal Vartotojo ID.

Kai vartotojas įveda laikiną slaptažodį, sistema tikrina ar atitinka įvestas slaptažodis pagal vartotojo vardą esantį “Laikinas_kodas” lentoje ir ar slaptažodis nebuvo panaudotas ir ar nėra pasenęs. Programinis kodas yra pateiktas antrame priede.

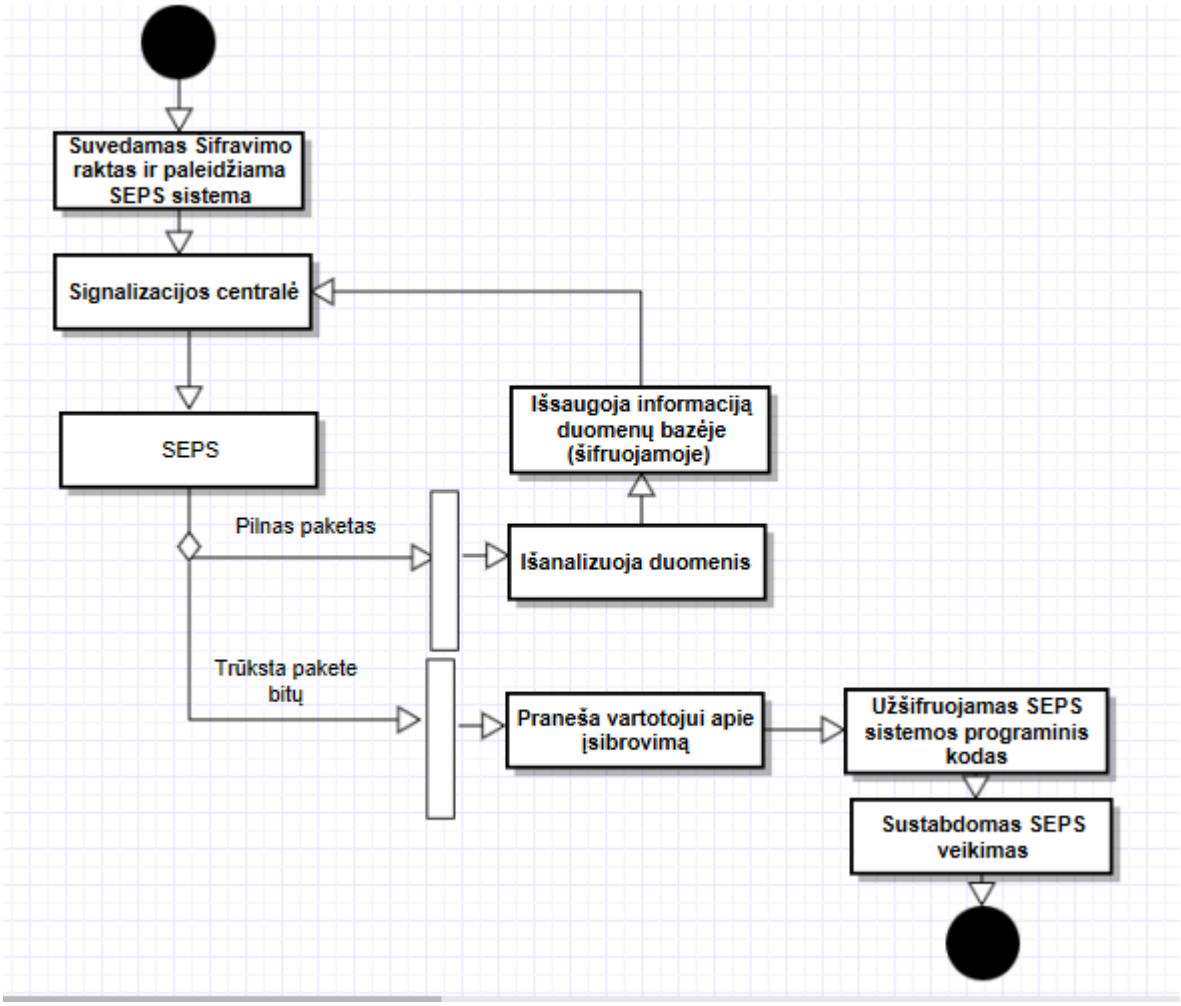
Žinoma papildomo kodo pagalba bus sunkiau įsilaužti į svetainę turint vien prisijungimo duomenis, bet visiškai apsisaugoti nuo įsilaužėlių - neįmanoma, todėl nereikia aklai tuo pasitikėti.

3.3. Šifravimas su *eCryptfs*

Saugumo spragą atminties kortelėje išspręsti pasirinkta šifravimo metodu, o užšifruoti naudosime *eCryptfs* šifravimo programinę įrangą.

Pirmiausia, į mikrokompiuterį reikia įsidiegti *eCryptfs* įrankius. Su šiais įrankiais pasirenkama kokią direktoriją užšifruoti. Pasirenkamas vienas vartotojas, su kuriuo vykdomas visas programinis kodas. Kadangi visas kodas bus patalpintas po vieno vartotojo namų (angl. Home) direktorija, užtenka užšifruoti tik viena direktoriją. Užšifravus vieną direktoriją vartotojas, norėdamas dirbti joje, turi įvesti papildomą slaptažodį. Kiti vartotojai, jeigu neturi prieigos prie to failo, negali matyti net tos direktorijos, bet vartotojas (adminstratorius), turintis visas teises, gali matyti direktoriją, bet ne failus kurie yra viduje. *ECryptfs* užšifruoja direktorija taip, kad kiti vartotojai mato atskirus du failus, kurie perspėja nelysti į svetimus failus. Tokiu būdu niekas negali perskaityti duomenų neatšifravus kortelės.

Kadangi sistemą vartotojui įdiegs atsakingas už paslaugą žmogus (žinoma tai gali būti ir pats vartotojas, bet tada jis prisiima visas atsakomybes ir teises), tai vartotojas, bei visi kiti asmenys turi būti apsaugoti nuo patekimo į sistemos programinį kodą. Dėl to specialistui suteikiamas lapas (ne elektroninis variantas), su *SEPS* sistemos šifravimo slaptažodžiu. Įdiegęs sistemą vartotojui specialistas atšifruoja sistemą ir paleidžia ją darbui. Sistemoje įdiegtas programinis kodas, kuris bet kokiems sistemos trukdžiams nutrauktų sistemos darbą, bet prieš tai ją vėl užšifruoja pasinaudojus raktu (kaip pavaizduota 3.4 pav.).

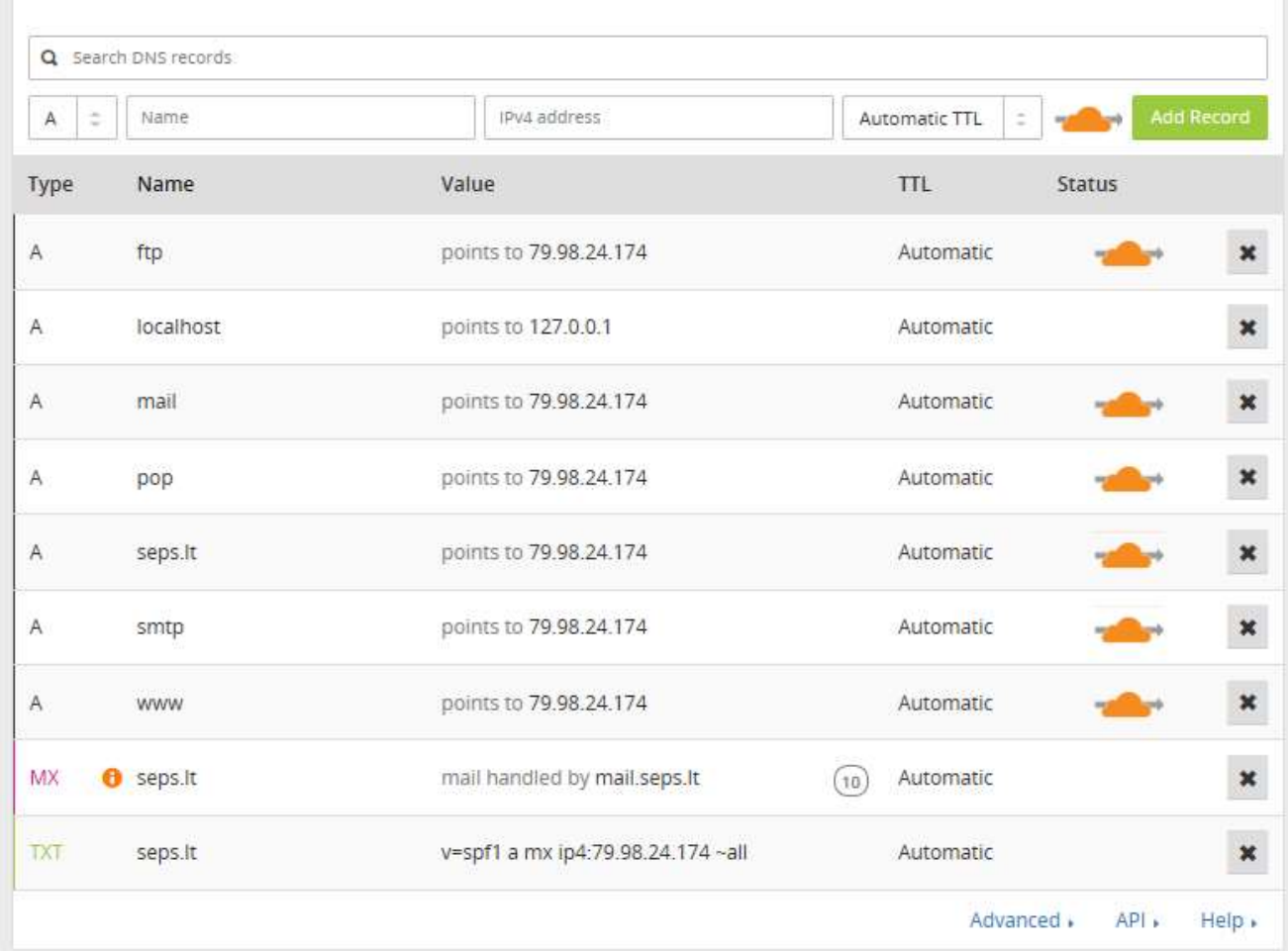


3.4 pav. Sistemos veiksmų diagrama užšifravus sistemą.

3.4. Apsauga nuo DDOS

Apsisaugojant nuo internetinių atakų naudojamas *Cloudflare*. Ši tarnyba tikrina visą įeinantį srautą į pasirinktą svetainę. Šiuo atveju svetainė *seps.lt.* Ji užregistruojama į *Cloudflare.* Registracijos metu reikia nurodyti planą, kokio saugumo lygį naudosime. Siūlomi net keturi lygiai, bet kadangi svetainėje nebus didelio kiekio vartotojų, bus naudojamas nemokamu paslaugų paketas iš *Cloudflare.* Jis ne tik apsaugo mūsų svetainę, bet ir pagreitina jos darbą, filtruodamas visą įeinantį srautą, jį apdorodamas ir paskui perleisdamas į mūsų serverį. Tai sutaupo mūsų serverio resursus, kurių turi užtekti pagrindiniam sistemos darbui.

Cloudflare ne tik apsaugo svetainę, bet ir su ja susijusius dalykus tokius kaip: *ftp* ir paštą. Plačiau pavaizduota 3.5 pav.



3.5 pav. *Cloudflare* saugomi objektai

Ši tarnyba turi ne tik saugojimo funkciją, bet ir leidžia analizuoti visą srautą, kuris padeda projektuoti ir priimti sprendimus.

3.5. Duomenų bazių apsauga

Duomenų bazių lygmuo saugomas tam, kad sistemos nebūtų įmanoma pažeisti SQL injekcijos atakomis:

1) Sukuriamas vartotojas, kuris turi teises tik skaityti duomenis iš vienos lentelės esančios duomenų bazėje. Tuo siekiama, kad vartotojas negalėtų pamatyti slaptų duomenų. Tam reikės:
    - Sukurti naują vartotoją (CREATE USER 'user1'@'localhost' IDENTIFIED BY '123456');
    - Priskirti jam *Read* teises vienai lentelei (GRANT SELECT ON SEPS.Įvykiai TO 'user1'@'localhost');

2) Pakeičiamos visos dinamiškos struktūros, kuriose vartotojas kreipiasi, kad vartotojo įvedamos reikšmes būtų apdorotos iš anksto. Taip pat pridedamos klaidų gaudymo metodas, kuris iškilus klaidai negražintų jos vartotojui.

   Pavyzdžiui, viena iš tokių procedūrų - laikino kodo generavimas. Procedūra parametrizuojame su *sp_executesql* funkcija. Visas kodas patalpinamas į "*Begin Try*" ir "*End Try* " rėmus. Tada įvykus kokiai nors klaidai ją pagauname su "*Catch*" funkcija ir gražiname visas reikšmes į ankstesnę būseną su "*ROLLBACK TRANSACTION"*. Kaip atrodo programinis kodas pasinaudojus šia funkcija pateiktas trečiame priede.
3) Vartotojo įvedamas slaptažodis koduojamas su „TO_BASE64" funkcija. Taip apsisaugoma jei kitas vartotojas kokiu nors būdų gautų slaptažodį iš duomenų bazės.

3.6. *Signalizacijos įvykių apdorojimo sistemos* programinės įrangos ir duomenų apsaugos sprendimo realizavimo išvados

Atliktos sistemos saugumo spragų sprendimų realizavimas. Realizuoti visų lygmenų saugumo sprendimai. Pateikta nauja duomenų bazės schema, sekų diagrama užšifravus programinį kodą. Duomenų bazės saugumui pasitelkta funkcija, kuri parametrizuoja užklausas. Visas tinklo srautas leidžiamas per *CloudFlare* tarnybą. Atlikus realizaciją galima pradėti testavimą.

4. SIGNALIZACIJOS ĮVYKIŲ APDOROJIMO SISTEMOS PROGRAMINĖS ĮRANGOS IR DUOMENŲ APSAUGOS SPRENDIMO TESTAVIMAS

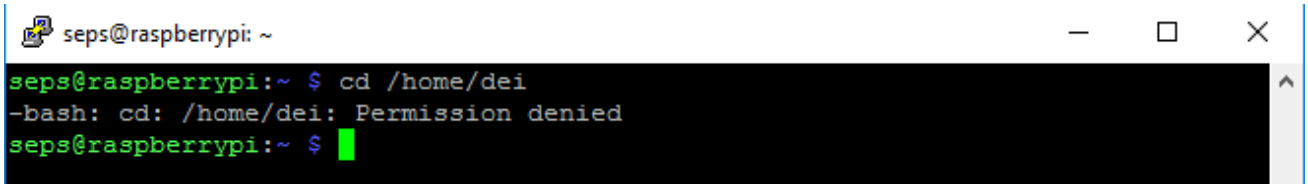
Šiame skyriuje aprašomi *Signalizacijos įvykių apdorojimo sistemos* papildomų saugumo metodų testavimai, kurie realizuoti praeituose skyriuose. Testavimas atliktas ant panašios sistemos kaip SEPS. Šioje sistemoje įdiegta tokia pati operacinė sistema – Raspbian. Įdiegtas programinis kodas ir sukurta internetinė svetainė su duomenų baze.

4.1. Atminties kortelės šifravimas

Užšifravus kortelę reikia ištestuoti ar kiti vartotojai negali matyti duomenų. Testavimo metu bus naudojamas „*Juodos dėžės*“ (angl *Black Box*) metodas. Šis metodas parodo ką paduodame sistemai ir ką turime gauti. Šifravimas bus bandomas keliais metodais. Pirmu metodu bus bandoma atidaryti failus su vartotoju, kuris neturi teisių visai matyti tą direktoriją. Antru metodu prisijungiama su administratoriumi ir bandoma nuskaityti failus, kurie yra užšifruoti vartotojo namų direktorijoje. Trečiu metodu atsidaromi failai su vartotoju, kuris turi teises matyti tą failą ir yra to šifravimo savininkas.

4.1.1. Pirmas metodas (svetimas vartotojas)

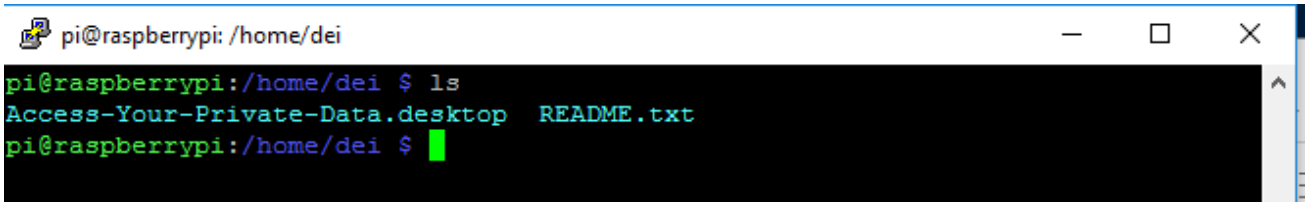
Pirmas metodas atliekamas susikuriant naują vartotoją „*seps*“, kuris turi teises matyti tik savo namų direktorijoje esančius failus. Su šiuo vartotoju bandant nueiti į kitą direktoriją, kurioje yra užšifruoti duomenys pranešama, kad šis vartotojas neturi teisių atidaryti šio aplanko. Kaip pavaizduota 4.1 pav.



4.1 pav. Jungiantis su vartotoju neturinčiu teisių

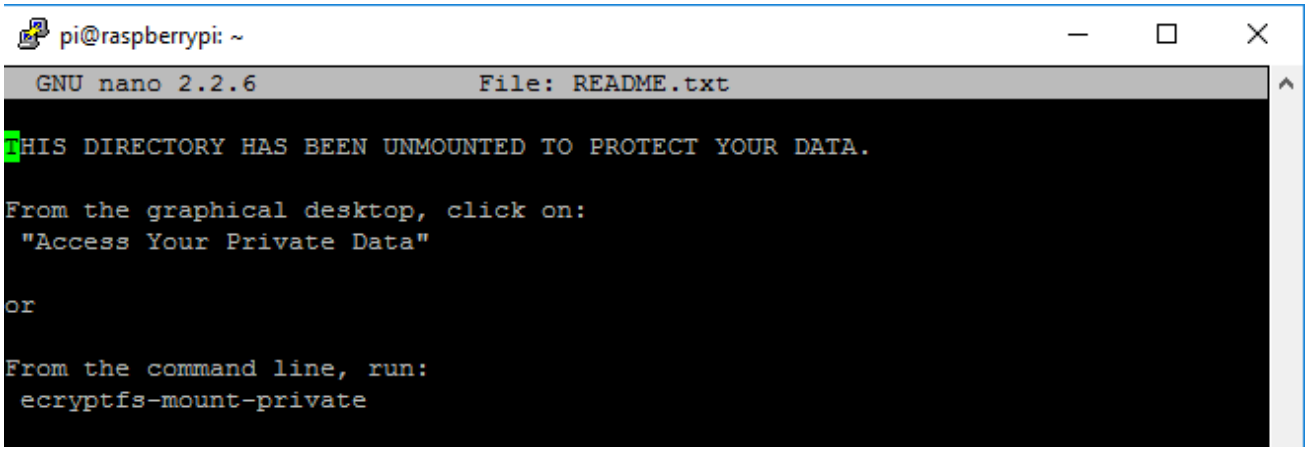
4.1.2. Antras metodas (administratorius)

Antru metodu prisijungiama su administratoriumi „*pi*“ ir nueinama į „*dei*“ vartotojo namų direktoriją. Administratorius turi teises nueiti į šią direktoriją, todėl to nuėjus ten matomi keli failai. Kaip atrodo *eCryptfs* sugeneruoti failai matoma 4.2 pav.



4.2 pav. Neatšifruota „dei“ namų direktorija

Pirmas failas skirtas grafinę sąsaja naudojantiems vartotojams. Paspaudus ant jo sistema leistų vartotojui įvesti slaptažodį, kad atšifruotų direktoriją. Atsidarius teksto failą matomA, kad tai yra *eCryptfs* sugeneruotas failas, kuriame rašoma kad tai yra užšifruota direktorija ir norint ją atšifruoti reikia įvesti papildomą komandinę eilutę. Tam taip pat reikės slaptažodžio. Kaip atrodo tekstinis failas sugeneruotas *eCryptfs* matomas 4.3 pav.



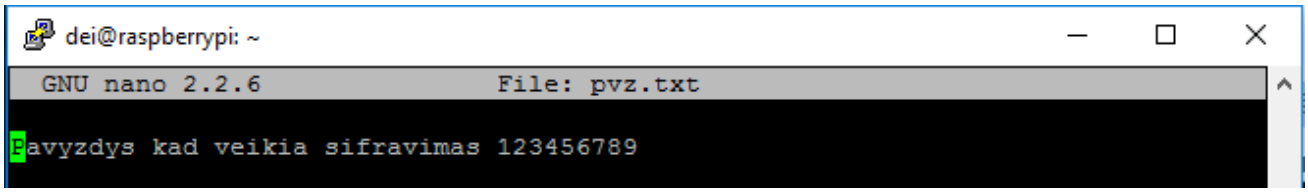
4.3 pav. Failas, „README.txt“ atidarytas su administratoriaus teisėmis.

4.1.3. Trečias metodas (atšifravus)

Trečiu metodu naudojamas sekantis vartotojas, kuris yra užšifruotos namų direktorijos šeimininkas. Prisijungus su šiuo vartotoju būtina iškart atšifruoti direktorija, nes kitaip jisai negali nieko daryti (nesuteiktos teisės). Atšifravus namų direktoriją šeimininkas mato visus failus ir aplankus. Kaip atrodo visi failai esantys namų direktorijoje galima pamatyti 4.4 pav.

4.4 pav. Atšifravus „dei“ vartotojo namų direktoriją matomi failai

Patikrinama ar matomi tik failai ar ir duomenys esantys viduje, pabandoma atsidaryti, kurį nors failą (patikrinimui ar veikia šifravimas sukurtas pvz.txt failas). Atsidarius failą matoma, kad jis yra atšifruotas ir galime nuskaityti visus duomenis. Atšifruoto failo duomenys atvaizduoti 4.5 pav.



4.5 pav. Atšifruoto failo pavyzdys

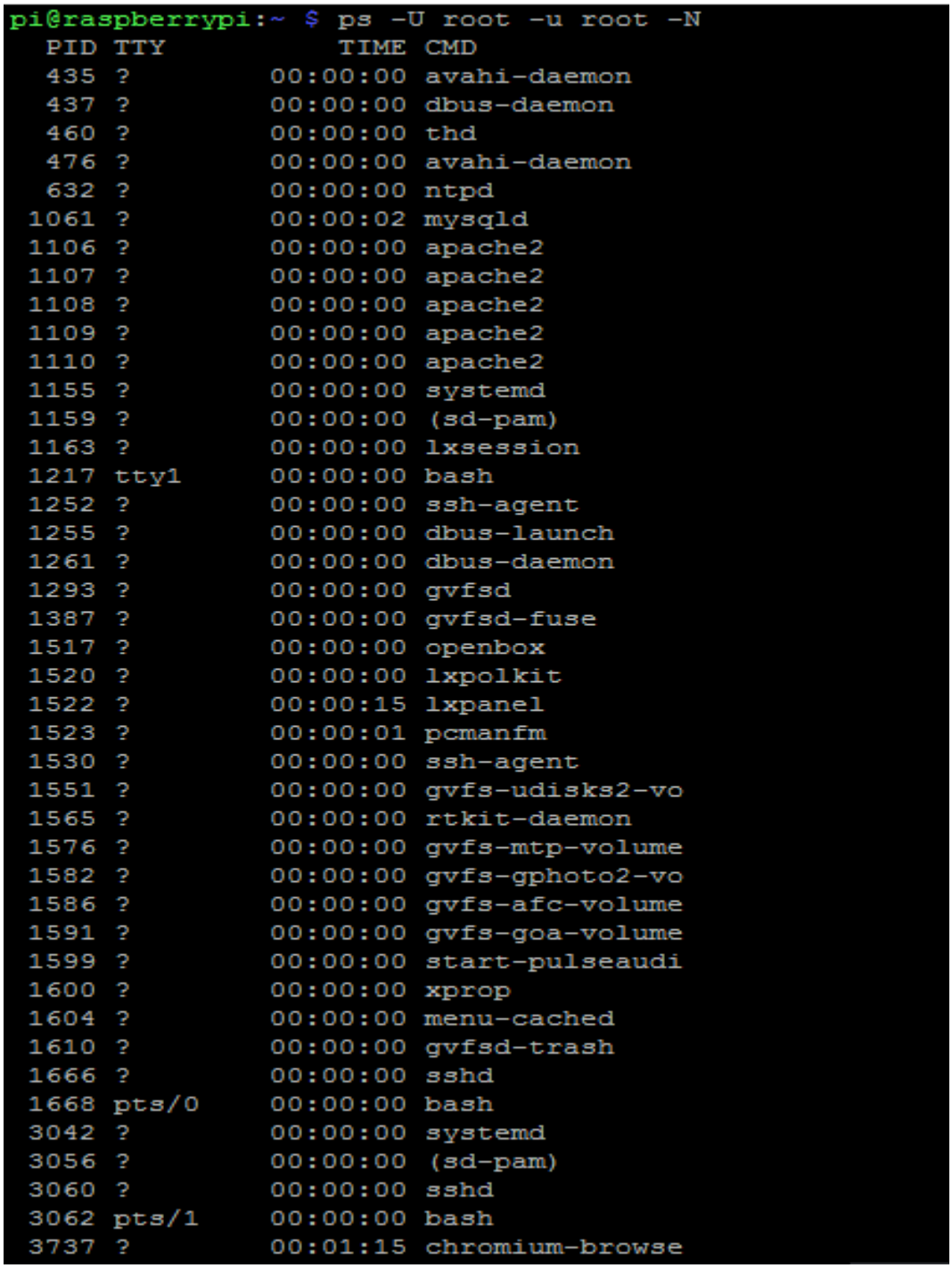
4.1.4. Atmintie kortelės šifravimo testavimo išvados

Atlikus tris bandymus matoma, kad visais testavimo bandymais šifravimo metodas pasiteisino. Bandant atidaryti su vartotoju, kuris neturi teisių, jam neleidžiama net prieiti prie direktorijos. Bandant atidaryti su administratoriumi rodoma, jog direktorija yra užšifruota ir neleidžiama net pažiūrėti koki failai yra direktorijoje. Trečiu bandymu jungtasi su vartotoju, kuris turi visas teises į direktoriją, suvedus slaptažodį rodo visus failus bei galima juos valdyti.

4.2. Operacinės sistemos lygmens saugojimo testavimas

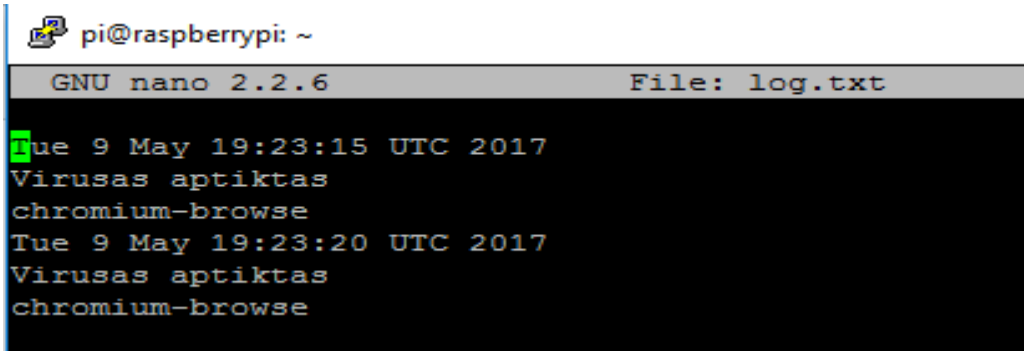
Operacinės sistemos lygmuo saugojama nuo pašalinių procesų veikimo. Įdiegta papildoma programa, kuri sukasi vartotojo „*dei*“ fone. Ši programa tikrina visus sukančius procesorius ir jei randa, kad tas procesas nepatenka į sąrašą jį sunaikina, užrašo įvykį į žurnalo failą ir išjungia sistemą.

Pirmiausia bus testuojama ar programinis kodas aptinka pašalinius procesus. Pašalinę programą paleidžiama su bet kuriuo vartotoju. Testavimo metu paleidžiama *chrome* naršyklė. Visas paleistų programų sąrašas (kartu ir *chrome* naršyklės programa) atvaizduotas 4.6 pav.



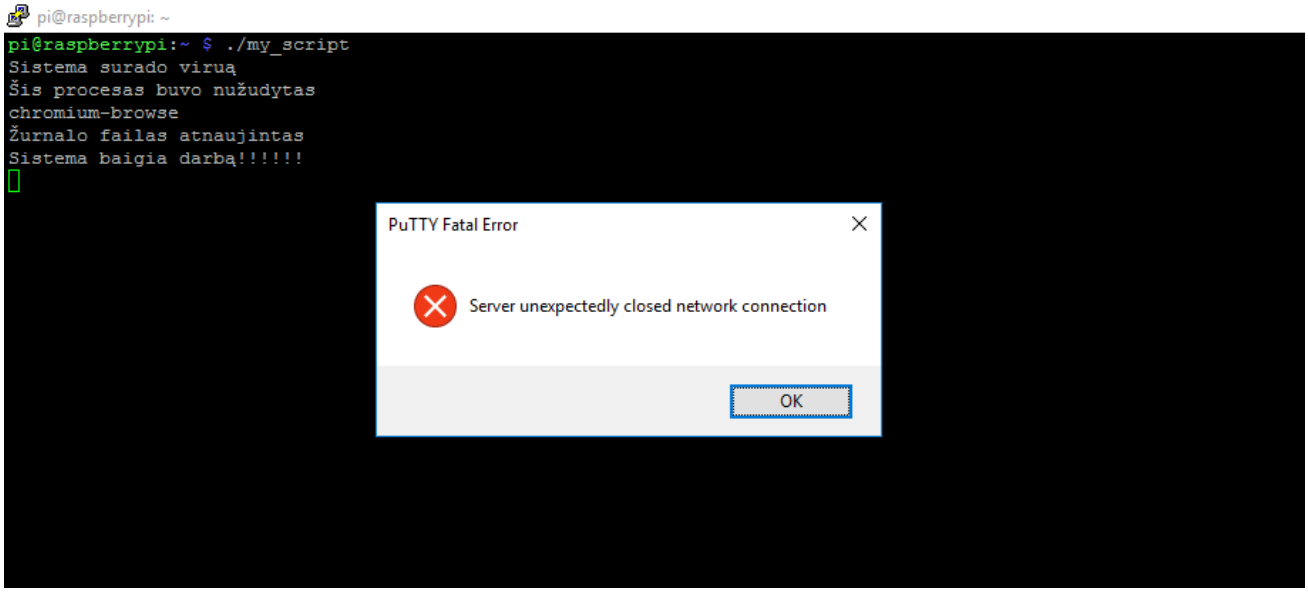
4.6 pav. Testavimo metu veikiančių programų sąrašas

Sistema aptikus netinkamą procesą jį sunaikina ir įrašo į žurnalų failą. Kaip atrodo žurnalo failas pateikta 4.7 pav. Iš žurnalo failo matoma kada ir koks procesas sutrukdė sistemos darbą.



4.7 pav. Žurnalų failas

Po įrašymo į žurnalų failą programa save išjungia. Taip sistema apsisaugoja nuo pašalinės programinės įrangos, kuri gali pakenkti sistemos darbui arba surinkti konfidencialius duomenis. Paleidus šį programinį kodą ne fone, matoma kokia seka vykdomi žingsniai. Šie žingsniai pateikti 4.8 pav.

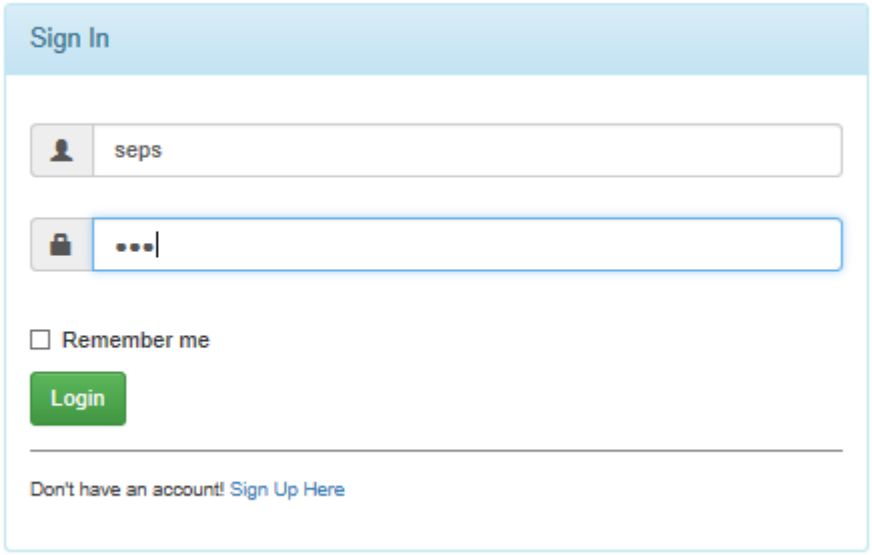


4.8 pav. Programinis kodas aptiko pašalinį procesą

4.3. Papildomo kodo testavimas

Vartotojo lygmenyje saugojamas vartotojo prisijungimas papildomu kodu. Vartotojas norėdamas prisijungti prie svetainės turi suvesti savo prisijungimo duomenis. Kai vartotojas prisijungia prie svetainės jam iššoka papildomas langas, kuriame reikia suvesti papildomą kodą, kuris buvo sugeneruotas ir jam išsiųstas. Bet prieš iššokant vartotojo langui, su papildomu kodo vedimu, serveris sugeneruoja papildomą kodą, jį išsiunčia vartotojui elektroniniu paštu, kuris nurodytas duomenų bazėje. Taip vartotojas suvedęs papildomą kodą, kurį rado elektroniniam pašte, gali prisijungti prie sistemos.

Visa ši seka bus įvykdyta ir ištestuota ar visi žingsniai yra atliekami iš sistemos. Pirmiausia prisijungiame prie svetainės su savo prisijungimo duomenimis, kaip pavaizduota 4.9 pav.



4.9 pav. Vartotojo prisijungimo langas

Tada gaunamas papildomas kodas. Suvedame papildomą kodą į laukelį svetainėje ir prisijungiame prie svetainės. Jei kodas suvestas teisingas prisijungiama prie sistemos, jei klaidingas išmeta klaidą.

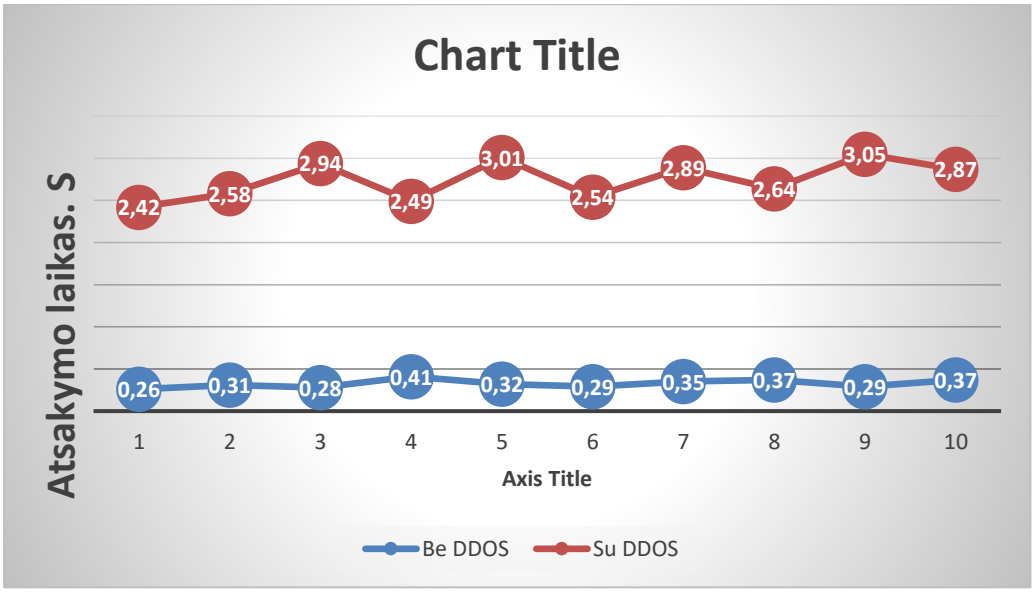
4.4. Tinklo saugos su *CloudFlare* testavimas

Tinklo apsaugai naudojama *CloudFlare* tarnyba. Ją ištestavus patikriname ar ji atitinka mūsų keliamus tikslus. Testavimui naudojama *LOIC* programinė įranga. Su šia įranga sugeneruojama DDOS ataka į mūsų internetinę svetainę ir taip patikrinama kaip veikia *CloudFlare* tarnyba.

Pirmiausia atliekami bandymai be *CloudFlare* tarnybos. Paleidžiama minimalia užklausa ir tikrinama per kiek laiko atsiunčiamas atsakymas. Atliekami 10 bandymų be DDOS atakų ir su jomis. Bandymai užrašyti 4.1 lent. ir atvaizduoti 4.14 pav.

4.1 Lentelė Bandymai be *CloudFlare* tarnybos

Be CloudFlare	1	2	3	4	5	6	7	8	9	10	Vid
Be DDOS	0,26	0,31	0,28	0,41	0,32	0,29	0,35	0,37	0,29	0,37	0,325
Su DDOS	2,42	2,58	2,94	2,49	3,01	2,54	2,89	2,64	3,05	2,87	2,743

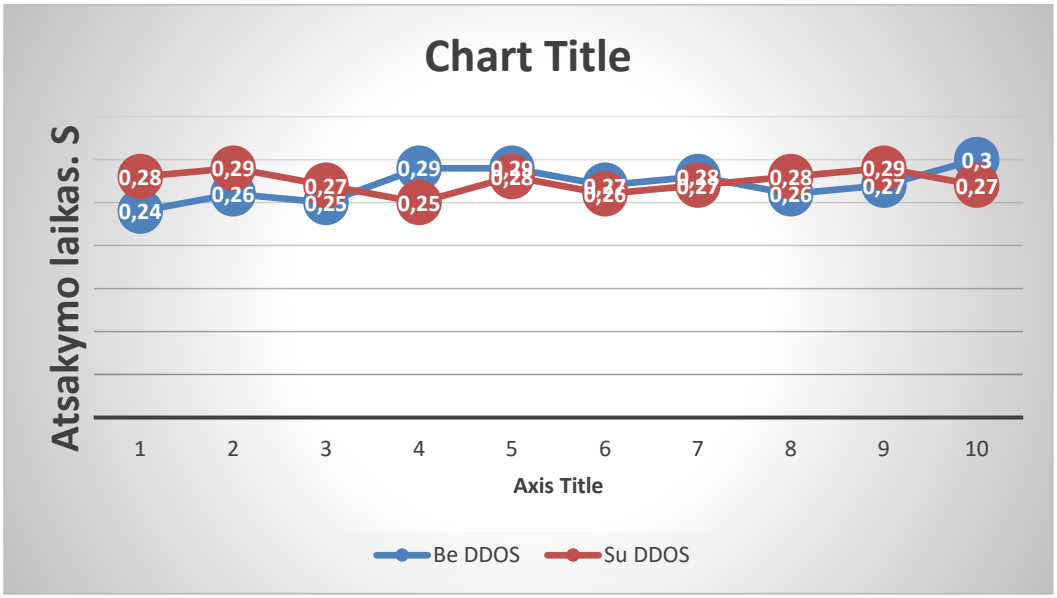


4.10 pav. Bandymų rezultatai be *CloudFlare* tarnybos

Toliau atliekami bandymai su *CloudFlare* tarnyba. Daromi 10 bandymų su tokia pačia užklausa. Bandymai užrašyti 4.2 lent ir atvaizduoti 4.15 pav.

4.2 lentelė Bandymai su *CloudFlare* tarnyba

Su CloudFlare	1	2	3	4	5	6	7	8	9	10	Vid
Be DDOS	0,24	0,26	0,25	0,29	0,29	0,27	0,28	0,26	0,27	0,3	0,271
Su DDOS	0,28	0,29	0,27	0,25	0,28	0,26	0,27	0,28	0,29	0,27	0,274



4.11 pav. Bandymų rezultatai su *CloudFlare* tarnyba

Atlikus šiuos bandymus ir įvertinama, kad paklaida gali būti iki 0.1s. Dėl matavimo paklaidos ir pašalinių procesų veikimo matoma, kad DDOS atakos, kai nėra *CloudFlare* tarnybos stipriai sulėtina

užklausų gražinimo laiką, bet įdiegus *CloudFlare* DDOS atakos visiškai neturėjo įtakos užklausų gražinimo laikui. *CloudFlare* patenkina lūkesčius ir apsaugo internetinę svetainę

4.5. Signalizacijos įvykių apdorojimo sistemos programinės įrangos ir duomenų apsaugos sprendimo testavimo išvados

Testavimas atliktas su panašia sistema, kurioje buvo ištestuoti, keli metodai: šifravimas, procesų tikrinimas operacinėje sistemoje, prisijungimas su papildomo kodu ir apsauga nuo DOS atakų.

Ištestavus dokumento šifravimą matoma, kad žmogus neturintis teisių ir neiššifravus dokumento negali matyti, kas yra tame dokumente.

Procesų tikrinimo programa aptiko pašalinį procesą, jį sunaikino, įrašė įrašą į žurnalo failą ir išjungė sistemą.

Vartotojui bandant prisijungti prie sistemos, papildomo kodo generatorius sugeneruoja kodą. Vartotojas gali prisijungti tik įvedus papildomą kodą.

Naudojant *CloudFlare* tarnybą svetainė buvo apsaugota nuo DDOS atakų.

Ištestavus visus saugumo metodus matoma, kad jie veikia puikiai ir įvykdo reikalavimus.

5. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1) Išanalizuota *Signalizacijos įvykių apdorojimo sistema,* aptiktos saugumo spragos ir suskirstytos į saugumo lygmenis. Nustatyti ir išanalizuoti galimi saugumo metodai kiekvienam lygmeniui. Išanalizavus saugumo lygmenis matoma, kad kiekvienam lygmeniui bus skirti papildomi saugumo metodai. Fizinis lygmuo bus saugomas, koreguojant programinį kodą ir pridedant papildoma daviklį. Operacinės lygmenį saugosime, sukurdami programą, kuri apsaugo nuo kenkėjiškų programų. Duomenų bazių lygmuo bus saugomas, koreguojant užklausų funkcijas ir apribojant vartotojo teises. Tinklas bus saugosimas pasitelkiant *CloudFlare* tiekėjus. Jie apsaugos tinklą nuo daugelio rūšių atakų. Vartotojo lygmuo bus saugomas su papildomu kodu. Vartotojas jungdamasis prie sistemos turės suvesti papildomą kodą.
2) Suprojektuojami saugumo spragų sprendimo metodai. Išanalizuota kaip kiekvienas metodas paveiks sistemos darbą. Pateikiamos sekų diagramos ir veiksmų diagramos. Užšifravus kortelę prisidės dar du žingsniai prie analizavimo programos. Pridėjus papildomo kodo generavimą, vartotojas prisijungdamas turės atlikti papildomą veiksmų seką. Programa, kuri saugos operacinę sistemą, veiks fone ir netrukdys sistemos darbui.
3) Realizuoti visų lygmenų saugumo sprendimai. Pateikta nauja duomenų bazės schema, sekų diagrama užšifravus programinį kodą. Duomenų bazės saugumui pasitelkta funkcija, kuri parametrizuoja užklausas. Visą tinklo srautą leidžiame per *CloudFlare* tarnybą.
4) Testavimas atliktas ant panašios sistemos. Atlikus testavimą paaiškėjo, jog testuojant svarbu susikurti gerą testavimo planą - t.y. metodika, aplinka ir kaip testuosime sistemą. Šis planas įtakoja kiek galimų variantų bus patikrinta, prie kurių sistema gali veikti nekorektiškai. Taigi, kuo tikslesnis bus sudarytas planas tuo galutiniame produkte bus mažiau klaidų.

6. PRIEDAI

6.1. Signalizacijos paketų analizavimo programos kodas

```
while (a=1)
{
  if  (digitalRead(CLK) == LOW)
  {
    //=====================================
    long time1 = millis();
    while (digitalRead(CLK) != HIGH);
    long time2 = millis();
    long timeDiff = time2 - time1;
    //=====================================
    if (timeDiff > clockTrukmesRibaMs)
    {
      bool done = false;
      int i = 0;
      while (!done)
      {
        if (digitalRead(CLK) == LOW)

        {
          //=====================================
          if (digitalRead(DTA) == HIGH)
          {
            msg += "0";
          }
          else
          {
            msg += "1";
          }
          //=====================================
```

```
    long time3 = millis();
    while (digitalRead(CLK) == LOW)
    {
     long time4 = millis();
     long timeDiff2 = time4 - time3;
     if (timeDiff2 > clockTrukmesRibaMs)
     {
      done = true;
      break;
     }
    }
  }
}
      string amsg = msg;
      msg = "";
      string zona = "";
      int length;
      length=amsg.size();
      If (length > 79 or (length = 80 and left(amsg)=Pirmasbit))
                  /*Tolimesnis programos kodas*/
      Else
                  FILE * fp;
                  Now = time(0);
                  Char ch;
                  If(NULL == (Fp =fopen(„securitylog.txt“,“w“)))
                        {return 0}
                  Printf(„\n“ Now „ - Įvyko įsilaužimas“);
                  While((ch=getchar())!=EOF)
                        Putc(ch,fp);
                  Fclose(fp);
                  Return 0;
```

6.2. Prisijungimas prie internetinės svetainės su laikinu kodu

```
//Laikino kodo generavimas
```

```
Function generateRandomString($lenght = 8)
{$characters = '
        0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
$chalength = strlen($characters);
$randomCode = ‘’;
For ($i = 0; $i < $length; $i++) {
        $randomCode .= $characters[rand(0,$chalength – 1)];
        }
        Return $randomCode
}

//Laikino kodo patalpinimas į duomenų bazę

$result =mysql_query(“SELECT 1 FROM Laikinas_kodas where USERID = $userid LIMIT)
If (mysql_fetch_row($result)){
        $sql = mysql_query „UPDATE Laikinas_kodas SET PASSWORD=$randomCode
,ACTIVE=“1“, CTIME =date() WHERE USERID = $userid“
}
Else{
        $myfile = fopen(„sqllog.txt“,“w“);
        $sql =mysql_query “INESRT INTO Laikinas_kodas (USERID, PASSWORD, EMAIL
ACTIVE, CTIME)
        VALUES ($userid , $randomCode, $email, „1“, date())“;
        If ($conn->query($sql) == TRUE) {
                Fwrite($myfile,‘Sucesfully added’;
                Fclose($myfile); }
        Else
                {
                Fwrite($myfile, „\n Error in “ );
                Fwrite($myfile, $sql );
        Fclose($myfile); }
}


// Išsiunčiamas laiškas vartotojui su jo laikinu kodu
```

```
$message = “Jūsų“. $userid . „ laikinas kodas yra” . $randomCode;
Mail($email,”Neatsakyti”,$message);

//Tikrinimas ar kodas tinkamas ir ar nepraėjo galiojimo laikas

$CData =mysql_query(“SELECT CData FROM Laikinas_kodas where USERID = $userid and PASSWORD = $insertcode and ACTIVE = „1“)
If(empty(CData)){

echo '<script language="javascript">';
echo 'alert("Wrong code")';
echo '</script>';
}
Else {
        If(CData <= date() + $curtime)
                //Tolimesnis kodas
        Elseif {    echo '<script language="javascript">';
                    echo 'alert("Time run away this code is not working")';
                    echo '</script>';
        }
}
```

6.3. Sp_executesql naudojimo pavyzdys

```
        SELECT      @ParameterDefinition = '@userid varchar(10),
                                           @randomCode varchar(12),
                                           @email varchar (30)';
        BEGIN TRY
            BEGIN TRANSACTION

         SELECT @result = N’mysql_query(“SELECT 1 FROM Laikinas_kodas where USERID = @userid LIMIT)’

    If (mysql_fetch_row(@result)){
         SELECT @sql = N’UPDATE Laikinas_kodas SET PASSWORD=@randomCode ,ACTIVE=‘‘1‘‘, CTIME =getdate() WHERE USERID = @userid‘
```

```
}
Else{
        SELECT @sql =N‘INESRT INTO Laikinas_kodas (USERID, PASSWORD, EMAIL
ACTIVE, CTIME)
        VALUES (@userid , @randomCode, @email, ,‘1‘‘, getdate())‘;

            EXECUTE sp_executesql
                    @result,
                    @ParameterDefinition,
                    @ userid = @ userid;



            EXECUTE sp_executesql
                    @sql,
                    @ParameterDefinition,
                    @randomCode = @randomCode,
                    @ userid = @ userid,
                    @email = @email;
            COMMIT TRANSACTION
        END TRY

        BEGIN CATCH
                    ROLLBACK TRANSACTION
        END CATCH

select
```

7. NAUDOTA LITERATŪRA

[1]. „Apie OpenSSL“ [Tinkle]. https://www.openssl.org/ [Kreiptasi 2016 Lapkričio 12]

[2]. „OpenSSL licencijos“ [Tinkle]. https://people.gnome.org/~markmc/openssl-and-the-gpl.html [Kreiptasi 2016 Lapkričio 12]

[3]. „Apie Beringso“ [Tinkle]. https://boringssl.googlesource.com/boringssl/ [Kreiptasi 2016 Lapkričio 12]

[4]. „Apie LibreSSL“ [Tinkle]. http://www.libressl.org/ [Kreiptasi 2016 Lapkričio 12]

[5]. „Apie Ecryptfs“ [Tinkle]. http://ecryptfs.org/about.html [Kreiptasi 2016 Lapkričio 13]

[6]. „Ecryptfs naudojami algoritmai“ [Tinkle]. http://manpages.ubuntu.com/manpages/zesty/en/man1/ecryptfs-setup-private.1.html [Kreiptasi 2016 Lapkričio 13]

[7]. „Apie Ecryptfs“ [Tinkle]. https://www.howtoforge.com/how-to-encrypt-directories-partitions-with-ecryptfs-on-debian-squeeze [Kreiptasi 2016 Lapkričio 13]

[8]. „Apie Truecrypt“ [Tinkle] http://truecrypt.sourceforge.net/ [Kreiptasi 2016 Lapkričio 14]

[9]. „Apie Truecrypt, jo naudojimas ir jo trūkumai“ [Tinkle] https://www.raspberrypi.org/forums/viewtopic.php?f=41&t=6225 [Kreiptasi 2016 Lapkričio 14]

[10]. „Apie EncryptedHome“ [Tinkle] https://help.ubuntu.com/community/EncryptedHome [Kreiptasi 2016 Lapkričio 14]

[11]. „Apie cryptSetup“ [Tinkle] http://paxswill.com/blog/2013/11/04/encrypted-raspberry-pi/ [Kreiptasi 2016 Lapkričio 14]

[12]. „Apie cryptSetup“ [Tinkle] http://open-desk.org/?p=259 [Kreiptasi 2016 Lapkričio 14]

[13]. „CryptSetup luks naudojimo instrukcija“ [Tinkle] http://www.cyberciti.biz/hardware/howto-linux-hard-disk-encryption-with-luks-cryptsetup-command/ [Kreiptasi 2016 Gruodžio 12]

[14]. „Apie cryptSetup luks, naudojimi algoritmai“ [Tinkle] http://askubuntu.com/questions/97196/how-secure-is-an-encrypted-luks-filesystem [Kreiptasi 2016 Gruodžio 12]

[15]. „*Mod_evsasive* naudojimas ir apsaugojimas apache serverį“ [Tinkle] https://www.digitalocean.com/community/tutorials/how-to-protect-against-dos-and-ddos-with-mod_evasive-for-apache-on-centos-7 [Kreiptasi 2017 Sausio 15]

[16]. „SQL injekcijos“ [Tinkle] http://www.techyfreaks.com/2012/05/manual-sql-injection-tutorial.html [Kreiptasi 2017 Sausio 21]

[17]. Hossain Shahriar "Mutation-based testing of buffer overflows, sql injections, and format string bugs" Queen's University Kingston, Ontario, Canada 2008 [žiūrėta 2017-03-20] Prieiga per : https://www.researchgate.net/publication/237237332_MUTATION-BASED_TESTING_OF_BUFFER_OVERFLOWS_SQL_INJECTIONS_AND_FORMAT_STRING_BUGS SQL Injections sprendimo būdai

[18]. „Sesijos užgrobimo atakos. Kas jos yra ir kaip apsisaugoti" [Tinkle]]http://www.acros.si/papers/session_fixation.pdf [Kreiptasi 2017 Sausio 21]

[19]. Omar Elejla „Intrusion Detection Systems of ICMPv6- based DDoS attacks" Universiti Sains Malaysia 2016 [žiūrėta 2017-03-21] Prieiga per: https://www.researchgate.net/publication/311958034_Intrusion_Detection_Systems_of_ICMPv6-based_DDoS_attacks

[20]. Etienne Janot „Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM" Concordia University College of Alberia. 2014 [žiūrėta 2017-03-21] Prieiga per: https://www.researchgate.net/publication/266225425_Preventing_SQL_Injections_in_Online_Applications_Study_Recommendations_and_Java_Solution_Prototype_Based_on_the_SQL_DOM

[21]. Mohamed Rua „The impact of sql injection attacks on the security of databases" - Managment & Science University, Malaysia 2017 [žiūrėta 2017-03-22] Prieiga per : https://www.researchgate.net/publication/316609616_THE_IMPACT_OF_SQL_INJECTION_ATTACKS_ON_THE_SECURITY_OF_DATABASES

[22]. Ian Pye „Lock, deadlock and abstractions: experiences with multi-threaded programming at CloudFlare, Inc" 2012 [ziūrėta 2017-03-19] Prieiga per: https://www.researchgate.net/publication/254004409_Locks_deadlocks_and_abstractions_experiences_with_multi-threaded_programming_at_CloudFlare_Inc

[23]. „Apie CloudFlare" [Tinkle] https://api.cloudflare.com/#getting-started-endpoints [Kreiptasi 2017 Kovo 19]

[24]. Zong-pu Jis „A novel security private cloud solution based on eCryptfs" 2013 [Tinkle 2017-04-01] Prieiga per: https://www.researchgate.net/publication/261166771_A_novel_security_private_cloud_solution_based_on_eCryptfs?ev=srch_pub

[25]. Grant Fritchey „Parameter Sniffing" 2014 [žiūrėta 2017-04-02] https://www.researchgate.net/publication/312707505_Parameter_Sniffing