KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Edvinas Baranauskas

Įterptinių kompiuterinių sistemų programinio kodo apsauga

Baigiamasis magistro darbas

Vadovas
Prof. dr. Egidijus Kazanavičius

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Įterptinių kompiuterinių sistemų programinio kodo apsauga

Baigiamasis magistro darbas
Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Prof. dr. Egidijus Kazanavičius
(data)

Recenzentas
(parašas) Doc. dr. Stasys Maciulevičius
(data)

Projektą atliko
(parašas) Edvinas Baranauskas
(data)

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Edvinas Baranauskas

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga, 621E10003

(Studijų programos pavadinimas, kodas)

„Įterptinių kompiuterinių sistemų programinio kodo apsauga“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano **Edvino Baranausko** baigiamasis projektas tema „Įterptinių kompiuterinių sistemų programinio kodo 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)*

Baranauskas, E. „Įterptinių kompiuterinių sistemų programinio kodo apsauga“. Magistro baigiamasis projektas / vadovas prof. dr. Egidijus Kazanavičius; Kauno technologijos universitetas, informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2016. 62 p.

SANTRAUKA

Produktų ir komponentų piratavimas padaro didžiulę ekonominę žalą. Dažniausiai aukomis tampa kompanijos, kurių produktai pasisavinami ir kopijuojami. Kopijuotos prekės parduodamos tokiomis kainomis, kurios iškraipo rinkos konkuravimą. Manoma, kad globaliu mastu piratinės programinės įrangos yra apie 10 proc. ir tai padaro ekonominę žalą, kuri yra maždaug 300 milijardų eurų. Taip pat prarandama daugiau nei 2,5 milijonai darbo vietų.

Šiuo darbu siekiama išplėsti įterptinių sistemų saugos užtikrinimą per išorinę prieigą. Kitas uždavinys – užtikrinti programinio kodo, kuris saugomas SD kortelėje, ir kompiuterinės sistemos duomenų, saugomų operatyviojoje atmintyje, apsaugą nuo komercinio šnipinėjimo ir piktavališko kodo modifikavimo.

Tyrimo sritis – įterptinių sistemų saugos algoritmai, metodai bei priemonės, skirti apsaugoti „Raspberry Pi“ architektūros kompiuterinėje sistemoje esančią programinę įrangą. Darbo tikslas yra sukurti „Raspberry Pi“ architektūros programinės įrangos apsaugos priemones ir atlikti jų eksperimentinę / probleminę realizaciją bei tyrimą.

Darbe atlikta įterptinių sistemų įsilaužimų būdų bei apsaugos priemonių analizė. Taip pat, atsižvelgiant į algoritmų greitaveiką, energijos suvartojimą, bei disko vietos užimtumą, atlikta kriptografijos algoritmų analizė. Atlikus analizę, sudaryta „Raspberry Pi“ architektūros įterptinės sistemos apsaugos metodologija. Sudarius apsaugos priemonių metodologiją, realizuotas prototipas. Atlikta prototipo veikimo analizė ir parašytos darbo rezultato išvados.

Baranauskas, Edvinas. *Software Security of Embedded System*: *Master's* thesis in Information and Information Technology Security / supervisor assoc. prof. dr. Egidijus Kazanavičius. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Information and Information Technology Security

Key words: Embedded System, Software Security

Kaunas, 2016. 62 p.

SUMMARY

Product piracy and components makes huge economic damage. In most cases, the victims are companies whose products are absorbed and copied. Copied goods are sold at prices which distort the market compete. It is estimated that globally pirated software is about 10 percent and it makes economic damage, which is estimated at 300 billion euros. It also lost more than 2.5 million jobs.

The aim of this work is to expand the embedded systems security assurance through external access. Another challenge - to ensure the program code, that is stored on the SD card, and the computer system data that is stored in the operational memory, protection from commercial spyware and malicious code modification.

Study area - embedded systems security algorithms, methods and measures for software protection of the "Raspberry Pi" architecture computer system. The aim is to create a software security tools for "Raspberry Pi" architecture and carry out their experimental / problematic realization and investigation.

The analysis of embedded systems hacking techniques and safeguards is made. Also, according to the speed, power consumption, and disk space availability of algorithms, cryptographic algorithms analysis is made. After analysis "Raspberry Pi" architecture embedded system security methodology is concluded. Include security methodology implemented prototype. Prototype performance analysis is made and written conclusions from the results.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

DES (angl. *Data Encryption Standart*) – duomenų šifravimo standartas.

3DES (angl. *Triple Data Encryption Standart*) – trigubas duomenų šifravimo standartas.

AES (angl. *Advanced Encryption Standart*) – sudėtingesnis šifravimo standartas.

RC4 (angl. *Ron's Code 4*) – Rono Kodas 4.

CAST – simentrinis šifravimo algoritmas.

Blowfish – simentrinis šifravimo algoritmas.

Užkrovimo nepaleidus ataka (angl. *cold boot attack*) – ataka, kurios metu užpuolikas gali gauti šifravimo raktus iš operatyviosios atminties.

Jėgos ataka (angl. *brute-force attack*) – kriptoanalizės metodas, kai išbandomi visi galimi rakto ar slaptažodžio variantai.

Užprogramuotasis (*angl. hard-coded*) – įvesties parametrai įterpti tiesiai į programinį kodą.

Rankos paspaudimo autentifikavimo protokolas (angl. *Challenge-Handshake Authentication Protocol*) – protokolas atsakingas už besijungiančių šalių autentifikavimą ir apsikeitimą kodais, reikalingais užmegzti ar atnaujinti ryšio seansą.

RSA – viešojo rakto šifravimo algoritmas.

Diffie-Hellman – viešojo rakto šifravimo algoritmas.

DSA – viešojo rakto šifravimo algoritmas0.

Centrinis serveris – serveris, su kuriuo komunikuoja įterptinės sistemos įrenginys.

DB (angl. *database*) – duomenų bazė.

SSL sertifikatas (angl. *SSL certificate*) – tai elektroninis dokumentas, padedantis klientams nustatyti svetainės tapatybę ir užšifruoti tarp kliento ir serverio siunčiamą informaciją.

SD kortelė (angl. *SD card*) – atminties kortelė.

RAM (angl. *Random Access Memory*) – operatyvioji atmintis.

OpenSSL – programinės įrangos biblioteka.

LED (angl. *light-emitting diode*) – šviesos diodas.

Aparatinis saugumo raktas (angl. *dongle*) – nedidelis įtaisas, jungiamas prie kompiuterio per vieną iš jo išorinių jungčių. Naudojamas informacijai apsaugoti nuo nesankcionuoto kopijavimo.

OS (angl. *operating system*) – operacinė sistema

MAC (angl. *Media Access Control*) adresas – unikalus 12 simbolių tinklo įrenginio adresas.

Maskavimas (angl. *obfuscation*) – darymas sunkiau suprantamu, dviprasmišku.

Maišos funkcija (angl. *hash function*) – funkcija f = f (x), priskirianti argumentui x pseudoatsitiktinį skaičių, vadinamą maišos kodu (angl. hash code). Tam pačiam argumentui funkcija visada turi duoti tokį patį rezultatą, taigi ji nėra atsitiktinė.

Maišos reikšmė (angl. *hash value*) – reikšmė, kurią apskaičiuoja maišos funkcija.

Kontrolinė suma (angl. *checksum*) – reikšmė, laikoma arba persiunčiama kartu su duomenimis ir apskaičiuojama įvykdžius tam tikrą matematinę funkciją su tais duomenimis. Naudojama patikrinti, ar sudarantys duomenys nebuvo pakeisti (pažeisti).

Baitų kodas (angl. *byte code*) – taip pat žinomas kaip p-kodas (portatyvus kodas), yra nurodymų forma rinkinys, sukurtas interpretatoriui veiksmingai vykdyti programinę įrangą.

SHA-1 – kriptografinė maišos funkcija.

SHA-256 – kriptografinė maišos funkcija.

Įterptinė sistema (angl. *embedded system*) – tai į kitą įrenginį įterpta elektroninė sistema, kuri valdo tą įrenginį pagal jai užduotą programą.

Apgrąžos inžinerija (angl. *reverse engineering*) – gaminio analizės būdas, norint nustatyti, iš ko ir kaip jis pagamintas.

ĮVADAS

Šis darbas priklauso informacijos ir informacinių technologijų saugos studijų programai.

Įterptinė sistema – tai į kitą įrenginį įterpta elektroninė sistema, kuri valdo tą įrenginį pagal joje įdiegtą programą. Įterptinėse sistemose pagrindinė sudedamoji dalis yra skaitmeninis procesorius arba branduolys.

Sprendžiama problema:

Įterptinės kompiuterinės sistemos, programinės įrangos ir kodo apsaugos metodų sukūrimas ir tyrimas. Šiuo darbu siekiama išplėsti saugos užtikrinimą per išorinę prieigą. Kita problema – užtikrinti programinio kodo, kuris saugomas SD kortelėje, ir kompiuterinės sistemos duomenų, saugomų operatyviojoje atmintyje (RAM), apsaugą nuo komercinio šnipinėjimo ir piktavališko kodo modifikavimo.

Tyrimo sritis ir objektas:

Įterptinių sistemų saugos algoritmai, metodai bei priemonės, skirti apsaugoti „Raspberry Pi“ architektūros kompiuterinėje sistemoje esančią programinę įrangą, kuri atlieka energijos apskaitos funkciją, atsižvelgdama į kritinius algoritmų / programų parametrus (greitaveiką, kodo apimtį, baterijos ilgaamžiškumą, saugos lygį ir kt.).

Darbo tikslas ir uždaviniai:

Sukurti „Raspberry Pi“ architektūros programinės įrangos apsaugos priemones ir atlikti jų eksperimentinę / probleminę realizaciją bei tyrimą.

Uždaviniai**:**

- Atlikti įterptinių sistemų / architektūrų ir jų saugos priemonių analizę;
- Atlikti duomenų ir programų šifravimo / dešifravimo algoritmų analizę;
- Sudaryti įterptinių sistemų programinės įrangos saugos priemonių struktūrą;
- Sudaryti įterptinių sistemų programinės įrangos apsaugos algoritmą ir atlikti metodų validavimą;
- Sudaryti programinės įrangos saugos užtikrinimo bei diegimo metodiką;
- Atlikti praktinę įterptinės sistemos programinės įrangos saugumo realizaciją;
- Atlikti praktinę saugos algoritmo realizaciją ir jo tyrimą.

Darbo rezultatai ir jų svarba:

Sukurtos „Raspberry Pi“ architektūros programinės įrangos apsaugos priemonės, atsižvelgiant į kritinius algoritmų / programų parametrus. Programinis kodas bus apsaugotas nuo komercinio šnipinėjimo ir piktavališko kodo modifikavimo.

Darbo struktūra:

1. Programinės įrangos ir duomenų saugos užtikrinimo analizė:
    a. Programinės įrangos saugumo situacija rinkoje;

    b. Įterpimų sistemų / architektūrų ir jų saugos priemonių analizė;
    c. Duomenų ir programinės įrangos šifravimo / dešifravimo algoritmų analizė;
    d. Išvados.
2. Programinės įrangos apsaugos metodologija:
    a. Pradinė konfigūracija;
    b. Programinės įrangos apsauga;
    c. Išvados.
3. Apsaugos priemonių prototipo realizacija:
    a. Įterptinės sistemos įrenginio apsaugos priemonės ir programinės įrangos veikimas;
    b. Centrinio serverio apsauga ir programinės įrangos veikimas;
    c. Išvados.
4. Įterptinės sistemos programinio kodo apsaugos prototipo analizė:
    a. Įterptinės sistemos parametrų pasikeitimų analizė;
    b. Įterptinės sistemos programinio kodo apsaugos prototipo veikimo analizė;
    c. Išvados
5. Išvados.
6. Literatūra.
7. Priedai.

1. PROGRAMINĖS ĮRANGOS IR DUOMENŲ SAUGOS UŽTIKRINIMO ANALIZĖ

Programinės įrangos ir duomenų saugumo užtikrinimo analizė skirta surasti galimus saugumo trūkumus įterptinėse sistemose ir jų apsaugos būdus. Todėl šiame skyriuje analizuojami mikrovaldikliai ir jų saugumo trūkumai, siūlomi apsaugos metodai bei šifravimo / dešifravimo algoritmai.

1.1. Programinės įrangos saugumo situacija rinkoje

Produktų piratavimo neigiamas poveikis – tai grėsmė, kuri turi būti įvertinta rimtai. Tai paveikia ne tik originalios įrangos gamintojus, bet ir vartotojus, nes jie apgaudinėjami. Dėl nelegalių kopijų naudojimo gamintojai praranda didelę rinkos dalį, nes dažniausiai tokia įranga būna pigesnė. Vartotojai nesąmoningai naudoja abejotino saugumo ir patikimumo produktus.

Mažinant investicijas į produktą, atsiranda darbo ir mokesčių sumažėjimas, o tokie nuostoliai silpnina ekonomiką.

Produktų ir komponentų piratavimas padaro didžiulę ekonominę žalą, kuri pastoviai siekia naujus rekordus. Dažniausios aukos yra kompanijos, kurių produktai yra pasisavinami ir kopijuojami. Kopijuotos prekės parduodamos tokiomis kainomis, kurios iškraipo rinkos konkuravimą. Vartotojai taip pat patiria neigiamas pasekmes, pavyzdžiui, jei nestandartinė abejotino saugumo ir patikimumo klastotė yra parduodama nieko neįtariančiam pirkėjui, kuris mano, kad jis perka originalų firmos gaminį. Manoma, kad globaliu mastu, piratinės programinės įrangos yra apie 10 proc. ir tai padaro apie 300 milijardų eurų ekonominę žalą. Taip pat prarandama daugiau nei 2,5 milijonai darbo vietų.

Atsižvelgiant į esamą situaciją rinkoje, nesunku padaryti išvadas, kad įterptinių sistemų apsauga nuo piratavimo ir komercinio šnipinėjimo yra tikrai svarbi [1].

1.2. Įterpimų sistemų / architektūrų ir jų saugos priemonių analizė

1.2.1. Mikrovaldiklių architektūrų analizė

„Arduino“

Galima prijungti keletą LED, jutiklių tiesiogiai į plokštę. Norint programuoti „Arduino“, reikia specialios programinės įrangos, sukurtos būtent jam (ją galima parsisiųsti). Iš esmės su šia programine įranga galima įkelti kodą tiesiogiai į „Arduino“ per USB jungtį. Įkėlus kodą, galima atjungti USB kabelį ir prijungti bateriją, paleisti programą. Sparta 16 MHz, darbinė atmintis 2 KB, o jo paties atmintis 32 KB. Neturi interneto jungties. Turint mažai atminties yra ribojamas programinės įrangos dydis. Šis mikrovaldiklis negali būti naudojamas kaip serveris ir negalima vidinė duomenų bazė. Programinė įranga iš šio įrenginio gali būti pasisavinama prijungiant jį prie kompiuterio arba naudojant užkrovimo nepaleidus ataką (*angl. cold boot attack*) [2].

„Raspberry Pi“ modelis B

„Raspberry Pi“ veikimui reikalinga operacinė sistema. Viskas saugojama SD kortelėje, kurios talpa gali būti iki 32 GB. Šį prietaisą galima prijungti prie interneto. Jo procesorius – ARM1176JZF-S 700 MHz. Prie „Raspberry Pi“ galima prijungti klaviatūrą, pelę, monitorių, paleisti „Linux“ operacinę sistemą. Tinka įterptinėms sistemoms ar projektams, kurie reikalauja daugiau interaktyvumo ar didesnės duomenų apdorojimo galios. Turi 17 skaitmeninių bendrosios paskirties įėjimų / išėjimų. Darbinė atmintis siekia 512 MB. Šį mikrovaldiklį galima naudoti kaip serverį ir į jį įdiegti vidinę duomenų bazę. RAM atmintis integruota kartu su CPU, todėl sistema nėra atspari paleidimo išjungus atakai. Esant tokios atakos galimybei negalima laikyti šifravimo raktų atmintyje, nes jie gali būti nuskaityti. Jeigu galima fizinė prieiga prie įrenginio, galima išimti SD kortelę ir naudojant kompiuterį nuskaityti duomenis arba pakeisti prisijungimo slaptažodį [2].

Tai, kad šis įrenginys gali būti pajungtas prie interneto sumažina jo saugumą. Neužtenka vien tik to, kad bus paleistos OS saugumo funkcijos. Taip pat reikia žiūrėti, kad atsisiųsta trečiųjų šalių programinė įrangą būtų iš patikimų šaltinių. Siunčiantis iš nepatikimų šaltinių programinė įranga gali būti įdiegiama kartu su kenkėjiška programine įranga, kuri gali sukelti duomenų vagystę. Jeigu „Raspberry Pi“ integruotas su kitais įrenginiais, taip pat ir jų programinė įranga turi būti patikima ir atnaujinta [3].

ARM1176JZF-S turi „TrustZone“ saugumo plėtinį. „TrustZone“ plėtinys įjungia programinės įrangos saugumo aplinką. Ši technologija nesaugo techninės įrangos nuo atakų, tad reikia imtis papildomų priemonių, norint apsaugoti techninę įrangą ir patikimą kodą. Patikimas kodas yra santykinai mažas blokas, kuris veikia saugioje procesoriaus vietoje ir suteikia saugumą visai sistemai [4].

„BeagleBone Black“

Šis mikrovaldiklis turi 1 GHz ARM Cortex-A8 procesorių ir 512 MB darbinės atminties. Jo vidinė atmintis yra 2 GB. Turi 65 skaitmeninius bendrosios paskirties įėjimus / išėjimus. Palaiko tokias operacines sistemas kaip „Linux“, „Android“, „Ubuntu“ ir daug kitų. Turi interneto jungtį. Taip pat šį mikrovaldiklį galima naudoti kaip serverį ir įdiegti vidinę duomenų bazę. Galimos tos pačios atakos, kaip ir „Raspberry Pi“ įrenginiui [5].

„PCDuino“

„PCDuino“ – PC miniplatforma, kuri veikia kaip personaliniai kompiuteriai, todėl kad turi operacinę sistemą. Palaiko „Android“ ir „Ubuntu“ operacines sistemas. Suderinamas su „Arduino“. Palaiko *Python, C* ir kitas programavimo kalbas. „PCDuino“ turi 1GHz ARM Cortex A8 procesorių ir 1 GB operatyvinės atminties. Taip pat, kaip ir „BeagleBoneBlack“, jis turi 2 GB vidinę atmintį. Į jį dedasi SD kortelė, kurios talpa gali būti iki 32 GB. Jį galima naudoti kaip serverį ir įdiegti vidinę duomenų bazę. Galimos tos pačios atakos, kaip ir „Raspberry Pi“ įrenginiui [2].

1.2.2. Programinės įrangos, esančios įterptinėje sistemoje, saugumo grėsmės bei apsaugos būdai

Šiame darbe nesprendžiama įterptinių sistemų aparatūros ir architektūros sprendimų. Todėl šiame skyriuje analizuojami tik programinės įrangos pasisavinimo ir apsaugos būdai, o ne visos įterptinės sistemos apsauga.

Įterptinės sistemos sudėtingumas

Kuo sudėtingesnė įterptinė sistema, tuo daugiau galimų įsilaužimo būdų. Linijinis techninės / programinės įrangos augimas sukuria kur kas didesnį nei linijinį sudėtingumo augimą. Todėl tampa sudėtingiau nustatyti visus galimus įsilaužimo būdus. Sudėtingumo trūkumas gali būti panaudotas įsilaužiant į įterptinę sistemą. Tad labai svarbu kontroliuoti sudėtingumą atsižvelgiant į saugumą [6].

Nuotolinė prieiga

Tradiciškai galvojama, kad įterptinė sistema veikia kaip atskiros dalys, apsaugotos nuo interneto pavojaus. Bet yra daugelis priežasčių, kodėl jose naudojama prieiga prie interneto. Prijungimas prie interneto suteikia nuotolinio valdymo galimybę. Į valdymo užduotis įtraukiamas ir sistemos programinės įrangos atnaujinimas. Interneto ryšys atveria naujas panaudojimo galimybes, bet taip pat padidina ir įsilaužimo galimybes [7].

Norint apsaugoti įrenginį nuo įsilaužimo iš nuotolinės prieigos, pirmiausia reikia atlikti pradinę sistemos konfigūraciją. Į pradinę konfigūraciją įtraukiama nereikalingų procesų išjungimas, sistemos OS atnaujinimas, sistemos numatyto vartotojo bei slaptažodžio pakeitimas ir t.t. Jei įterptinė sistema dirba kaip serveris, reikia atlikti serverio apsaugos konfigūraciją.

Prisijungimui per SSH galima naudoti raktų poros autentifikavimą. Jį panaudoti yra daug saugiau nei paprastai suvesti slaptažodį, norint autentifikuoti naudotoją [8].

Perduodamas turinys turi būti autentifikuojamas ir validuojamas tiek įterptinėje sistemoje, tiek kitoje sistemoje, su kuria įterptinė sistema yra susieta. Taip pat visi duomenys turi keliauti saugiu tinklu, kad jų nebūtų galima perskaityti. Sistemai pastebėjus duomenų neatitikimus, turi būti imtasi veiksmų. Jei įrenginys priima blogus duomenis iš tariamai patikimos sistemos, tada tai gali būti panaudojama įsilaužimui į įrenginį. Pats įrenginys turi turėti unikalų identifikacijos numerį, kuriuo remiantis galima jį autentifikuoti [9].

Vietinė ar nuotolinė prieiga prie įrenginio

Ši sąlyga reikalauja, kad užpuolikas turėtų tam tikrų privilegijų, kurios leidžia prieigą prie įrenginio paslaugų arba funkcijų. Ši prieiga gali būti apribota vietinės prieigos arba nuotolinės prieigos galimybėmis. Dažniausiai tokia prieiga reikalauja paprasto vartotojo, o ne administratoriaus privilegijų. [10]

Tiesioginė prieiga prie įrenginio

Tiesioginis fizinis ryšys reiškia, kad užpuolikas gali prieiti prie fizinio įrenginio. Užpuolikui nereikia jokių privilegijų prie prietaiso paslaugų [10].

Fiziškai priartėjęs užpuolikas

Užpuolikui nebūtina fizinė prieiga prie įrenginio. Jai naudojamas bevielis interneto ryšys, tai užpuolikui gali pakakti būti bevielio ryšio diapazone, kad galėtų prisijungti prie įrenginio. Taip neturint prieigos prie pačio įrenginio, gali būti perimami tinklu keliaujantys duomenys [10].

Bloga sistemos konfigūracija

Sistema gali būti apsaugota, bet blogai sukonfigūruota. Taip atsiranda saugumo spragos, kuriomis užpuolikas gali pasinaudoti [10].

Programavimo klaidos

Daugelis pažeidžiamumų atsiranda dėl programavimo klaidų, kurios sukelia kontrolės srauto atakas (*angl. control flow attacks*) [10].

Silpna prieigos kontrolė ar autentifikavimas

Daugelis prietaisų naudoja numatytuosius arba silpnus slaptažodžius. Kai kurie prietaisai turi užprogramuotuosius (*angl. hard-coded*) slaptažodžius, kurie palieka atsarginį įėjimą tiems, kas juos žino. Toks pažeidžiamumas leidžia užpuolikui su minimaliomis pastangomis apeiti prieigos kontrolės mechanizmus [10].

Netinkamai naudojama kriptografija

Kai kuriuose įrenginiuose kriptografija yra naudojama autentifikavimo arba konfidencialių duomenų slėpimo tikslais. Dažnai kriptografija naudojama netinkamai, o tai priveda prie kritinės saugumo klaidos. Pavyzdžiui: naudojamas silpnas atsitiktinių skaičių generatorius kriptografijos raktų generavimui [10].

Kenkėjiška programinė įranga

Užpuolikas gali mėginti užkrėsti įterptinę sistemą kenkėjiška programine įranga. Kenkėjiškų programinių įrangos yra įvairių tipų. Bendra savybė ta, kad jos visos turi nepageidaujamą, potencialiai žalingą funkcionalumą, kuris pridedamas prie užkrėstos sistemos. Užkrėstas įrenginys gali pakeisti savo elgesį ir taip pridaryti daug žalos [10].

Paruošto paketo arba įvesties įterpimas

Tai metodas, kuris naudojamas prieš protokolus, naudojamus įterptinės sistemos įrenginyje. Panaši ataka yra įvesties pakeitimas įterptinės sistemos įrenginyje. Tiek paketų, tiek įvesties

apdorojimo atakos pasinaudoja vykdymo pažeidžiamumais protokoluose arba kitose programinės įrangos vietose [10].

Slaptas pasiklausymas

Tai pasyvus atakos metodas, kurio metu užpuolikas perima siunčiamus arba gautus duomenis iš įterptinės sistemos prietaiso. Tose žinutėse gali būti slaptos informacijos, kuri yra silpnai apsaugota arba neapsaugota visiškai. Taip pat gauta informacija gali būti panaudota kitiems atakų būdams [10].

Paleidimo neužkrovus ataka

Paleidimo neužkrovus ataka įmanoma dėl to, kad visi duomenys, kurie yra saugojami operatyviojoje atmintyje (RAM), išjungus maitinimą nėra iškart prarandami. Jie gali būti pasiekiami dar keletą sekundžių. Negana to, įmanoma šį periodą padidinti iki kelių minučių arba valandų. Todėl per šį trumpą periodą duomenys iš operatyviosios atminties gali būti nuskaitomi. Šis atakos būdas sukurtas tam, kad būtų galima pasiekti informaciją užšifruotame diske. Dauguma sistemų slaptą raktą laiko operatyviojoje atmintyje. Yra keli apsisaugojimo būdai su skirtingais efektyvumo laipsniais.

Vienas akivaizdus apsaugos būdas yra nesaugoti slaptojo rakto operatyviojoje atmintyje ilgiau nei būtina. Panaudojus slaptą raktą, jį reikia pašalinti iš operatyviosios atminties ir tą vietą perrašyti kitu informacijos bloku.

Kitas būdas yra nesaugoti slaptų raktų RAM atmintyje. Spendimas – branduolio modifikacija, kurios metu pakeičiama slapto rakto saugojimo vieta už RAM atminties ribų.

Taip pat galima apsisaugoti nuo šios atakos, pasirenkant techninę įrangą, kuri atspari tokiai atakai. Pavyzdžiui: duomenis, laikomus operatyviojoje atmintyje, užšifruoti ir dešifruoti kai jie yra naudojami [11].

Jėgos (*angl. brute-force*) ataka

Silpna kriptografija arba silpni autentifikavimo metodai gali būti pažeisti brutalios jėgos paieškos atakos. Ji apima pagrindines paieškos atakas, nukreiptas prieš šifruotus algoritmus, tokius kaip šifrai ir MAC funkcijos, žodyno atakas, nukreiptas prieš slaptažodžius, naudojamus tapatumui nustatyti. Ši ataka yra įmanoma tik tada, kai paieškos erdvė yra pakankamai maža [10].

Paprasto vartotojo pieiga

Tai ataka, kai naudojantis sistema kaip paprastam vartotojui, išnaudojami nesaugūs įrenginiai arba protokolai. Taip teisėtas naudotojas gali išnaudoti neapsaugotus mechanizmus. Pavyzdžiui: užpuolikas gali pasiekti failus esančius įterptinės sistemos įrenginyje kaip legalus vartotojas, jei pats įrenginys neturi integruoto prieigos kontrolės mechanizmo [10].

Paslaugų blokavimas

Yra daugelis atakų, kurios priveda prie paslaugų blokavimo. Tokios atakos metu įrenginys pradeda blogai veikti arba visiškai sustabdomas jo darbas [10].

Vientisumo pažeidimas

Tai atakos padarinys, kai pažeidžiamas kai kurių duomenų ar prietaiso kodo vientisumas. Tai apima failų ir konfigūracijos nustatymų modifikaciją. Taip pat apima neteisėtą programinės įrangos atnaujinimą įrenginyje [10].

Programinė įranga SD kortelėje

Tokiuose įrenginiuose kaip „Raspberry Pi“ visa programinė įranga yra laikoma SD kortelėje. Todėl, jei programinis kodas yra neužšifruotas, jį galima pasiekti išimant kortelę iš įrenginio ir įdedant į kompiuterį. Taip programinę įrangą galima kopijuoti arba modifikuoti. Vienintelis būdas apsaugoti programinę įrangą – ją užšifruoti. Bet slapto rakto laikymas toje pačioje SD kortelėje didelės naudos neduotų, nes įsibrovėlis juo pasinaudojęs gali dešifruoti programinę įrangą.

Programinės įrangos apgrąžos inžinerija

Norint pasiekti tinkamo lygio sistemos apsaugą, reikia atskirai išanalizuoti programinės ir techninės įrangos apsaugos būdus.

Prieiga prie mikrovaldiklio kompiuterinės kalbos arba FPGA vykdymo, suteikia galimybę, visą programinės įrangos logiką ištraukti iš lusto. Žinios, panaudotos procesoriaus ir jo komandų rinkinyje, leidžia mašininį kodą atversti į asemblerio kodą. Kadangi asemblerio kodas dažniausiai nėra pakankamai suprantamas, egzistuoja papildomi įrankiai (dekompiliatoriai), kurie paverčia asemblerio kodą į geriau suprantamą pseudo kodą.

Jei kas nors geba perskaityti operatyviąją atmintį, tokią kaip RAM arba kaip nors kitaip išgauti mikroprogramos kodą, tuomet turimos inžinerijos ir susijusių sričių žinios dažnai padeda išgauti svarbią informaciją, rinkmenų nustatymus arba slaptus parametrus. Ši informacija taip pat suteikia galimybę taikyti mikroprogramoje manipuliacijas. Saugumo užklausos mikroprogramoje gali būti pažeistos arba pakeistos. Praktiniai pavyzdžiai apima nesankcionuotą programinės įrangos atrakinimą (privilegijų, teisių išplėtimą). Tuomet, kai tik gaunama prieiga prie kodo, ji gali būti lengvai perleidžiama struktūriškai identiškiems lustams. Nors kodo kopijos patvirtinimas nėra paprastas procesas, yra būdų kaip gauti skaitmeninį vandens ženklą kodo lygyje arba įrodyti, pasitelkiant statistinius metodus, kad atsitiktinai grįžtančios tokios pat kodinės sekos tikimybė yra per maža. Jei kopija gali būti patikrinta, galima imtis teisėtų priemonių prieš klastojimą.

Daugelis šalių pasitelkia priemones tam, kad apeitų programinės įrangos kopijavimo apsaugos mechanizmą komerciniais tikslais. Be to, stengdamosi apsaugoti nelegalias programinio kodo kopijas,

dauguma kompanijų nenori atskleisti to, kaip jie integravo programinę įrangą į savo techninę įrangą. Metodai arba skaičiavimai, kurie vystėsi įmonės viduje, yra dažnai lemiamas konkurencingas pranašumas, todėl jie turi būti saugomi. Gamintojai didelį dėmesį skiria apsaugai nuo produktų kopijavimo ir apgrąžos inžinerijai. Dėl šios priežasties tiek programinės įrangos programuotojai, tiek dizaineriai siekia apsaugoti savo kūrinius nuo šnipinėjimo. Iš tikro, nėra jokių ypatingų metodų, kurie apsaugotų programinę įrangą nuo trečiųjų šalių atakų. Efektyvi apsauga prieš duomenų šnipinėjimą ar atvirkštinį programinės įrangos produktų projektavimą reikalauja atitinkamo techninės įrangos palaikymo. Jei pabaigtos programinės įrangos dvejetainis kodas yra pasiekiamas, tuomet galima išanalizuoti apsaugos matus, bet tik tuos, kurie yra pagrįsti programine įranga. Pavyzdžiui, mažai naudinga įtraukti kriptografinį procesą ar raktą į programinį kodą tam, kad apsaugotų kitus programinės įrangos komponentus. Taip yra todėl, kad, nepaisant visko, programinė įranga negali būti visiškai apsaugota nuo modifikavimo.

Būtini kriptografiniai procesai ir raktai visada yra laikomi techninės įrangos modulio viduje ir nedalyvauja programinės įrangos dvejetainiame kode. Jie negali būti ištraukti iš programinės įrangos – nei per statistinę dvejetainio kodo analizę, nei simuliuojant programinę įrangą virtualioje aplinkoje. Techninė ir programinė įranga formuoja funkcinį vienetą ir nėra įmanoma keisti ar analizuoti užšifruotos programinės įrangos komponentų be specialios programinės įrangos.

Programinės įrangos apsaugos matai, pagrįsti technine įranga, nėra savaime saugūs. Pirmieji aparatinio saugumo rakto (angl. *dongle*) sprendimai yra to pavyzdžiai: tikslas – tik nustatytos tapatybės vartotojams, t.y., tiems, kurie fiziškai turi aparatinį saugumo raktą, leisti naudoti programinės įrangos produktus. Pirmųjų aparatinių saugumo raktų sprendimų trūkumai buvo tai, kad programinės įrangos dvejetainis kodas buvo lengvai išanalizuojamas. Jei užklausą apie aparatinio saugumo rakto egzistavimą buvo galima įdėti dvejetainiame kode, šią užklausą buvo galima lengvai praleisti, pavyzdžiui, keičiant asemblerio „Branch Equal“ komandą į „Branch Not Equal“. Jei rinkmena buvo modifikuota pasitelkiant šį būdą, ji galėjo būti vykdoma visiškai nepasitelkiant aparatinio saugumo rakto.

Šiuolaikiniai sprendimai nėra taip lengvai pergudraujami. Iš pateiktų pavyzdžių galima pasimokyti, kad programinės įrangos produktai turi būti apsaugoti nuo manipuliavimo, tam, kad techninės įrangos apsauga teiktų rezultatus. Egzistuoja gausybė įvairiausių, pažangiausių įrankių, kuriuos lengva susirasti internete, ir galima panaudoti programinei įrangai manipuliuoti ir perprojektuoti. Produktas, kurio pagrindą sudaro įsitvirtinusi programinė įranga yra lengviau apsaugomas, nes gamintojas paprastai turi daugiau techninės ir programinės įrangos konfigūracijos pasirinkimų savo diapazone, ypač, jei visi komponentai buvo gauti iš vieno šaltinio ir gali koordinuoti vienas su kitu (saugumo požiūriu).

Pagrindinis saugumo tikslas gali būti užkirsti įsilaužėliui kelią suprasti programinės įrangos tikslą bei įrangos komponentų ir vidaus funkcijų funkcionalumą. Šiam tikslui naudojamos priemonės, kurios maskuoja programinės įrangos kodą [1].

1.3. Duomenų ir programinės įrangos šifravimo / dešifravimo algoritmų analizė

Kriptografija, tai mokslas naudojantis matematiką užšifruoti ir dešifruoti duomenims. Kriptografija leidžia saugoti ypač slaptą informaciją ir siųsti ją nesaugiais tinklais, taigi jos negali perskaityti niekas kitas, kaip tik gavėjas.

Kriptografijos saugumui tikrinti yra naudojama kriptoanalizė. Paprasta kriptoanalizė – tai loginės kombinacijos, matematinių įrankių panaudojimas, kantrybė, pasiryžimas ir, žinoma, – sėkmė.

1.3.1. Simetriniai šifravimo algoritmai

Simetriniai šifravimo algoritmai yra pagrįsti E (M, K) = C užšifravimu ir D (C, K) = M dešifravimu, naudojant bendrą slaptą raktą K. Visi simetriniai užšifravimo metodai skirstomi į blokinius ir srautinius šifrus.

Srautinis šifras užšifruoja pavienius bitus ir jo užšifravimo rakto ilgis yra lygus užšifruojamų duomenų ilgiui. Dėl vykdomų šifravimo operacijų, kiekvieną kartą užšifruojant tuos pačius duomenis, gaunamas skirtingas raktas. Šio šifro problema – sunku sutarti dėl slaptų raktų tarp dviejų šalių [12].

1.1 lentelėje pateikiama dažniausiai naudojamų srautinių ir blokinių šifrų apžvalga.

1.1 lentelė. Srautinių ir blokinių šifrų apžvalga

Algoritmas	Blokas (bitais)	Slaptas raktas (bitais)
DES	64	56
3DES	64	112, 168
AES (Rijndael)	128, 192, 256	128, 192, 256
Blowfish	64	32 – 448
CAST	64	40 – 128
RC4	Srautinis šifras	> 2048

1.3.1.1. *Algoritmų palyginimo taškų sistema*

1.3.1.1.1. Tikslumas ir patikimumas

Tikslumas ir patikimumas yra pagrindinės kriptografinės technikos. Norint patikrinti tikslumą, kiekvienas algoritmas turi būti patvirtintas ir patikrintas. Patvirtinimas yra įvykdomas palyginant įvesties tekstą su išvesties tekstu po iššifravimo. Daugialypis užšifravimas ir žinutės atkodavimas garantuoja technikos patikimumą [13].

1.3.1.1.2. Trukmė

Dėl procesoriaus greičio apribojimo, šifravimo trukmė yra labai svarbus kriterijus įterptinėms sistemoms. Dėl to šio kriterijaus maksimali reikšmė – 50 taškų.

1.2 lentelė. Lyginamoji vykdymo trukmės balų skalė

Diapazonas (ms)	Taškai	Diapazonas (ms)	Taškai	Diapazonas (ms)	Taškai
0 – 8	50	136 – 144	33	272 – 280	16
8 – 16	49	144 – 152	32	280 – 288	15
16 – 24	48	152 – 160	31	288 – 296	14
24 – 32	47	160 – 168	30	296 – 304	13
32 – 40	46	168 – 176	29	304 – 312	12
40 – 48	45	176 – 184	28	312 – 320	11
48 – 56	44	184 – 192	27	320 – 328	10
56 – 64	43	192 – 200	26	328 – 336	9
64 – 72	42	200 – 208	25	336 – 344	8
72 – 80	41	208 – 216	24	344 – 352	7
80 – 88	40	216 – 224	23	352 – 360	6
88 – 96	39	224 – 232	22	360 – 368	5
96 – 104	38	232 – 240	21	368 – 374	4
104 – 112	37	240 – 248	20	374 – 380	3
112 – 120	36	248 – 256	19	382 – 400	2
120 – 128	35	256 – 264	18	400 – 408	1
128 – 136	34	264 – 272	17	> 408	0

Atskirai atliekami šifravimo $T_{šif}$ ir dešifravimo $T_{deš}$ laikų matavimai. Dėl atsitiktinių klaidų matavimai kartojami n kartų. 1.2 lentelėje taškai suskirstyti pagal bendrą šifravimo ir dešifravimo laiką T_{bend} [13].

Šifravimo laiko vidurkis apskaičiuojamas naudojant formulę:

$$\overline{T}_{šif} = \frac{1}{n}\sum_{i=1}^{n} T_{šif_i} \qquad (1.1)$$

Dešifravimo laiko vidurkis apskaičiuojamas naudojant formulę:

$$\overline{T}_{deš} = \frac{1}{n}\sum_{i=1}^{n} T_{deš_i} \qquad (1.2)$$

Bendra šifravimo / dešifravimo trukmė T_{bend} apskaičiuojama pagal formulę:

$$T_{bend} = \overline{T}_{šif} + \overline{T}_{deš} \qquad (1.3)$$

1.3.1.1.3. Energijos sunaudojimas

Šis kriterijus leidžia palyginti vykdomus algoritmus. Dėl energijos sunaudojimo svarbumo, šio kriterijaus maksimali reikšmė – 30 taškų.

1.3 lentelė. Lyginamoji energijos sunaudojimo balų skalė

Diapazonas (µWh)	Taškai	Diapazonas (µWh)	Taškai	Diapazonas (µWh)	Taškai
0 – 0,4	30	4,0 – 4,4	20	8,0 – 8,4	10
0,4 – 0,8	29	4,4 – 4,8	19	8,4 – 8,8	9
0,8 – 1,2	28	4,8 – 5,2	18	8,8 – 9,2	8
1,2 – 1,6	27	5,2 – 5,6	17	9,2 – 9,6	7
1,6 – 2,0	26	5,6 – 6,0	16	9,6 – 10,0	6
2,0 – 2,4	25	6,0 – 6,4	15	10,0 – 10,4	5
2,4 – 2,8	24	6,4 – 6,8	14	10,4 – 10,8	4

2,8 – 3,2	23	6,8 – 7,2	13	10,8 – 11,2	3
3,2 – 3,6	22	7,2 – 7,6	12	11,2 – 11,6	2
3,6 – 4,0	21	7,6 – 8,0	11	11,6 – 12,0	1

Energija E gaunama paskaičiuojant srovę I, įtampą U ir bendrą šifravimo / dešifravimo laiką T_{bend}. Atlikus bandymą pastebėta, kad I ir U nekinta, tad jų vidurkiai neskaičiuojami [13].

Sunaudota energija vienam šifravimo / dešifravimo ciklui įvykdyti paskaičiuojama pagal formulę:

$$\boldsymbol{E = U * I * T_{bend}} \qquad (1.4)$$

1.3.1.1.4. Atminties reikalavimas

Atminties dydis yra dar vienas svarbus palyginimo kriterijus. Dėl griežto saugomos atminties apribojimo, ne visi kriptografiniai metodai gali būti panaudojami. Kai kurios sistemos turi ribotą atminties talpą kriptografinėms technikoms, todėl šio kriterijaus maksimali reikšmė – 20 taškų.

1.4 lentelė. Lyginamoji disko atminties sunaudojimo balų skalė

Diapazonas (B)	Taškai	Diapazonas (B)	Taškai	Diapazonas (B)	Taškai
0 – 2	20	14 – 16	13	28 – 30	6
2 – 4	19	16 – 18	12	30 – 32	5
4 – 6	18	18 – 20	11	32 – 34	4
6 – 8	17	20 – 22	10	34 – 36	3
8 – 10	16	22 – 24	9	36 – 38	2
10 – 12	15	24 – 26	8	38 – 40	1
12 – 14	14	26 – 28	7	> 40	0

1.4 lentelėje taškai suskirstyti, pagal failo padidėjimą po šifravimo. Tai apskaičiuojama patikrinant užšifruoto failo dydį ir iš jo atimant originalaus failo dydį [13].

1.3.1.1.5. Saugumas

Saugumas yra būtinas kriptografijos kriterijus. Saugumo ir patikimumo lygio visiškai nustatyti negalima. Daugiausia ką galima padaryti, tai patikrinti saugumą prieš labiausiai žinomas kriptografines atakas. Todėl šis kriterijus nėra vertinamas taškais [13].

1.3.1.2. Šifravimo metodų veikimo analizė

Visi testai atliekami naudojant „Raspberry Pi“ B modelį, kuriame įdiegta *OpenSSL* biblioteka. Kiekvieno algoritmo testavimui naudojamas 128 KB failas, kuris buvo šifruojamas ir dešifruojamas po 1000 kartų. Toliau pateikiami laiko, atminties ir energijos suvartojimo rezultatai reikalingi vienam šifravimo ir dešifravimo ciklo įvykdymui.

1.3.1.2.1. DES

Šifravimas trunka 141,968 ms, o dešifravimas trunka 92,567 ms. Bendras šifravimo ir dešifravimo laikas yra 234,535 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05

V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 6,580 µWh. Užšifruoto failo dydis padidėjo 24 B.

1.5 lentelė. DES rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	234,535 ms	21
Energijos suvartojimas	6,580 µWh	14
Disko vieta	24 B	9
Iš viso		44

1.3.1.2.2. 3DES

Šifravimas trunka 167,327 ms, o dešifravimas trunka 131,239 ms. Bendras šifravimo ir dešifravimo laikas yra 298,556 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05 V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 8,376 µWh. Užšifruoto failo dydis padidėjo 24 B.

1.6 lentelė. 3DES rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	298,556 ms	13
Energijos suvartojimas	8,376 µWh	10
Disko vieta	24 B	9
Iš viso		32

1.3.1.2.3. AES-256

Šifravimas trunka 131,517 ms, o dešifravimas trunka 85,703 ms. Bendras šifravimo ir dešifravimo laikas yra 217,220 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05 V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 6,094 µWh. Užšifruoto failo dydis padidėjo 32 B.

1.7 lentelė. AES-256 rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	217,220 ms	23
Energijos suvartojimas	6,094 µWh	15
Disko vieta	32 B	5
Iš viso		43

1.3.1.2.4. Blowfish

Šifravimas trunka 131,075 ms, o dešifravimas trunka 83,039 ms. Bendras šifravimo ir dešifravimo laikas yra 214,114 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05 V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 6,007 µWh. Užšifruoto failo dydis padidėjo 24 B.

1.8 lentelė. Blowfish rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	214,114 ms	24
Energijos suvartojimas	6,007 µWh	15
Disko vieta	24 B	9
Iš viso		48

1.3.1.2.5. CAST

Šifravimas trunka 123,927 ms, o dešifravimas trunka 82,240 ms. Bendras šifravimo ir dešifravimo laikas yra 206,167 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05 V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 5,784 µWh. Užšifruoto failo dydis padidėjo 24 B.

1.9 lentelė. CAST rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	206,167 ms	25
Energijos suvartojimas	5,784 µWh	16
Disko vieta	24 B	9
Iš viso		50

1.3.1.2.6. RC4

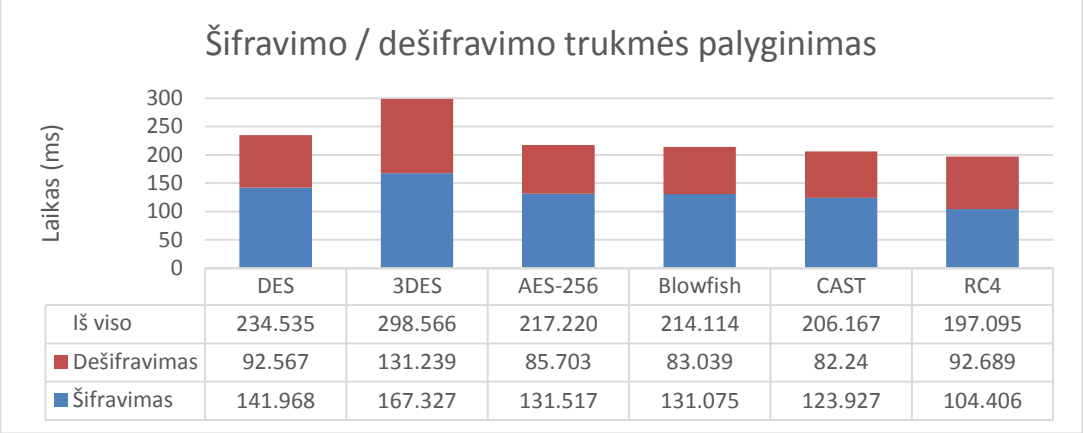
Šifravimas trunka 104,406 ms, o dešifravimas trunka 92,689 ms. Bendras šifravimo ir dešifravimo laikas yra 197,095 ms. Norint sužinoti energijos suvartojimą matuojama įtampa U = 5,05 V ir srovė I = 20 mA ir paskaičiuojama galia P = 101 mW. Energijos suvartojimas 5,530 µWh. Užšifruoto failo dydis padidėjo 16 B.

1.10 lentelė. RC4 rezultatų santrauka

Kriterijus	Matavimo vertė	Taškai
Trukmė	197,095 ms	26
Energijos suvartojimas	5,530 µWh	17
Disko vieta	16 B	13
Iš viso		56

1.3.1.3. Gautų rezultatų palyginimas

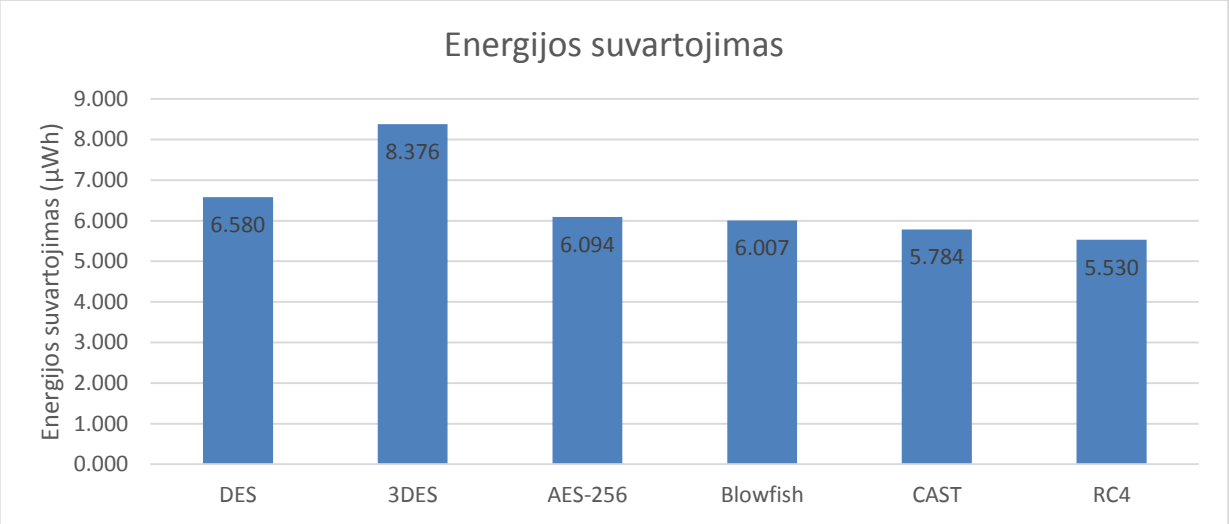
Užšifravimo ir dešifravimo trukmės palyginimas parodytas 1.1 paveiksle. Pasirodo, kad kiekvienam algoritmui dešifravimui reikia mažiau laiko nei šifravimui. Tiktai RC4 šifravimo ir dešifravimo trukmė beveik tokia pati. Įvykdytas palyginimas parodė, kad RC4 algoritmas (197,095 ms) yra greičiausias.



	DES	3DES	AES-256	Blowfish	CAST	RC4
Iš viso	234.535	298.566	217.220	214.114	206.167	197.095
Dešifravimas	92.567	131.239	85.703	83.039	82.24	92.689
Šifravimas	141.968	167.327	131.517	131.075	123.927	104.406

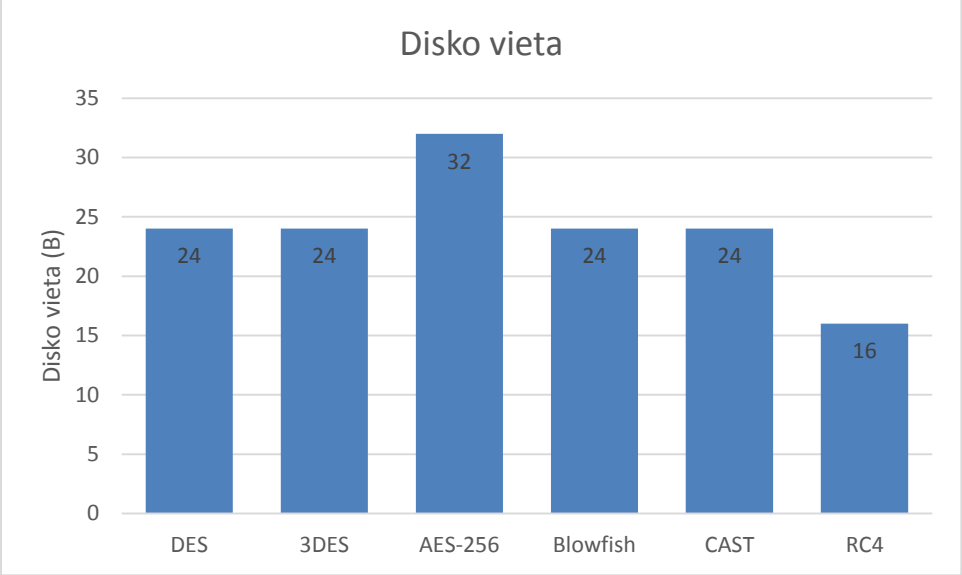
1.1 pav. Trukmės palyginimas šifruojant ir dešifruojant

Energijos suvartojimas pavaizduotas 1.2 paveiksle. Iš diagramos matyti, kad RC4 reikalingas mažiausias (5,530 µWh), o 3DES didžiausias (8,376 µWh) energijos kiekis.



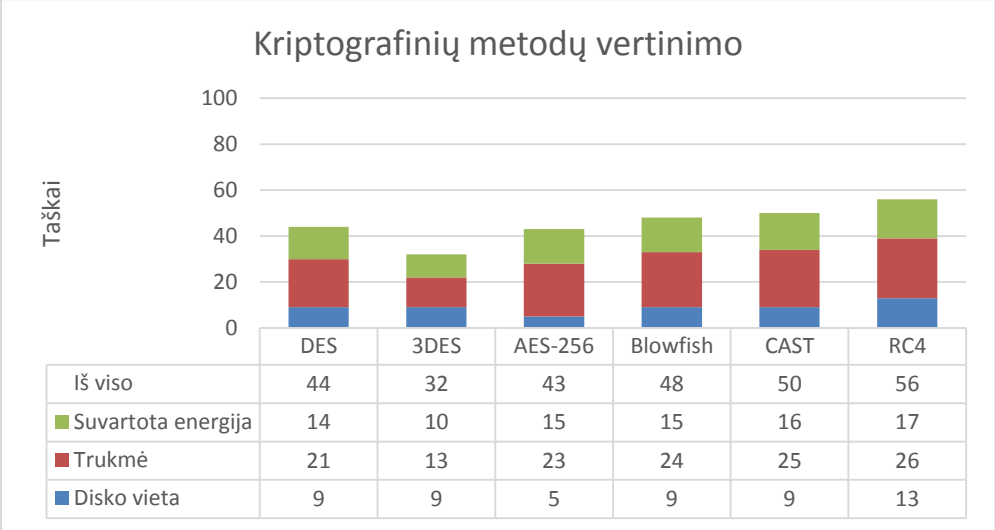
1.2 pav. Energijos suvartojimo diagrama

Disko vietos užimtumas pavaizduotas 1.3 paveiksle. Algoritmų DES, 3DES, Blowfish ir CAST disko vietos užimtumas vienodas. Šios diagramos rezultatas – AES-256 reikalauja daugiausia disko vietos, RC4 pasiekė geriausią rezultatą – 16 B.



1.3 pav. Reikalingos disko vietos diagrama

Visų rezultatų suvestinė atvaizduojama 1.4 paveiksle. Iš rezultatų matyti, kad galingiausias algoritmas pagal užsibrėžtus kriterijus yra RC4 (56 taškai). Antras pagal taškų skaičių yra CAST (50 taškų).



	DES	3DES	AES-256	Blowfish	CAST	RC4
Iš viso	44	32	43	48	50	56
Suvartota energija	14	10	15	15	16	17
Trukmė	21	13	23	24	25	26
Disko vieta	9	9	5	9	9	13

1.4 pav. Kriptografinių metodų vertinimo diagrama

1.3.1.4. Rezultatų apibendrinimas

Dėl to, kad šifravimo / dešifravimo algoritmas bus naudojamas įterptinėje sistemoje, buvo pasirinkta analizuoti tiktai simetrinius šifravimo algoritmus. Svarbiausi vertinimo kriterijai buvo teisingumas, patikimumas, energijos suvartojimas, šifravimo / dešifravimo trukmė ir atmintis. Iš rezultatų matyti, kad labiausiai tinkamas yra RC4 algoritmas, todėl jis ir bus naudojamas.

1.3.2. Asimetrinio šifravimo algoritmai

Asimetrinis šifravimo metodas dar kitaip vadinamas viešojo rakto metodu. Viešojo rakto metodas buvo sukurtas siekiant išvengti raktų perdavimo ir slaptumo problemos. Jame naudojama

dviejų matematiškai susijusių raktų pora: viešas ir privatus raktas. Tačiau jie sudaromi taip, kad iš viešo rakto nėra paprasta nustatyti privataus rakto. Dokumento autentiškumas užtikrinamas skaitmeniniu parašu, sukurtu naudojant privatų raktą, o jį patikrinti galima panaudojant viešą raktą. Taip pat galima užšifruoti tekstą naudojant viešą raktą – dešifravimui būtina turėti privatų raktą [14].

Labiausiai naudojami asimetrinio šifravimo algoritmai yra RSA, *Diffie-Hellman* ir DSA.

Viešojo rakto kriptografija bus naudojama tik pirmą kartą užsikraunant įterptinei sistemai, kitų raktų perdavimui. Tad šifravimo / dešifravimo greitis nėra svarbus. Todėl pasirenkamas RSA šifravimo metodas, nes jis jau yra įtrauktas į *OpenSSL* biblioteką.

1.3.3. Maišos funkcijos

Kriptografijoje maišos funkcijos priima tam tikrą duomenų kiekį, juos transformuoja ir grąžina fiksuoto dydžio simbolių eilutę. Tai yra vadinama maišos reikšme.

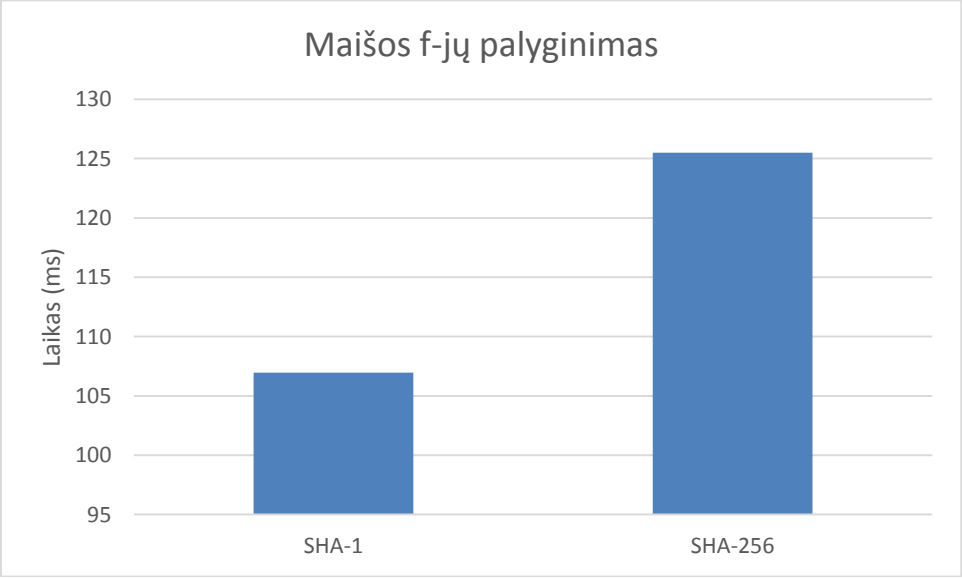
Maišos reikšmė yra tiksli didesnio duomenų kiekio ar dokumento reprezentacija, tai yra tam tikras skaitmeninis pirštų antspaudas. Kadangi šios maišos funkcijos yra vienakryptės, jos plačiai paplitę kriptografijoje. Taip pat jas galima taikyti duomenų integralumui ir korektiškumui tikrinti [15].

Maišos funkcijos bus naudojamos kontrolinių sumų apskaičiavimui. Šiai paskirčiai dažniausiai naudojamos SHA-1, SHA-256 maišos funkcijos. Maišos funkciją naudojant įterptinėje sistemoje svarbiausia greitis. Naudojamas 1 MB failas. Atliekami maišos funkcijos laiko T matavimai. Dėl atsitiktinių klaidų matavimai kartojami n = 1000 kartų.

Maišos funkcijos laiko vidurkis apskaičiuojamas naudojant formulę:

$$\overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \qquad (1.5)$$

Gauti rezultatai pateikiami 1.5 paveiksle.



1.5 pav. Maišos funkcijų vykdymo laiko palyginimo diagrama

Iš gautų rezultatų, pateiktų 1.5 paveiksle, matyti, kad SHA-1 veikia greičiau nei SHA-256, todėl jis bus naudojamas kontrolinių sumų skaičiavimui.

1.4. Išvados

Atlikus mikrovaldiklių architektūrų analizę buvo nustatyta galimos įsilaužimo vietos. Taip pat rasti būdai, padedantys nuo to apsisaugoti. Kadangi „Raspberry Pi“ įrenginyje kodas saugomas SD kortelėje, todėl reikalinga programinės įrangos apsauga ją užšifruojant. Tam buvo atlikta dažniausiai naudojamų simetrinių šifravimo algoritmų analizė. Sudaryta taškų sistema atsižvelgiant į algoritmo vykdymo laiką, suvartotą energiją ir failo padidėjimą po užšifravimo. Atlikus bandymus paaiškėjo, kad RC4 geriausiai atitinka iškeltus reikalavimus, todėl jis buvo pasirinktas programinio kodo šifravimui / dešifravimui.

Slaptas raktas bus laikomas centiniame serveryje, todėl jo apsikeitimui reikalingas asimetrinis kriptografijos algoritmas. Pasirinktas RSA kriptografijos algoritmas, todėl, kad jis įtrauktas į *OpenSSL* biblioteką ir nebereikia diegti papildomos programinės įrangos į įrenginį.

Autentifikavimui ir sistemos validavimui bus skaičiuojamos sistemos programinės įrangos komponentų kontrolinės sumos, todėl buvo atliktas maišos funkcijų palyginimas. Atsižvelgiant į vykdymo laiką, pasirinktas SHA-1 maišos funkcijos skaičiavimo algoritmas.

2. PROGRAMINĖS ĮRANGOS APSAUGOS METODOLOGIJA

Šiame skyriuje sudaroma programinės įrangos, esančios įterptinėje sistemoje, apsaugos metodologija. Atlikus analizę, aprašomos bendros įterptinės sistemos apsaugojimo taisyklės. Pateikiama metodologija skirta „Raspberry Pi“ įrenginyje esančios programinės įrangos apsaugai, bet pats apsaugojimo principas gali būti pritaikomas ir kitiems įrenginiams.

2.1. Pradinė konfigūracija

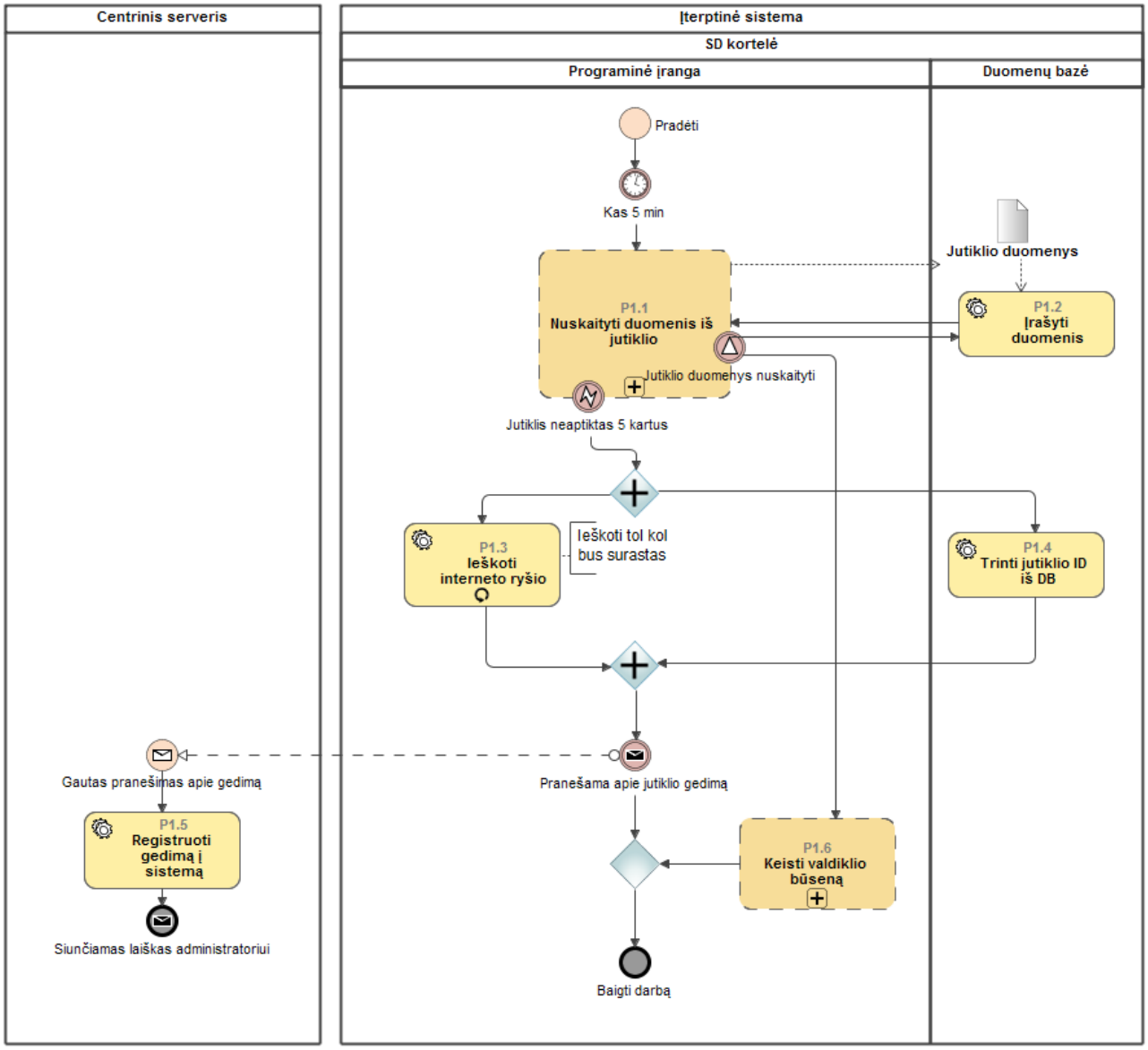
Svarbiausia tinkamai sukonfigūruoti įterptinėje sistemoje esančią operacinę sistemą [16].

Pagrindiniai dalykai, kurie turi būti atlikti konfigūruojant įrenginį:

1. Pakeisti numatytąjį slaptažodį;
2. Pakeisti numatytojo vartotojo vardą;
3. Išjungti visus nereikalingus procesus;
4. Įdiegti OS ir programinės įrangos atnaujinimus;
5. Įdiegti ugniasienę;
6. SSH prisijungimui naudoti autentifikavimo raktų porą;
7. Paruošti centrinį ir vidinį serverį:
    a. Įdiegti *web* serverį;
    b. Sukurti serveriui atskirą vartotoją su apribotomis teisėmis;
    c. Sukonfigūruoti serverį;
    d. Panaudoti SSL sertifikatą;
    e. Apsaugoti duomenų bazę.

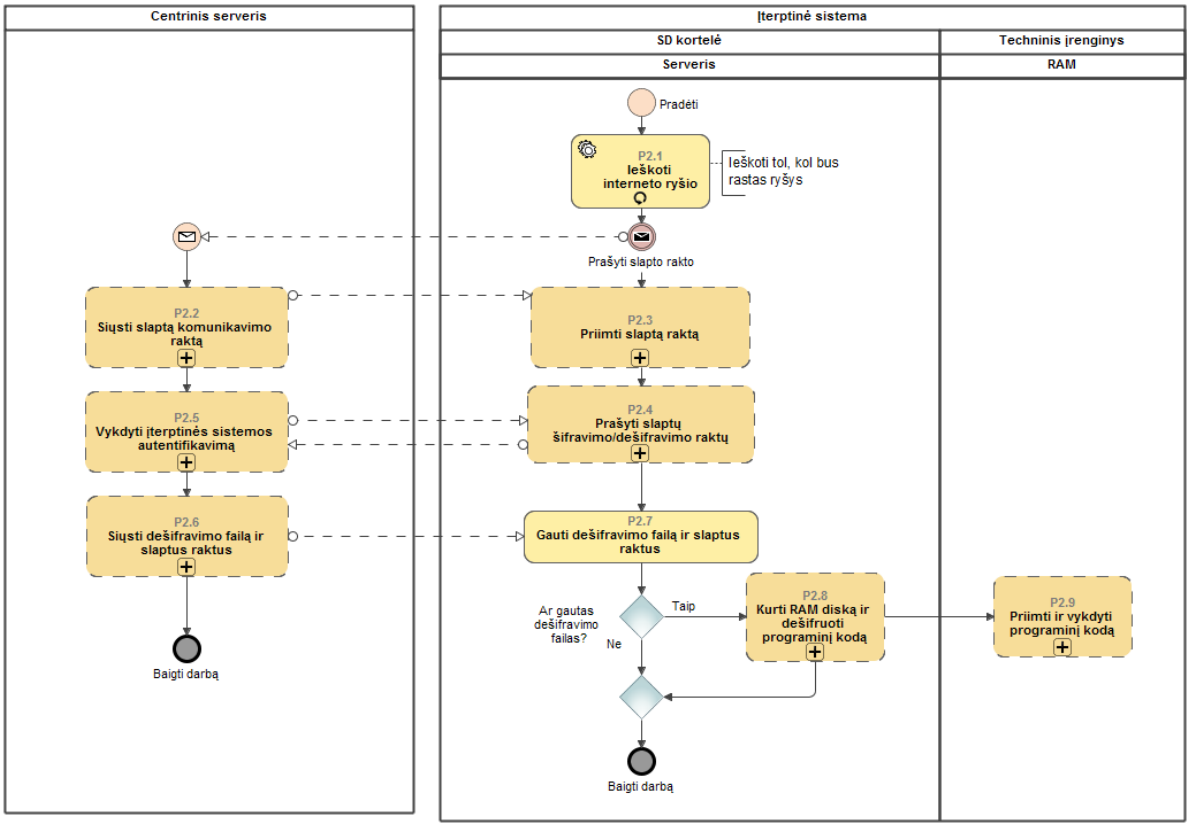
2.2. Programinės įrangos apsauga

Įterptinėse sistemose, kuriose nėra vidinės atminties, o tik SD kortelė, programinis kodas nėra apsaugotas. Bet kas, kas turi prieigą prie pačio įrenginio, gali nuskaityti ir pasisavinti SD kortelėje esančią programinę įrangą. 2.1 paveiksle vaizduojama standartinis įterptinės sistemos (bakalaurinio darbo metu darytos „Universiteto patalpų šildymo valdymo ir apskaitos sistemos“) valdymo procesas be programinės įrangos apsaugos.



2.1 pav. Standartinis programinės įrangos vykdymo procesas

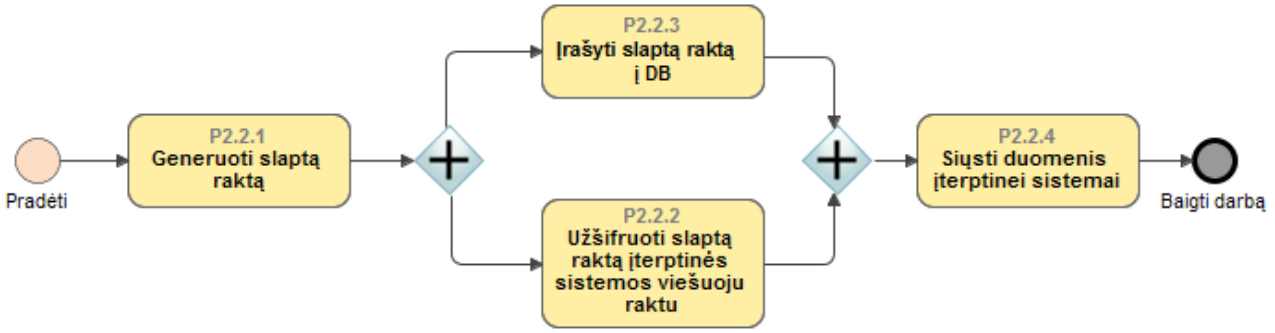
Siūlomas duomenų saugumo užtikrinimo procesas įterptinėje kompiuterinėje sistemoje pavaizduotas 2.2 pav.



2.2 pav. Įterptinių sistemų programinio kodo apsaugos veiklos proceso modelis

Užšifruotas programinis kodas patalpinamas SD kortelėje. Komunikacinis kanalas tarp įterptinės sistemos įrenginio ir centrinio serverio apsaugomas SSL protokolu. Kai įrenginys įjungiamas, tuomet vykdomas kreipimasis į centrinį serverį, prašant slapto rakto, skirto tolimesniam komunikavimui. Centrinis serveris sugeneruoja slaptą raktą ir jį užšifravęs įrenginio viešuoju raktu, siunčia raktą įrenginiui. Įrenginys, gavęs slaptą raktą, toliau siunčia ir gauna šifruotus duomenis naudodamas slaptą raktą. Įrenginys siunčia užklausą, kad gautų slaptus dešifravimo raktus, po to centrinis serveris pradeda įterptinės sistemos autentifikavimą ir validavimą. Autentifikavimui naudojamas rankos paspaudimo metodas. Jei aptikta, kad įrenginys yra nevalidus (pažeistas programinis kodas, aptikta sistemos pakeitimų ir t.t.), tuomet įrenginys yra užregistruojamas į blokuojamų sąrašą. Įvykus sėkmingam autentifikavimui ir validavimui, siunčiami duomenų, esančių DB, ir virtualaus RAM disko slapti raktai, bei programinio kodo dešifravimo failas. Sukuriamas virtualus RAM diskas su tiek vietos, kiek jos reikia programinei įrangai. Naudojant gautą raktą, diskas yra užšifruojamas. Programinis kodas dešifruojamas į RAM diską ir iš ten paleidžiamas. Slapti dešifravimų raktai RAM atmintyje laikomi tik tiek laiko, kiek reikalauja sistemos veikimas. Naudojant šį metodą ilgiau užtrunka tik programinės įrangos paleidimas įrenginio paleidimo metu, o tolimesnis veikimas beveik nesulėtėja.

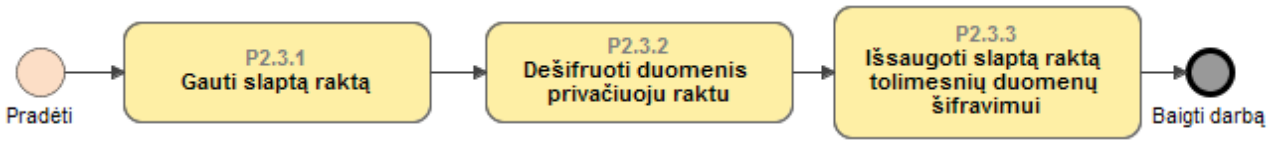
Išskleistas slapto komunikavimo rakto siuntimo subprocesas P2.2 pavaizduotas 2.3 paveiksle.



2.3 pav. Išskleistas „P2.2 Siųsti slaptą komunikavimo raktą“ subprocesas

Pirmiausia sugeneruojamas slaptas komunikavimo raktas. Jis įrašomas į duomenų bazę. Centriniame serveryje saugomas įterptinės sistemos įrenginio viešasis raktas, kuriuo užšifruojamas komunikavimo raktas. Užšifruotas komunikavimo raktas siunčiamas įterptinei sistemai. Kiekvienos naujos komunikacijos pradėjimo metu slaptas raktas keičiasi.

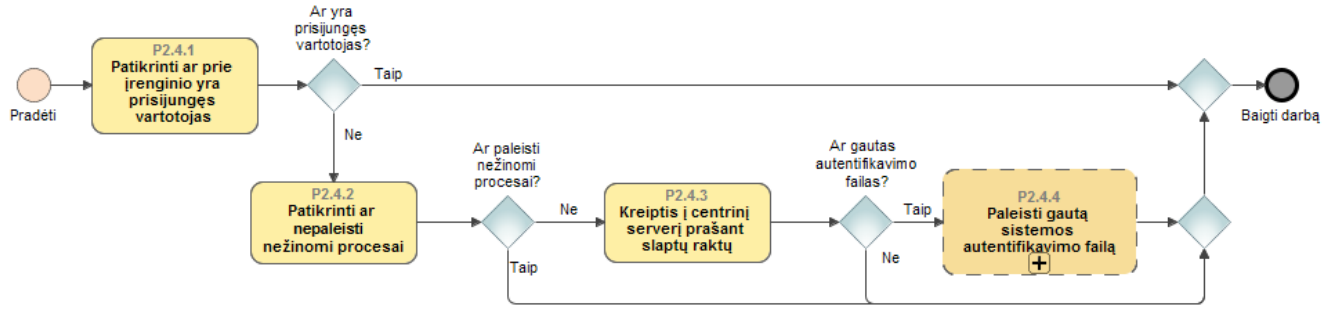
Slapto komunikavimo priėmimas pavaizduotas 2.4 paveiksle.



2.4 pav. Išskleistas „P2.3 Priimti slaptą raktą“ subprocesas

Kai gaunamas slaptas raktas, jis iššifruojamas panaudojant įrenginio privatųjį raktą. Iššifruotas raktas išsaugojamas, nes jis bus naudojamas tolimesnių perduodamų duomenų šifravimui / dešifravimui.

Gavus slaptą komunikavimo raktą, siunčiamas programinės įrangos dešifravimo ir kitų slaptų raktų prašymas, kuris pavaizduotas 2.5 paveiksle.

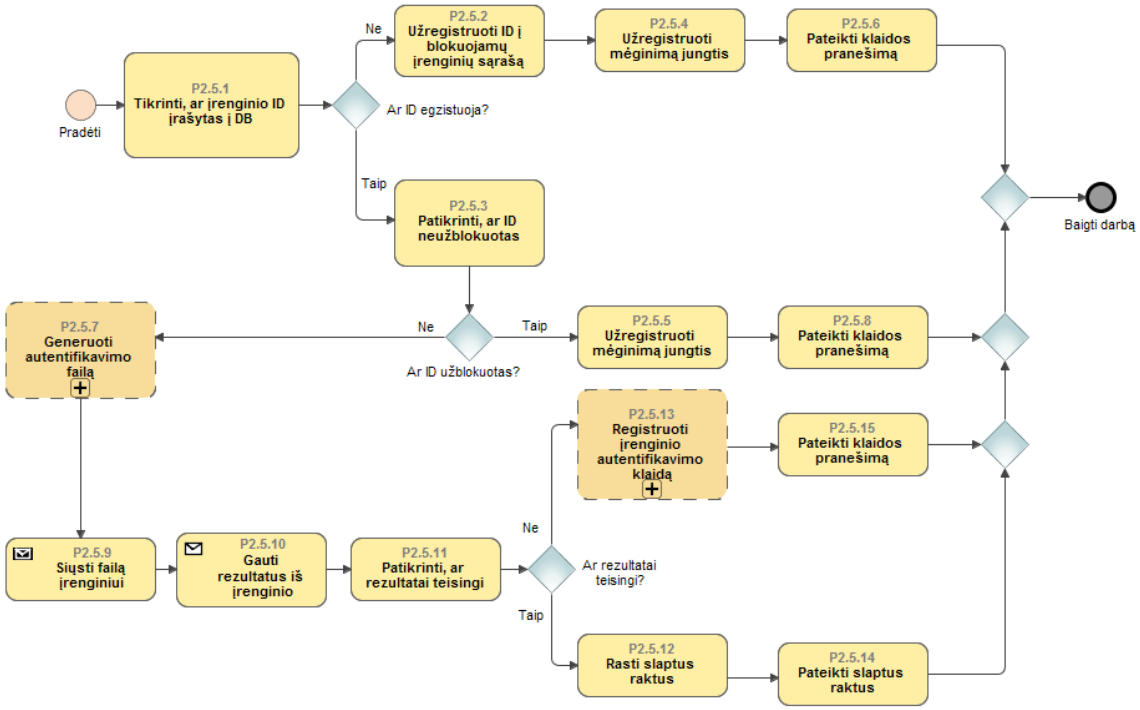


2.5 pav. Išskleistas „P2.4 Prašyti slaptų šifravimo / dešifravimo raktų“ subprocesas

Slaptų raktų prašymas nėra siunčiamas iš karto. Pirmiausia patikrinama, ar nėra prisijungusių vartotojų prie įrenginio (P2.4.1) ir ar nepaleisti nežinomi procesai (P2.4.2). Tik tada, jei abi sąlygos tenkinamos, kreipiamasi į centrinį serverį prašant slaptų raktų (P2.4.3). Centrinis serveris, patikrinęs, ar įrenginys registruotas sistemoje, atsiunčia sistemos autentifikavimo ir validavimo failą. Jei failas

gautas, jis paleidžiamas. Kitu atveju darbas baigiamas. Autentifikavimo ir validavimo failo vykdymo metu skaičiuojamos sistemos programinės įrangos komponentų kontrolinės sumos, patikrinama ar nėra prisijungusių vartotojų ir ar nepajungti neleistini procesai (ar nepaleista papildoma programinė įranga ir neprijungti neleistini įrenginiai). Į kontrolinių sumų skaičiavimą įtraukiama ir pačio failo kontrolinės sumos paskaičiavimas. Pačiame faile yra vienkartinė nuoroda, kuria siunčiami skaičiavimų rezultatai. Nuoroda galioja tiek laiko, kad įrenginys spėtų pateikti rezultatus. Negavus atsakymo per nustatytą laiką arba po atsakymo gavimo, nuoroda yra pašalinama.

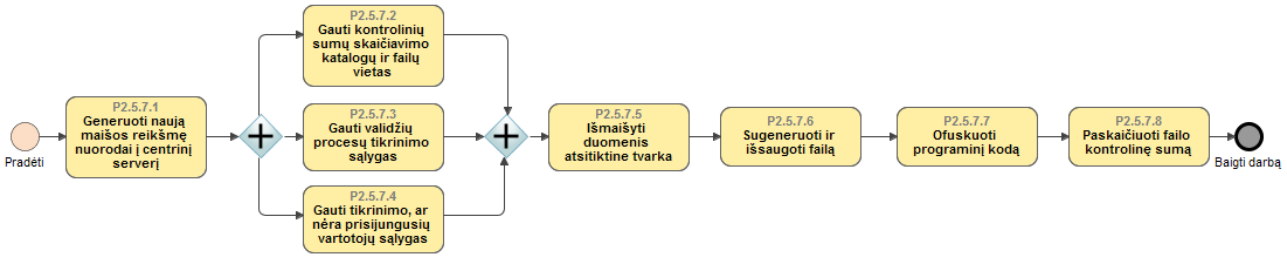
2.6 paveiksle pavaizduotas įterptinės sistemos autentifikavimas ir validavimas centrinio serverio pusėje.



2.6 pav. Išskleistas „P2.5 Vykdyti įterptinės sistemos autentifikavimą“ subprocesas

Autentifikuojant įrenginį pirmiausia patikrinama, ar užregistruotas įterptinės sistemos identifikacijos numeris. Jei neužregistruotas, numeris užregistruojamas į blokuojamų įrenginių sąrašą ir blokuojamas IP adresas. Taip pat užregistruojamas mėginimas jungtis ir grąžinamas klaidos pranešimas. Jei įrenginys registruotas, patikrinama ar jis nėra užblokuotas. Jei įrenginys užblokuotas, tuomet yra užregistruojamas bandymas jungtis ir pateikiamas klaidos pranešimas. Priešingu atveju, generuojamas autentifikavimo ir validavimo failas ir jis išsiunčiamas įrenginiui. Gauti rezultatai patikrinami su esamais duomenų bazėje. Jei rezultatai sutampa, tada įterptinei sistemai pateikiami slapti raktai. Kitu atveju užregistruojama autentifikavimo klaida ir pateikiamas klaidos pranešimas.

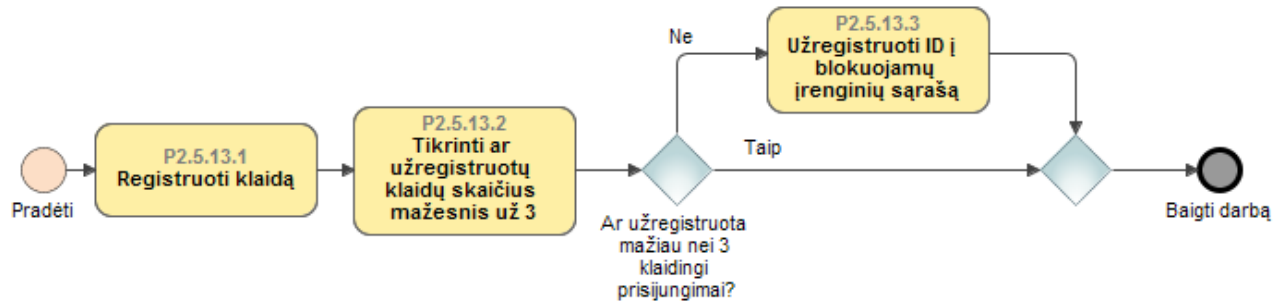
Autentifikavimo ir validavimo failo generavimas pavaizduotas 2.7 paveiksle.



2.7 pav. Išskleistas „P2.5.7 Generuoti autentifikavimo failą“ subprocesas

Pirmiausia sugeneruojama maišos funkcijos reikšmė, skirta vienkartinei nuorodai, į kurią įrenginys perduos skaičiavimo rezultatus. Prie maišos funkcijos reikšmės išsaugomas sugeneravimo laikas ir įrenginio identifikacijos numeris. Iš duomenų bazės paimami duomenys apie reikalingas kontrolinių sumų skaičiavimo katalogų ir failų vietas. Taip pat gaunamos procesų ir prisijungusių vartotojų tikrinimo sąlygos. Tuomet visi duomenys išmaišomi atsitiktine tvarka ir sugeneruojamas failas. Duomenys sumaišomi tam, kad kiekvieną kartą gražinamų duomenų eiliškumas būtų nenuspėjamas ir jų nebūtų galima suklastoti. Faile saugomas kodas užmaskuojamas. Jei įmanoma, jis paverčiamas į baitų kodą (angl. *byte code*). Pabaigoje paskaičiuojama failo kontrolinė suma.

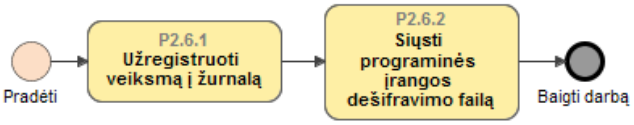
Įrenginio klaidos registravimo procesas pavaizduotas 2.8 paveiksle.



2.8 pav. Išskleistas „P2.5.13 Registruoti įrenginio registravimo klaidą“ subprocesas

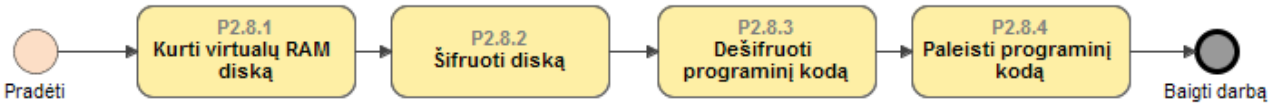
Pirmiausia į įvykių žurnalą yra užregistruojamas klaidingo autentifikavimo bandymas. Patikrinama, ar iš šio įrenginio buvo užregistruota klaidų anksčiau. Jei randama, kad užregistruota daugiau nei 2 klaidos, tuomet įrenginys įtraukiamas į blokuojamų įrenginių sąrašą. Kiekvieno sėkmingo autentifikavimo metu, ankstesni blogi mėginimai yra pašalinami iš įvykių žurnalo.

2.9 paveiksle pavaizduota slaptų raktų ir dešifravimo failo siuntimas. Pirmiausia veiksmas užregistruojamas į žurnalą ir tik tada siunčiamas programinės įrangos dešifravimo failas. Kartu su failu siunčiami RAM disko šifravimo ir duomenų bazėje esančių duomenų šifravimo slapti raktai.



2.9 pav. Išskleistas „P2.6 Siųsti dešifravimo failą ir slaptus raktus“ subprocesas

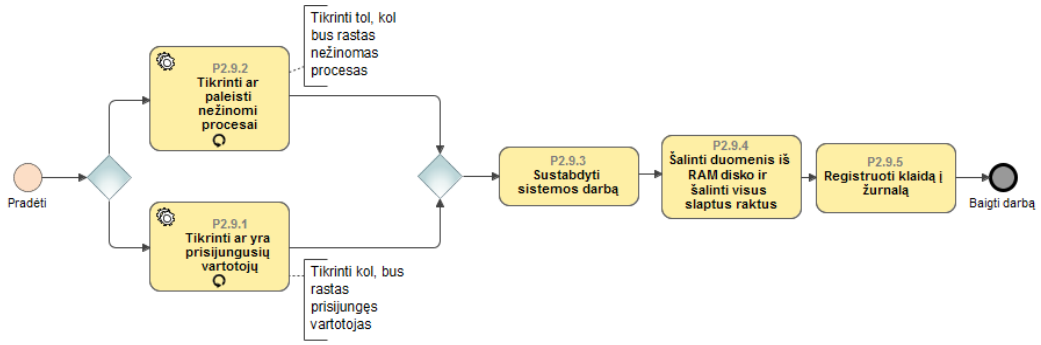
Gavus slaptus raktus, pradedamas kurti virtualus RAM diskas. Šis procesas pavaizduotas 2.10 paveiksle.



2.10 pav. Išskleistas „P2.8 Kurti RAM diską ir dešifruoti programinį kodą“ subprocesas

Pirmiausia sukuriamas virtualus RAM diskas. Tuomet, naudojant gautą slaptą disko šifravimo raktą, virtualus diskas yra užšifruojamas. Užšifravus diską, į jį dešifruojamas programinis kodas (apsaugota programinė įranga). Galiausiai paleidžiamas programinis kodas ir sistema pradeda darbą kaip pavaizduota 2.1 paveiksle. Skirtumas tik tas, kad duomenys duomenų bazėje šifruojami / dešifruojami gautu slaptu raktu, o seniau jie buvo saugomi nešifruoti.

Programos paleidimo metu paleidžiama ne tik apsaugota programinė įranga. Papildomai paleidžiama ir sistemos veikimo tikrinimo programinė įranga. Veikimo tikrinimo procesas pavaizduotas 2.11 paveiksle.

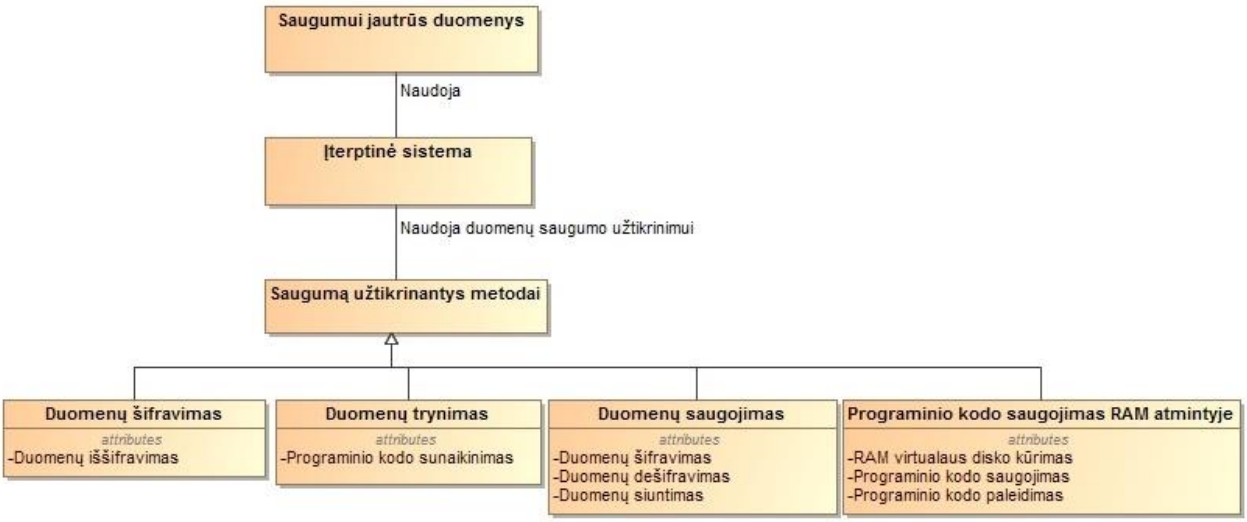


2.11 pav. Išskleistas „P2.9 Priimti ir vykdyti programinį kodą“ subprocesas

Sistema laukia, kol bus pastebėta vartotojo prisijungimas arba negalimo proceso paleidimas. Procesas gali būti programinė įranga arba prijungtas įrenginys. Pastebėjus vieną iš šių sąlygų, sustabdomas programinės įrangos vykdymas. Programinė įranga iš virtualaus RAM disko ir visi slapti raktai sunaikinami. Toje vietoje, kurioje stovėjo programinė įranga ir slapti raktai, kelis kartus perrašoma kita informacija, kad jos tikrai nebūtų galima atkurti. Pati klaida užregistruojama į įvykių žurnalą ir sistema baigia darbą.

2.2.1. Koncepcinis duomenų modelis

Programinė įranga, esanti įterptinėje sistemoje, yra užšifruota ir jos negalima įvykdyti be dešifravimo. Pagrindinis tikslas – duomenis dešifruoti taip, kad jų nebūtų galima pasisavinti.



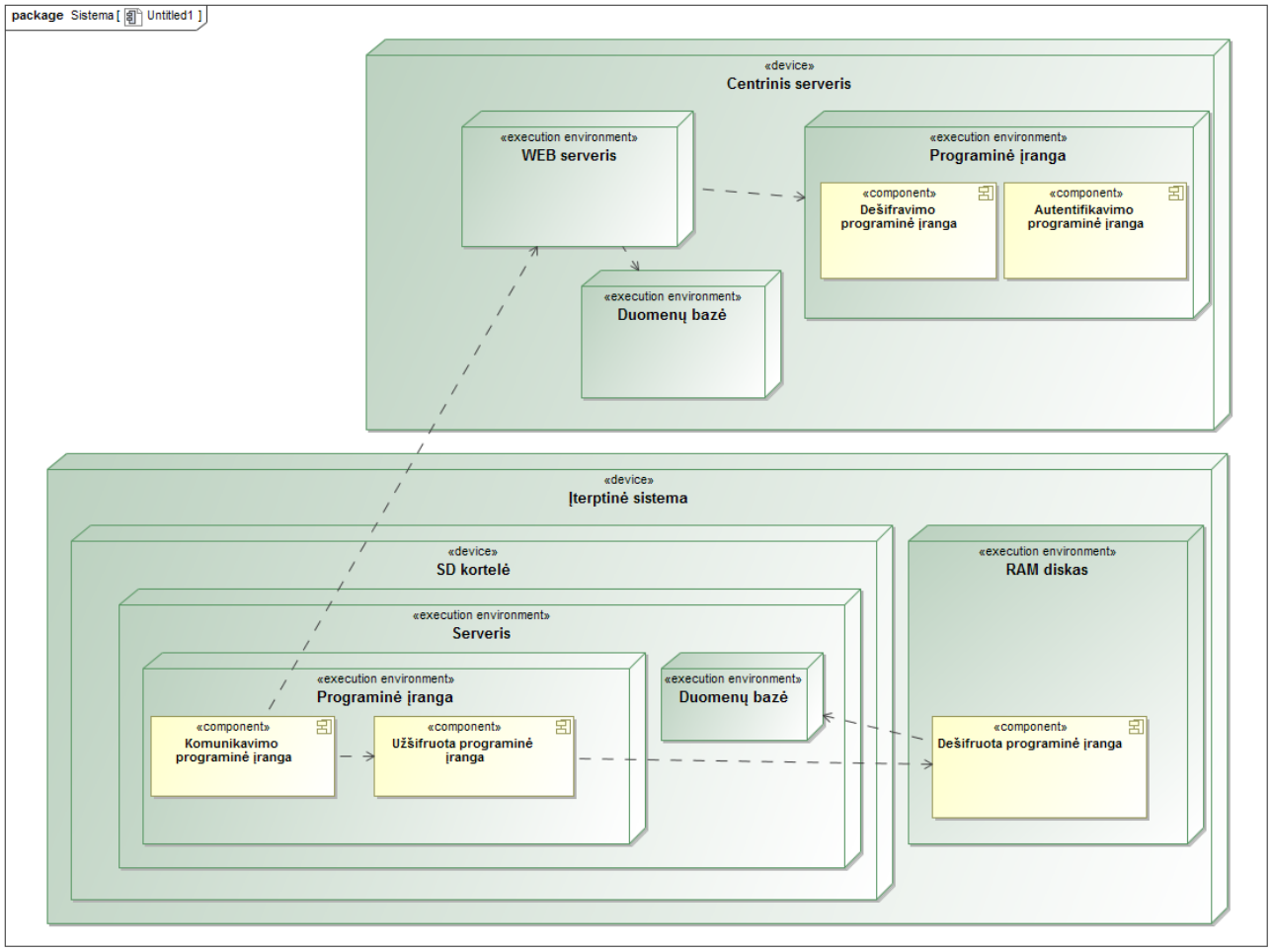
2.12 pav. Koncepcinis duomenų modelis

Paveikslėlyje 2.12 pavaizduotas duomenų modelis. Duomenys, esantys įterptinėje sistemoje, yra saugomi SD kortelėje. Šiuos duomenis naudoja įterptinė sistema tam, kad galėtų naudoti jutiklius ir valdiklius bei atlikti reikalingus skaičiavimus. Įterptinė sistema duomenų saugumo užtikrinimui naudoja tam skirtus metodus (virtualaus RAM disko sukūrimas, dešifruoti duomenys saugomi RAM atmintyje, pastebėjus nekorektišką veiklą, sistemos veikimo sustabdymas ir programinio kodo šalinimas). Taip atjungus įrenginio maitinimą arba ištraukus SD kortelę iš lizdo, ištrinamas dešifruotas programinis kodas iš RAM atminties. Galima iš kortelės pasiimti tik užšifruotus duomenis, kurie be dešifravimo rakto yra nenaudingi.

2.2.2. Įterptinių kompiuterinių sistemų programinio kodo apsaugos paketo diegimo modelis

Siūlomo sprendimo, realizuojančio duomenų saugumo užtikrinimą mobiliuose įrenginiuose, pagrindinės techninės dalys:

- Centrinis serveris;
- Įterptinė sistema;
- SD kortelė.



2.13 pav. Įterptinių kompiuterinių sistemų programinio kodo apsaugos paketo diegimo modelis

Visas sistemos diegimo modelis pavaizduotas 2.13 paveikslėlyje. Centrinis serveris naudojamas valdyti ir autentifikuoti įterptinėms sistemoms. Visa programos logika yra įdiegta įterptinėje sistemoje. Programinės įrangos, esančios įterptinėje sistemoje, pagrindiniai komponentai:

- Komunikavimo programinė įranga;
- Užšifruota programinė įranga;
- Iššifruotos programinės įrangos įkėlimo į RAM atmintį programa.

Programinės įrangos, esančios centiniame serveryje, pagrindiniai komponentai:

- Programinio kodo dešifravimo programinė įranga;
- Įrenginio autentifikavimo programinė įranga.

2.3. Išvados

Pagal atliktą analizę suskurta įtertinės sistemos programinės įrangos apsaugos metodologija. Kadangi negalima slaptų raktų saugoti įrenginyje arba SD kortelėje, naudojamas nuotolinis serveris, kuriame saugomi slapti raktai. Įrenginio autentifikavimui naudojamas rankos paspaudimo autentifikavimo metodas. Taip pat patikrinama ar nėra prijungtų papildomų įrenginių ir ar neprisijungę vartotojai per SSH prieigą. Įterptinė sistema gavus slaptus raktus programinį kodą dešifruoja į

užšifruotą virtualų RAM diską. Kadangi nepakanka apsaugoti programinio kodo tik sistemos paleidimo metu, sukuriama papildoma programinė įranga, kuri stebi sistemos darbą ir aptikus įsilaužimo galimybę, sustabdo sistemos darbą ir pašalina programinį kodą. Taip pat sudarytas įterptinių kompiuterinių sistemų programinio kodo apsaugos paketo diegimo modelis.

3. ĮTERPTINĖS SISTEMOS PROGRAMINIO KODO APSAUGOS PROTOTIPO REALIZACIJA

Šiame skyriuje atskirai aprašoma įterptinės sistemos įrenginio ir centrinio serverio apsaugos priemonės, bei naudojama programinė įranga. Taip pat aprašomas programinės įrangos veikimas.

3.1. Įterptinės sistemos įrenginio apsaugos priemonės ir programinės įrangos veikimas

3.1.1. Pradinė įtemptinės sistemos įrenginio konfigūracija

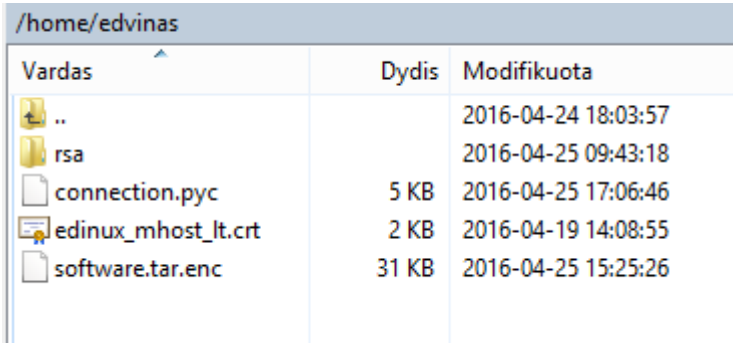
Pagrindiniai dalykai, kurie buvo atlikti konfigūruojant įrenginį:

1. Pakeistas numatytojo vartotojo vardas ir slaptažodis [17];
2. Išjungti visi nereikalingi procesai [17];
3. Įdiegti OS ir programinės įrangos atnaujinimai [17];
4. Įdiegta ugniasienė [17];
5. SSH prisijungimui naudojama autentifikavimo raktų pora [18];
6. Atlikti kiti pagrindiniai serverio konfigūravimo darbai.

3.1.2. Programinio kodo apsaugai naudojami failai

Visi failai, skirti programinės įrangos apsaugai, patalpinti „/home/edvinas“ vartotojo namų direktorijoje.

3.1 paveiksle pavaizduota pagrindinių failų, esančių įterptinės sistemos įrenginyje, struktūra.



3.1 pav. Įterptinės sistemos įrenginio failų struktūra

Direktorijoje „rsa“ saugomas privatus įrenginio raktas „private_device.pem“. Jis naudojamas slaptam komunikacijos rakto dešifravimui, kuris gaunamas iš centrinio serverio.

Failas „connection.pyc“ yra skirtas pradiniam ryšio su centriniu serveriu užmezgimui.

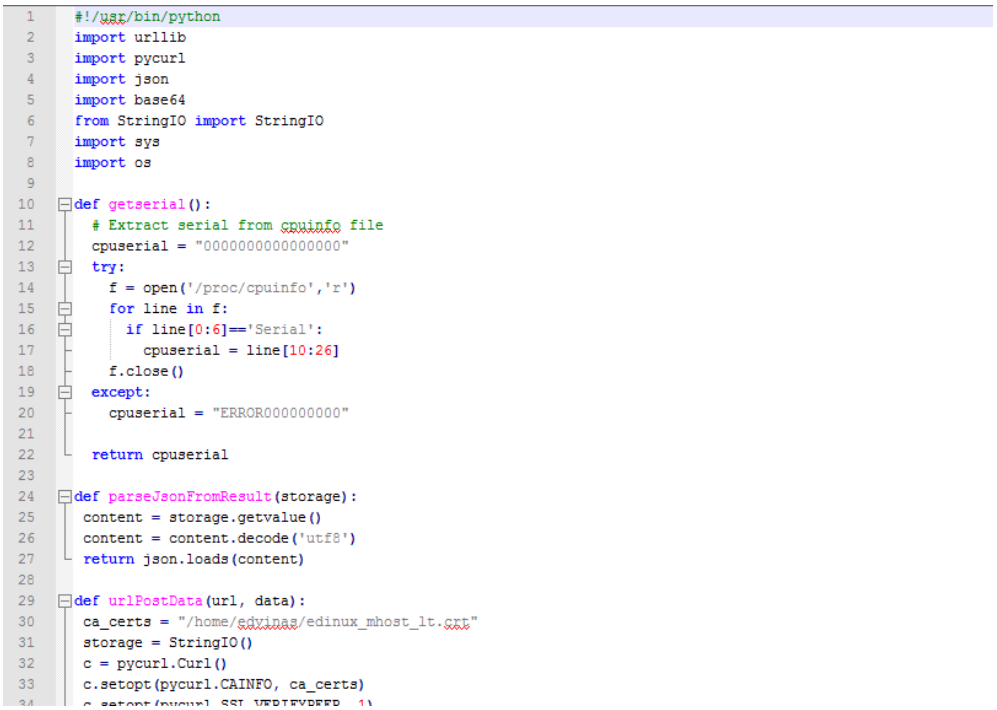
Centrinio serverio sertifikatas saugomas „edinux_mhost_lt.crt“ faile. Jis naudojamas centrinio serverio autentifikavimui.

Visas sistemos programinis kodas saugomas „software.tar.enc“ faile. Taip pat kartu su programiniu kodu užšifruotas ir „auth_run.pyc“ failas, skirtas sistemos veikimo validavimui viso veikimo metu.

3.1.3. Pradinio ryšio užmezgimo su centriniu serverio failo veikimas

Pradinio ryšio užmezgimo failas „connection.pyc“ parašytas „python“ programavimo kalba. Jis paleidžiamas, kai įterptinės sistemos įrenginys įjungiamas į elektros šaltinį. Failo paskirtis –kreiptis į centrinį serverį prašant slapto programinės įrangos dešifravimo rakto. Pačio failo programinis kodas yra maskuojamas ir paverčiamas į baitinį kodą.

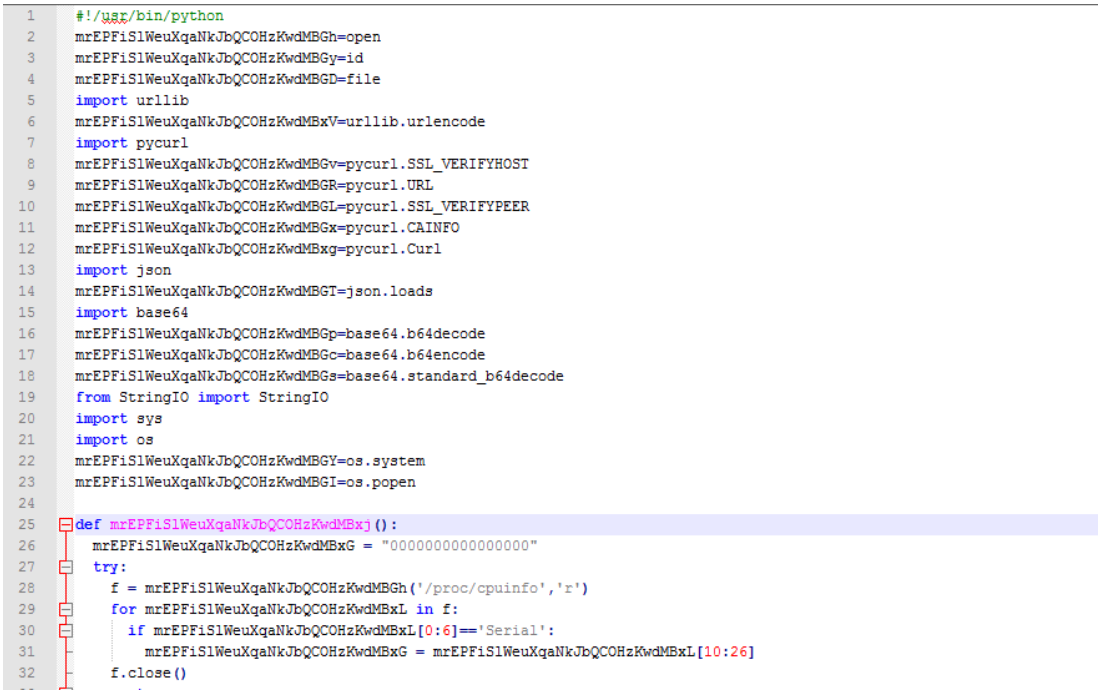
Neužmaskuotas programinis kodas pavaizduotas 3.2 paveiksle.



```
#!/usr/bin/python
import urllib
import pycurl
import json
import base64
from StringIO import StringIO
import sys
import os

def getserial():
  # Extract serial from cpuinfo file
  cpuserial = "0000000000000000"
  try:
    f = open('/proc/cpuinfo','r')
    for line in f:
      if line[0:6]=='Serial':
        cpuserial = line[10:26]
    f.close()
  except:
    cpuserial = "ERROR000000000"

  return cpuserial

def parseJsonFromResult(storage):
  content = storage.getvalue()
  content = content.decode('utf8')
  return json.loads(content)

def urlPostData(url, data):
  ca_certs = "/home/edvinas/edinux_mhost_lt.crt"
  storage = StringIO()
  c = pycurl.Curl()
  c.setopt(pycurl.CAINFO, ca_certs)
```

3.2 pav. Failo „connection.py“ programinio kodo pavyzdys

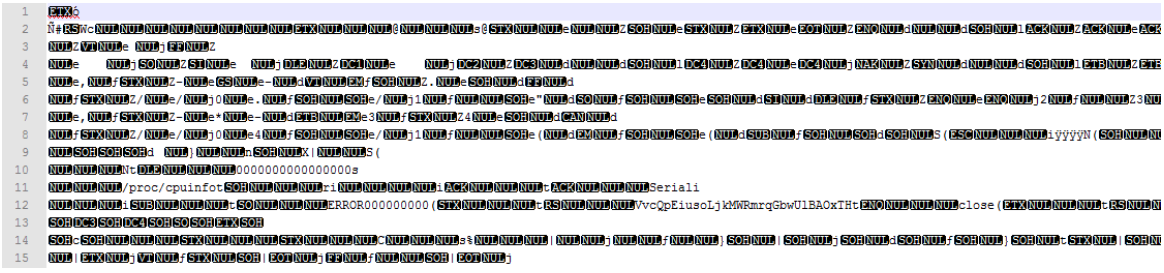
Iš 3.2 paveikslo matyti, kad programinis kodas įsilaužėliui lengvai suprantamas. Programos kintamųjų ir funkcijų pavadinimai atskleidžia, kas vykdoma programoje. Norint apsunkinti programos skaitomumą, programinis kodas užmaskuojamas. Užmaskuoto programinio kodo pavyzdys pateikiamas 3.3 paveiksle.



```
#!/usr/bin/python
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGh=open
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGy=id
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGD=file
import urllib
mrEPFiSlWeuXqaNkJbQCOHzKwdMBxV=urllib.urlencode
import pycurl
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGv=pycurl.SSL_VERIFYHOST
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGR=pycurl.URL
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGL=pycurl.SSL_VERIFYPEER
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGx=pycurl.CAINFO
mrEPFiSlWeuXqaNkJbQCOHzKwdMBxg=pycurl.Curl
import json
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGT=json.loads
import base64
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGp=base64.b64decode
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGc=base64.b64encode
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGs=base64.standard_b64decode
from StringIO import StringIO
import sys
import os
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGY=os.system
mrEPFiSlWeuXqaNkJbQCOHzKwdMBGI=os.popen

def mrEPFiSlWeuXqaNkJbQCOHzKwdMBxj():
  mrEPFiSlWeuXqaNkJbQCOHzKwdMBxG = "0000000000000000"
  try:
    f = mrEPFiSlWeuXqaNkJbQCOHzKwdMBGh('/proc/cpuinfo','r')
    for mrEPFiSlWeuXqaNkJbQCOHzKwdMBxL in f:
      if mrEPFiSlWeuXqaNkJbQCOHzKwdMBxL[0:6]=='Serial':
        mrEPFiSlWeuXqaNkJbQCOHzKwdMBxG = mrEPFiSlWeuXqaNkJbQCOHzKwdMBxL[10:26]
    f.close()
```

3.3 pav. Užmaskuoto „connection.py“ programinio kodo pavyzdys

Iš 3.3 paveikslo matyti, kad kodo skaitomumas pasidarė sudėtingesnis. Programinio kodo užmaskavimui naudojama „pyminifier“ biblioteka [19]. Užmaskuojant kodą, pakeičiami kintamųjų pavadinimai į atsitiktinius simbolius. Taip pat sumaišoma programinio kodo struktūra. Panaudojus užmaskavimą, programos skaitomumas pasidarė sudėtingesnis, bet vis dar suprantamas. Norint dar labiau apsunkinti programinio kodo skaitomumą jis paverčiamas į baitinį kodą. Baitinio kodo pavyzdys pateiktas 3.4 paveiksle.

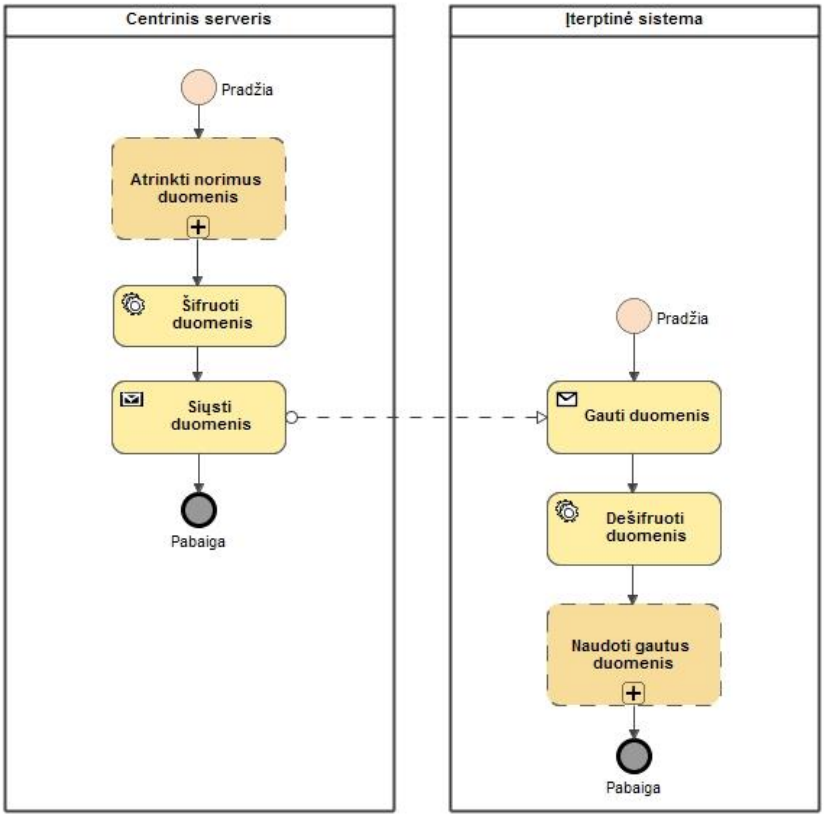


3.4 pav. Failo „connection.pyc“ baitinio kodo pavyzdys

Kaip matyti 3.4 paveiksle programinio kodo suprasti praktiškai neįmanoma. Matosi tik kelios programinio kodo nuotrupos, o pagrindinis kodas atvaizduojamas kaip neaiškūs simboliai. Toliau aprašomas programinio kodo veikimas.

Pirmiausia kreipiamasi į centrinį serverį perduodant unikalų įrenginio numerį (kreipimosi metu patikrinamas centrinio serverio SSL sertifikatas). Jei įrenginio numeris yra registruotas, tuomet gaunamas slaptas komunikavimo raktas užšifruotas įrenginio viešuoju raktu. Kitu atveju centrinis serveris užblokuoja nežinomą įrenginį. Gavus slaptą komunikavimo raktą, jis dekoduojamas naudojant slaptą įrenginio raktą ir naudojamas tolimesniam duomenų apsikeitimui. Duomenų apsikeitimas,

naudojant slaptą komunikavimo raktą, pavaizduotas 3.5 paveiksle. Duomenų apsikeitimui naudojamas RC4 simetrinis šifravimo algoritmas.



3.5 pav. Duomenų apsikeitimo naudojant slaptą raktą diagrama

Gavus slaptą komunikavimo raktą, kreipiamasi į centrinį serverį prašant slapto programinės įrangos dešifravimo rakto. Centrinis serveris sugeneruoja autentifikavimo failą (apie jo veikimą plačiau aprašyta 3.2.1.2 poskyryje) ir siunčia įrenginiui. Įrenginyje paleidžiamas autentifikavimo failas. Gauti rezultatai siunčiami į centrinį serverį ir patikrinama, ar tai būtent tas įrenginys. Taip pat patikrinama, ar įrenginyje nėra paleistų neleistinų procesų. Sėkmingo autentifikavimo metu gaunamas programinės įrangos dešifravimo failas bei duomenų bazės ir virtualaus RAM disko slapti raktai.

Pirmiausia sukuriamas virtualus RAM diskas. Sukuriamas aplankas, kuris bus naudojamas kaip RAM disko prijungimo vieta:

```
sudo mkdir /mnt/ramdisk
```

RAM disko sukūrimui naudojama mount komanda:

```
mount -t [TYPE] -o size=[SIZE] [FSTYPE] [MOUNTPOINT]
```

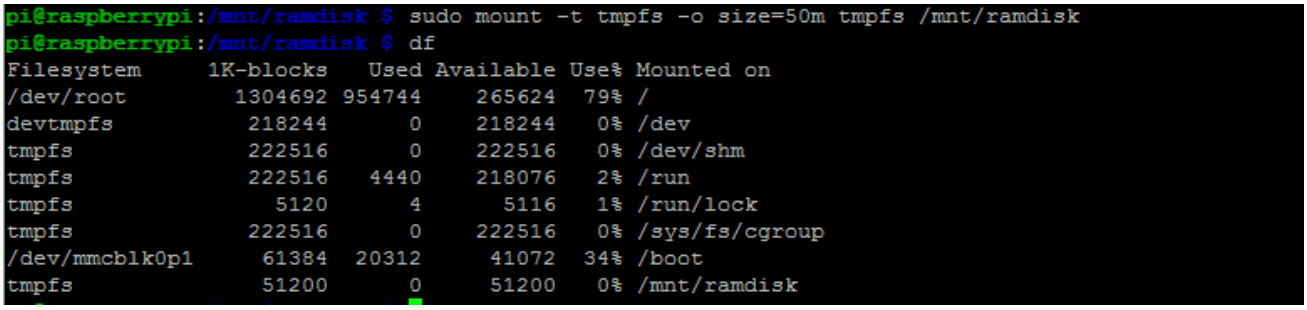
Pakeičiamos atributų reikšmės norimomis vertėmis:

- [TYPE] – RAM disko naudojamas tipas; **tmpfs** arba **ramfs**;
- [SIZE] – failų sistemos naudojamas dydis;
- [FSTYPE] – RAM disko naudojamas tipas: **tmpfs**, **ramfs**, **ext4**, ir t.t.;
- [MOUNTPOINT] – disko vieta.

Programiniai įrangai pakaks 2MB vietos:

```
sudo mount -t tmpfs -o size=2m tmpfs /mnt/ramdisk
```
[20]

Virtualus RAM diskas pavaizduotas 3.6 paveiksle.



3.6 pav. Virtualus RAM diskas

Sukūrus virtualų RAM diską, jis užšifruojamas naudojant „DM-Crypt LUKS“ disko užšifravimo algoritmą [21]. Užšifravus diską, paleidžiamas programinės įrangos dešifravimo failas. Programinė įranga dešifruojama tiesiai į virtualų RAM diską. Dešifruota programinė įranga paleidžiama ir sistema pradeda darbą.

3.1.4. Įterptinės sistemos veikimo stebėjimo failo veikimas

Nepakanka apsaugoti programinę įrangą tik įrenginio paleidimo metu. Bet kuriuo sistemos veikimo metu galima mėginti į jį išsilaužti. Tam kartu su programine įranga lygiagrečiai veikia „auth_run.pyc“ faile esanti programinė įranga. Šios programinės įrangos tikslas aptikti naujai atsiradusius procesus, prisijungusius vartotojus ir prijungtus įrenginius. Jei atsirado prisijungęs vartotojas, naujas negalimas procesas arba įrenginys, tada sustabdomas programos veikimas. Sustabdžius sistemos veikimą, sunaikinama programinė įranga, tai daroma naudojant komandą „shred” [22].

3.1.5. Duomenų šifravimas / dešifravimas

Duomenų šifravimui / dešifravimui naudojama *OpenSSL* biblioteka, kurioje yra reikalingi kriptografijos algoritmai (RC4 ir RSA).

3.1.5.1. RSA kriptografijos naudojimas

3.1.5.1.1. RSA raktų kūrimas

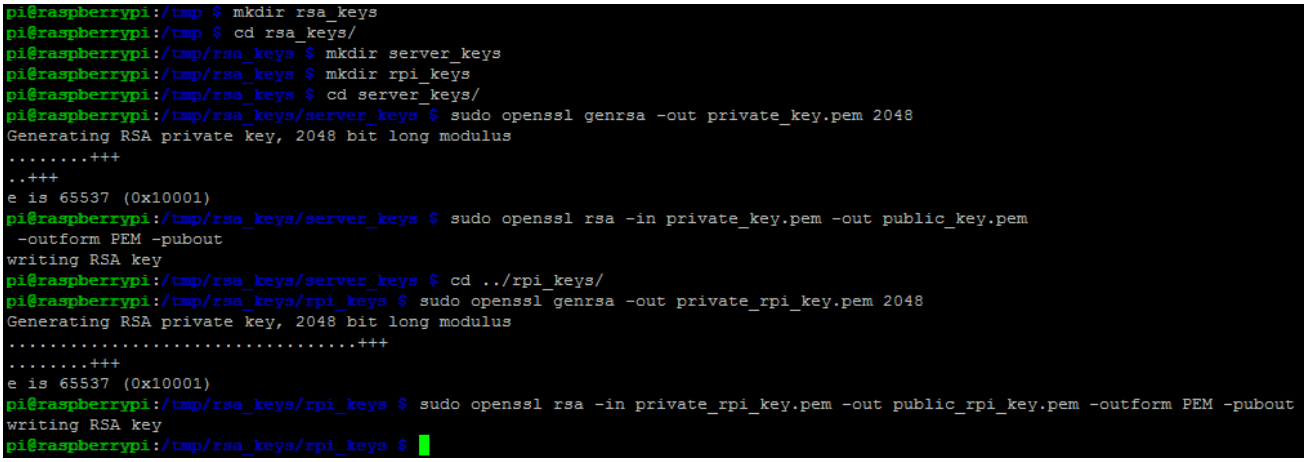
Pirmiausia sukuriama RSA privataus ir viešojo rakto pora.

Privačiojo rakto kūrimas (2048 bitai):

```
sudo openssl genrsa -out private_key.pem 2048
```

Viešojo rakto kūrimas:

```
sudo openssl rsa -in private_key.pem -out public_key.pem -outform PEM -pubout
```



3.7 pav. Sukurti privatūs ir viešieji RSA raktai

3.1.5.1.2. RSA failo užšifravimas

Failo užšifravimas naudojant RSA viešąjį raktą:

```
sudo openssl rsautl -encrypt -inkey public_key.pem -pubin -in encrypt.txt -out encrypt.dat
```

encrypt.txt – failas, kurį norima užšifruoti.

encrypt.dat – užšifruotas failas.

3.1.5.1.3. RSA failų dešifravimas

Failo dešifravimui naudojamas RSA privatusis raktas:

```
sudo openssl rsautl -decrypt -inkey private_key.pem -in encrypt.dat -out new_encrypt.txt
```

encrypt.dat – užšifruotas failas.

new_encrypt.txt – dešifruotas failas.

3.1.5.2. RC4 kriptografijos naudojimas

3.1.5.2.1. RC4 failo užšifravimas

Failo užšifravimas naudojant RC4:

```
sudo openssl rc4 -in encrypt.txt -out encrypt.enc
```

encrypt.txt – failas, kurį norima užšifruoti.

encrypt.enc – užšifruotas failas.

3.1.5.2.2. RC4 failo dekodavimas

Failo dekodavimas:

```
sudo openssl rc4 -d -in encrypt.enc -out new_encrypt.txt
```

encrypt.enc – užšifruotas failas.

new_encrypt.txt – dešifruotas failas.

3.1.6. Duomenų bazės apsauga

Duomenys, esantys duomenų bazėje, užšifruojami RC4 raktu, gautu iš centrinio serverio. Jis perduodamas kartu su programinės įrangos dešifravimo raktu. Tad visi surinkti duomenys, kurie stovi

duomenų bazėje, yra saugūs ir juos galima dešifruoti tik turint RC4 slaptą raktą. Taip pat duomenų bazės apsaugai panaudoti metodai aprašyti 7.1 priede.

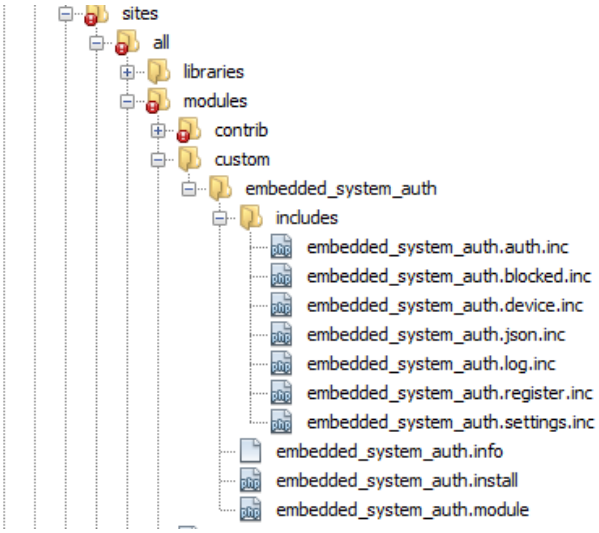
3.2. Centrinio serverio apsauga ir programinės įrangos veikimas

Serveris ir duomenų bazė sukonfigūruojami laikantis tų pačių principų, kaip ir konfigūruojant įterptinės sistemos įrenginio serverį. Todėl apie tai šiame poskyryje nerašoma. Šis poskyris yra skirtas aprašyti programinio kodo apsaugos metodologijos realizavimui centrinio serverio dalyje.

3.2.1. Įterptinės sistemos autentifikavimo programinė įranga

Centriniame serveryje pasirinkta naudoti „Drupal 7“ turinio valdymo platformą. Prototipe naudojama 1 lygio architektūra, bet realioje sistemoje patariama naudoti 3 lygių architektūrą dėl didesnio saugumo. Tikėtina, kad įsilaužėliai mėgins išgauti slaptus raktus iš centrinio serverio, o ne iš įterptinės sistemos įrenginio. Todėl centrinis serveris turi būti taip pat tinkamai apsaugotas.

Sukurtas „Drupal 7“ platformai skirtas modulis „embedded_system_auth“, skirtas įrenginio autentifikavimui ir slaptų raktų perdavimui. Sukurto modulio failų struktūra pavaizduota 3.8 paveiksle.



3.8 pav. Modulio „embedded_system_auth“ struktūra

Failų paskirtis:

- Faile „embedded_system_auth.install“ aprašyta duomenų bazės schema;
- Faile „embedded_system_auth.module“ aprašytos bendrai naudojamos funkcijos bei meniu nuorodos;
- Failas „embedded_system_auth.auth.inc“ skirtas sistemos autentifikavimui ir validavimui;
- Failas „embedded_system_auth.blocked.inc“ skirtas įrenginių ir IP adresų blokavimui;
- Failas „embedded_system_auth.device.inc“ skirtas įrenginio duomenų įvedimui ir redagavimui;
- Failas „embedded_system_auth.json.inc“ skirtas priimti / grąžinti duomenis iš įrenginiui;

- Failas „embedded_system_auth.log.inc“ skirtas registruoti įvykių žurnalą;
- Failas „embedded_system_auth.register.inc“ skirtas priimti duomenis iš įrenginio ir automatiškai užregistruoti leidžiamus procesus, vartotojus, bei įrenginius;
- Failas „embedded_system_auth.settings.inc“ skirtas sistemos globaliems nustatymams.

Plačiau apie centrinės sistemos veikimą aprašoma tolimesniuose poskyriuose.

3.2.1.1. Duomenų bazės struktūra ir saugomi duomenys

Duomenų bazė sukonfigūruojama laikantis tos pačios metodikos, kaip ir konfigūruojant įrenginio duomenų bazę.

Sukurto modulio paleidimui reikalingas „aes“ modulis [23]. Jis skirtas užšifruoti duomenis, esančius duomenų bazėje. Modulyje sukurtos duomenų bazės lentelės pavaizduotos 3.9 paveiksle.

dpl_embedded_system_auth_checksum	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	2	InnoDB	utf8_general_ci	16 KiB	
dpl_embedded_system_auth_device	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	1	InnoDB	utf8_general_ci	32 KiB	
dpl_embedded_system_auth_device_blocked	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	0	InnoDB	utf8_general_ci	32 KiB	
dpl_embedded_system_auth_ip_blocked	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	1	InnoDB	utf8_general_ci	32 KiB	
dpl_embedded_system_auth_log	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	212	InnoDB	utf8_general_ci	48 KiB	
dpl_embedded_system_auth_process	Peržiūrėti Struktūra Paieška Įterpti Išvalyti Šalinti	72	InnoDB	utf8_general_ci	16 KiB	

3.9 pav. Modulio sukurtos duomenų bazės lentelės

Lentelėje „dpl_embedded_system_auth_device“ saugoma informacija apie įterptinės sistemos įrenginį. Joje yra informacijas apie įterptinės sistemos įrenginio unikalų numerį, viešąjį įrenginio raktą bei vienkartinį nuorodos numerį, skirtą autentifikavimo duomenų grąžinimui. Taip pat joje saugomi programinės įrangos šifravimo, duomenų bazės duomenų šifravimo, komunikavimo slaptieji raktai.

Lentelėje „dpl_embedded_system_auth_checksum“ saugomos įrenginio komponentų kontrolinės sumos.

Lentelėje „dpl_embedded_system_auth_device_blocked“ saugomas užblokuotų įrenginių sąrašas.

Lentelėje „dpl_embedded_system_auth_ip_blocked“ saugomas užblokuotų IP adresų sąrašas.

Lentelėje „dpl_embedded_system_auth_log“ saugomi visi sistemos įvykiai.

Lentelėje „dpl_embedded_system_auth_log“ saugomi leistinų paleistų procesų, prisijungusių vartotojų ir prijungtų įrenginių sąrašai.

Visi saugumo reikalaujantys duomenys duomenų bazėje yra užšifruojami AES simetrinio rakto kriptografijos algoritmo slaptuoju raktu.

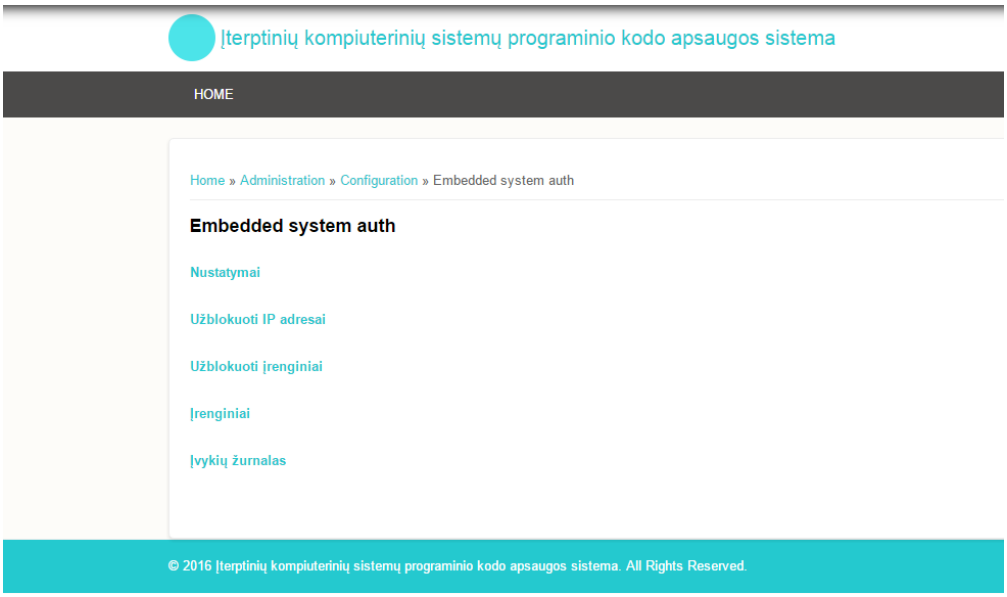
software_secret_key	database_secret_key	communication_secret_key	public_device_key
w0DDy4SuGjkPqvce7g3VCUVah8605iq/JQyJCzzPrfzpA5tgdl...	w0DDy4SuGjkPqvce7g3VCUVah8605iq/JQyJCzzPrfzpA5tgdl...	7EXBJnt234kTWrOeQH/TNQ==	4ul7YlftDQIPaOdvcveD
w0DDy4SuGjkPqvce7g3VCUVah8605iq/JQyJCzzPrfzpA5tgdl...	mnqf0psKaWK1Arw3a4HlaFDK5EGyPoRlqsPA7RnAGleGTXfXDP...		4ul7YlftDQIPaOdvcveD

3.10 pav. Duomenų bazėje esančių duomenų pavyzdys

Kaip matyti 3.10 paveiksle, duomenys duomenų bazėje užšifruoti ir jų be slapto rakto perskaityti negalima.

3.2.1.2. Įterptinių kompiuterinių sistemų programinio kodo apsaugos sistemos vartotojo sąsaja

3.11 paveiksle pateikta pagrindiniai vartotojo sąsajos meniu punktai.



3.11 pav. Pagrindiniai „embedded_system_auth“ modulio vartotojo sąsajos meniu punktai

Per vartotojo sąsają galima prieiti prie sistemos nustatymų, užblokuotų IP adresų ir įrenginių, įrenginių sąrašo bei įvykių žurnalo.

Įrenginių sąrašo atvaizdavimas pavaizduotas 3.12 paveiksle.

Home » Administration » Configuration » Embedded system auth » Įrenginiai

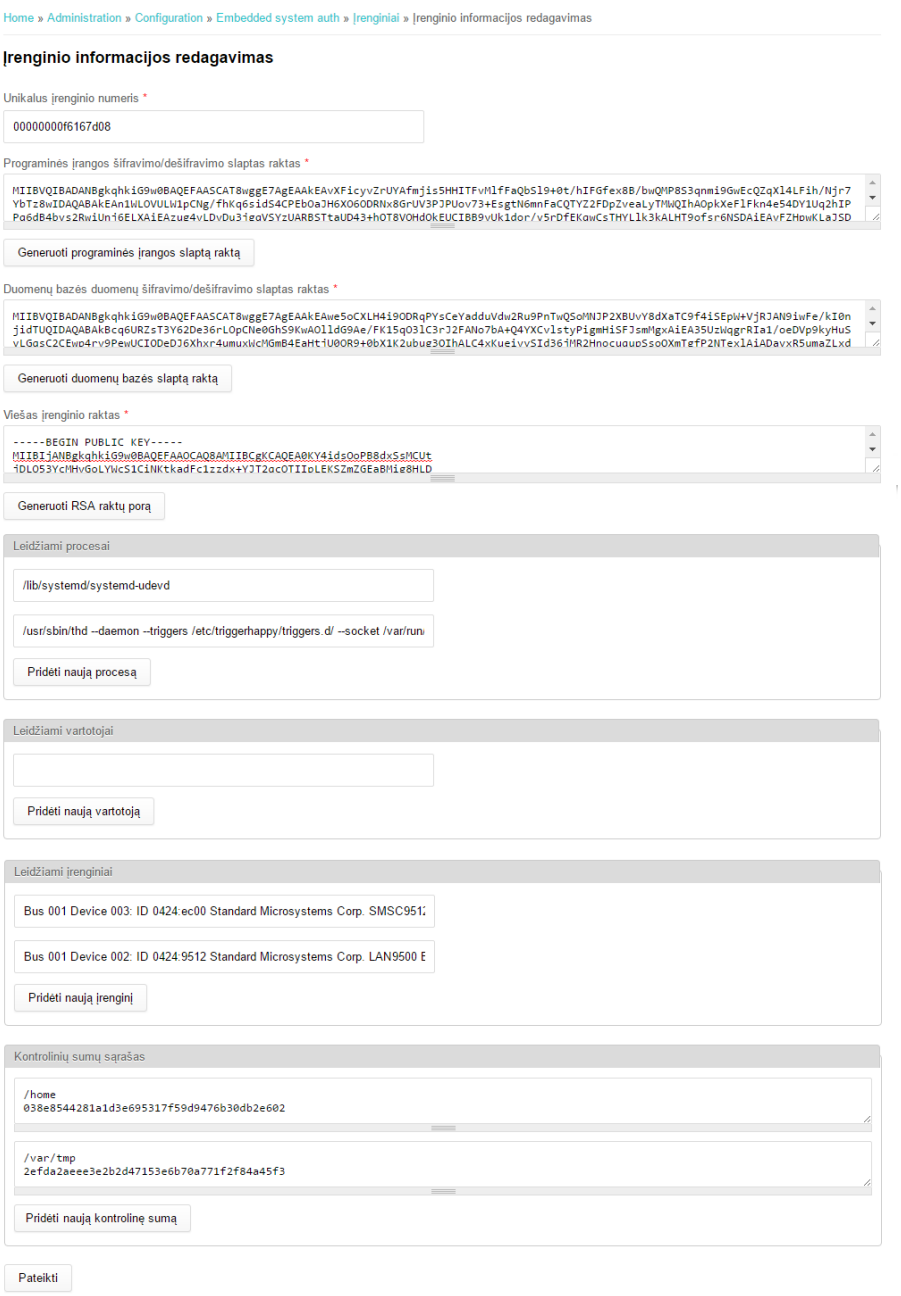
Įrenginiai

Pridėti naują

ID	Įrenginio ID	Būsena	Veiksmai
1	00000000f6167d08	Veikiantis	Pakeisti Ištrinti
2	00000000d87e44f0a	Užblokuotas	Pakeisti Ištrinti
3	00000000e52e85f0a	Užblokuotas	Pakeisti Ištrinti

3.12 pav. Užregistruotų įrenginių sąrašas

Iš įrenginių sąrašo lango galima matyti, kurie įrenginiai yra veikiantys, o kurie užblokuoti. Galima pakeisti įrenginio informaciją arba ištrinti įrenginį. Taip pat yra galimybė pridėti naują įrenginį į sąrašą. Paspaudus mygtuką „pakeisti“, patenkama į įrenginio informacijos redagavimo formą, kuri pavaizduota 3.13 paveiksle.



3.13 pav. Įrenginio informacijos redagavimo forma

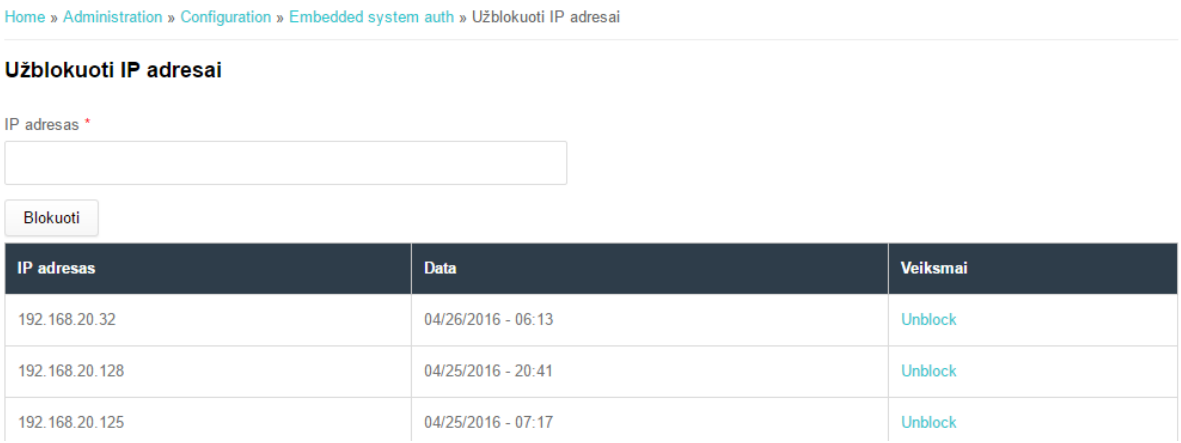
Įrenginio informacijos redagavimo formoje įvedamas įrenginio unikalus numeris, pagal kurį įrenginys identifikuojamas. Įvedami programinės įrangos ir duomenų bazės duomenų šifravimo slaptieji raktai. Juos galima suvesti arba sugeneruoti 512 bitų raktus automatiškai. Raktų generavimui naudojama *OpenSSL* biblioteka. Laukelyje „Viešas sistemos raktas“ patalpinamas viešas įrenginio raktas. Jei įrenginys dar neturi sugeneruotos raktų poros, tai ją galima sugeneruoti paspaudus mygtuką „Generuoti RSA raktų porą“. Sugeneruojama 2048 bitų privataus ir viešojo raktų pora. Tokiu atveju reikia privatųjį raktą nukopijuoti į įterptinės sistemos įrenginį.

Leidžiamų procesų sąraše įvedami tik tie procesai, kurie gali būti paleisti sistemos veikimo metu. Jei sistemoje paleista kita programinė įranga, aptinkamas neleistinas procesas, tai programinės įrangos slaptas šifravimo raktas nėra siunčiamas įrenginiui.

Testavimo tikslams galima įvesti leidžiamų vartotojų sąrašą. Standartiškai autentifikavimo metu patikrinama, ar sistemoje nėra prisijungusių vartotojų. Jei vartotojų nėra ir atitinka visi kiti reikalavimai, tik tada siunčiami slapti raktai. Tik tuomet, kai vartotojo vardas yra įrašomas į leidžiamų vartotojų sąrašą, jis gali būti prisijungęs autentifikavimo metu ir centrinis serveris perduos įrenginiui slaptus raktus.

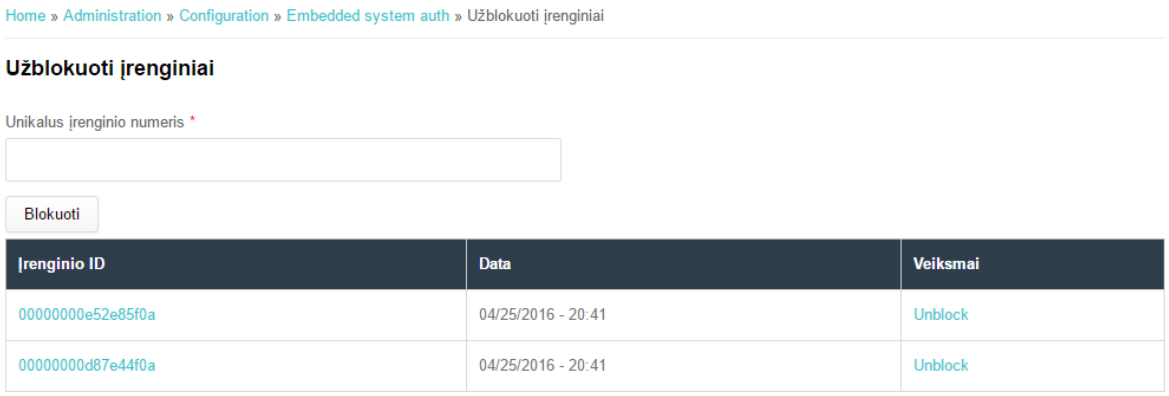
Leidžiamų įrenginių sąraše įrašomi įrenginiai, kurie gali būti prijungti prie įterptinės sistemos įrenginio sistemos paleidimo metu. Jei prijungtas neleistinas įrenginys, tuomet slaptieji raktai įrenginiui neperduodami.

Kontrolinių sumų sąraše įrašomos įterptinės sistemos komponentų kontrolinės sumos. Autentifikavimo metu jos paskaičiuojamos įrenginyje ir sulyginamos su esančiomis šiame sąraše. Taip patikrinama, ar nepakeista įrenginio programinė įranga.



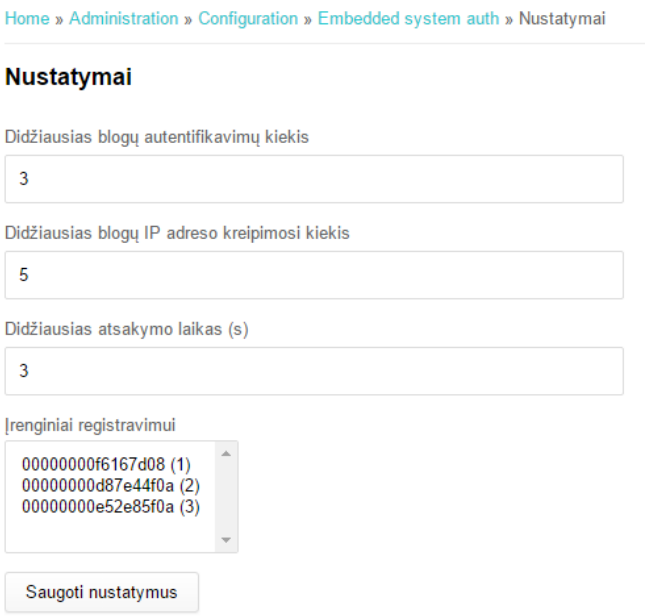
3.14 pav. Užblokuotų IP adresų sąrašas

3.14 paveiksle pateikta užblokuotų IP adresų sąrašo vartotojo sąsaja. Čia sistemos administratorius mato visus užblokuotus IP adresus. Jis gali juos atblokuoti arba į sąrašą pridėti naują IP adresą, kurį nori užblokuoti.



3.15 pav. Užblokuotų įrenginių sąrašas

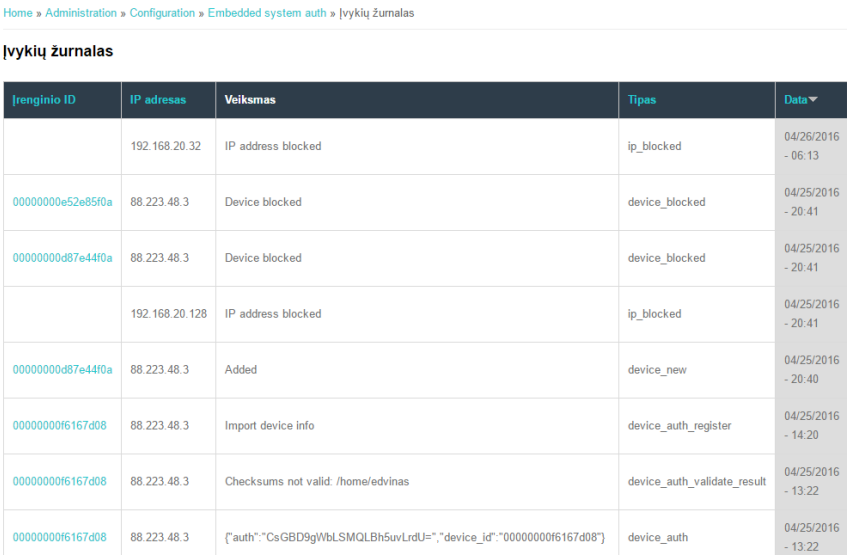
3.15 paveiksle pateikta užblokuotų įrenginių sąrašo vartotojo sąsaja. Čia sistemos administratorius mato visus užblokuotus įrenginius. Čia galima atblokuoti arba į sąrašą pridėti naują įrenginį, kurį norima užblokuoti.



3.16 pav. Sistemos nustatymų redagavimo forma

3.16 paveiksle pateikta sistemos nustatymų redagavimo vartotojo sąsaja. Sistemos administratorius gali nustatyti po kiek blogų autentifikavimų įrenginys yra užblokuojamas. Taip pat galima nustatyti po kiek blogų kreipinių užblokuojamas IP adresas, bei nustatyti laiko limitą, per kurį įrenginys turi būti autentifikuojamas.

Laukelis „Įrenginio registravimui“ yra skirtas pasirinkti įrenginio, kurio duomenis (leidžiamus įrenginius, procesus ir kontrolines sumas) norima importuoti iš pačio įrenginio, numeriui. Parašytas „auth_register.pyc“ failas, skirtas duomenų iš įrenginio surinkimui. Pradžioje parenkamas norimo įrenginio numeris. Tada įkeliamas failas į įterptinės sistemos įrenginį ir jis paleidžiamas. Automatiškai į sistemą suregistruojami tuo metu prijungti įrenginiai bei paleisti procesai. Paskaičiuojamos sistemos komponentų kontrolinės sumos.

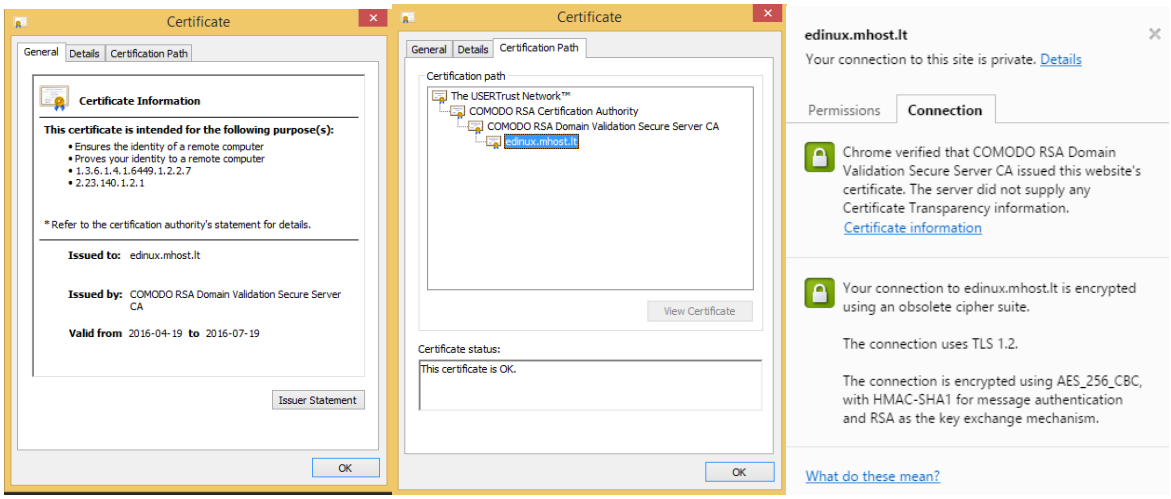


3.17 pav. Įvykių žurnalo rezultatų sąrašas

3.17 paveiksle pavaizduota įvykių žurnalo vartotojo sąsaja. Čia matomi duomenys apie visus sistemoje atliktus veiksmus. Registruojami veiksmai atlikti tiek administratoriaus, tiek pačios sistemos.

3.2.2. Duomenų apsauga tarp įrenginio ir centrinio serverio

Sistemoje vyksta duomenų mainai tarp įterptinės sistemos įrenginio ir centrinio serverio. Norint apsaugoti duomenis nuo perėmimo, naudojamas RC4 simetrinio šifravimo algoritmas duomenų užšifravimui. Kad būtų užtikrinamas saugus duomenų ryšys tarp centinio serverio ir kliento (sistemos administratoriaus arba įterptinės sistemos įrenginio) naudojamas 2048 bitų SSL sertifikatas. Taip išvengiama, kad keliaujantys duomenys būtų nuskaityti arba modifikuoti trečiųjų šalių.



3.18 pav. SSL sertifikato informacija

Kaip matoma iš 3.18 paveikslo, sertifikatas pasirašytas iki 2016-07-19. Naudojama 90 dienų sertifikato bandomoji versija, kurią išdavė „Comodo“ [24].

3.3. Išvados

Pagal sukurtą metodologiją atlikta prototipo realizacija. Programinė įranga, esanti įterptinėje sistemoje, parašyta *python* programavimo kalba. Programinė įranga maskuojama ir paverčiama į baitinį kodą, kad jį būtų sunkiau suprasti. Centriniame serveryje naudojamas SSL sertifikatas. Taip pat sukurtas „Drupal 7“ turinio valdymo platformai skirtas modulis, kuris yra naudojamas autentifikuoti ir perduoti slaptus raktus įterptinei sistemai. Sukurta vartotojo sąsaja, skirta sistemos administratoriui.

4. ĮTERPTINĖS SISTEMOS PROGRAMINIO KODO APSAUGOS PROTOTIPO ANALIZĖ

Realizavus prototipą buvo atlikta greitaveikos, energijos suvartojimo ir disko atminties užimtumo pasikeitimo analizė. Taip pat patikrinta, ar prototipas apsaugotas nuo kodo modifikavimo ir nelegalaus kodo pasisavinimo.

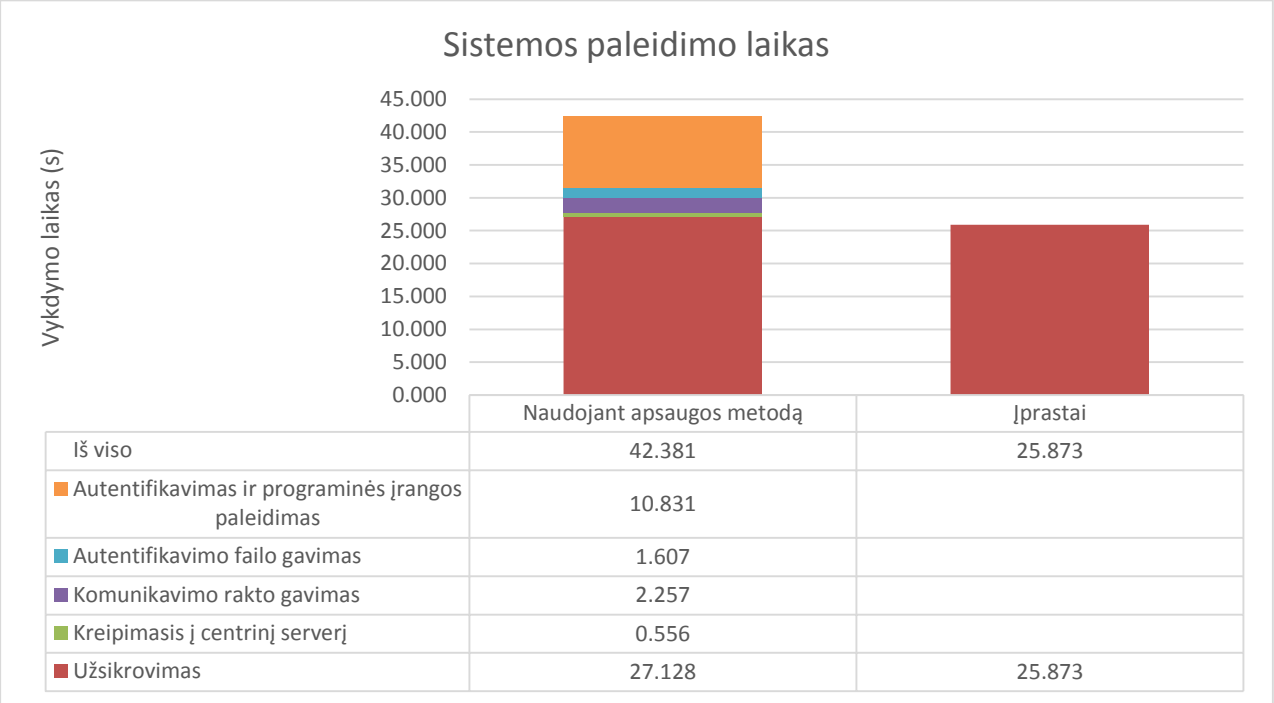
4.1. Įterptinės sistemos parametrų pasikeitimų analizė

Norint patikrinti sistemos paleidimo laiką, buvo išmatuojamas įprastos sistemos ir apsaugotos sistemos paleidimo laikas. Bandymai atliekami po 10 kartų.

Sistemos paleidimo laiko vidurkis paskaičiuojamas pagal formulę:

$$\overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \qquad (4.1)$$

Sistemos paleidimo laiko palyginimo diagrama pateikta 4.1 paveiksle.



	Naudojant apsaugos metodą	Įprastai
Iš viso	42.381	25.873
Autentifikavimas ir programinės įrangos paleidimas	10.831	
Autentifikavimo failo gavimas	1.607	
Komunikavimo rakto gavimas	2.257	
Kreipimasis į centrinį serverį	0.556	
Užsikrovimas	27.128	25.873

4.1 pav. Sistemos paleidimo laiko palyginimo diagrama

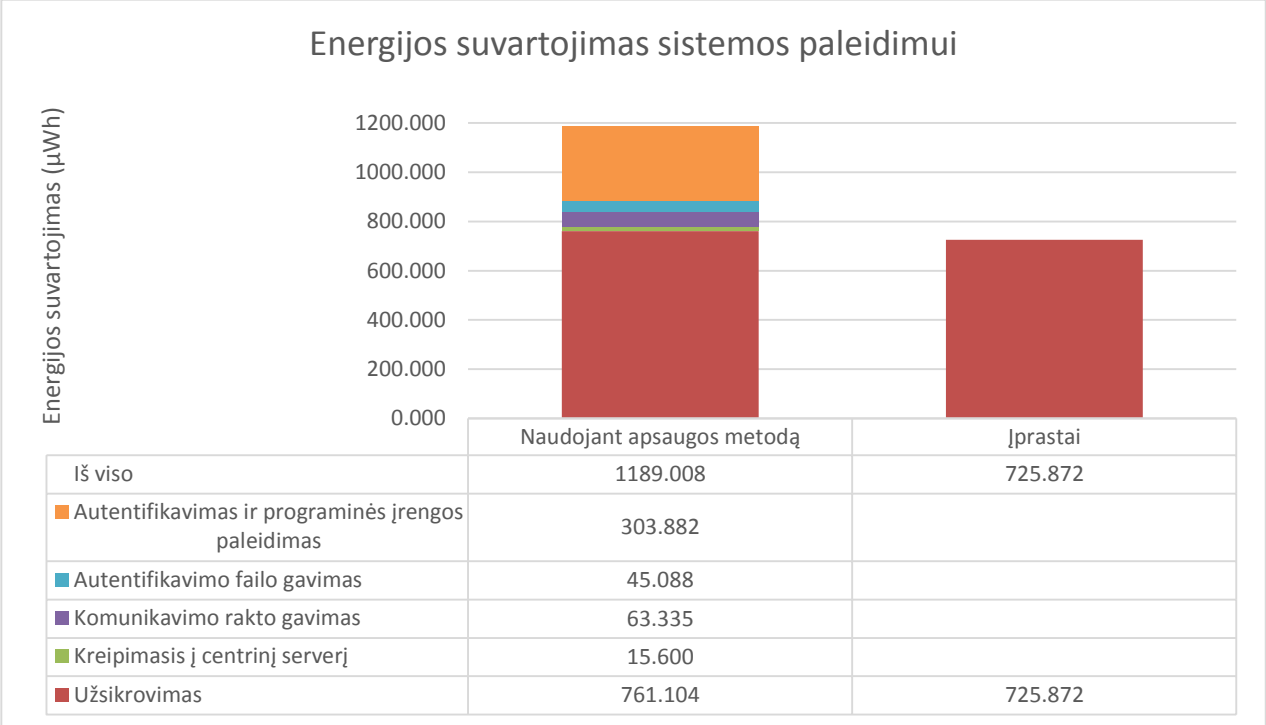
Iš 4.1 paveikslo matyti, kad sistemos paleidimo laikas apsaugojus programinį kodą padidėjo 16,508 s. Daugiausiai laikwo užtrunka pačio įrenginio užsikrovimas iki programinės įrangos paleidimo. Naudojant apsaugos priemones užsikrovimas pailgėjo nuo 25,873 s iki 27,128 s. Iš apsaugos priemonių vykdymo ilgiausiai užtrunka sistemos autentifikavimas, validavimas ir programinės įrangos paleidimas (10,831 s). Tai sudaro apytiksliai 66 proc. viso sistemos užsikrovimo laiko padidėjimo.

Sistemos energijos suvartojimo paskaičiavimui buvo papildomai išmatuota įtampa ir srovė. Įtampa U = 5,05 V ir srovė I = 20 mA.

Sunaudota energija sistemos paleidimui paskaičiuojama pagal formulę:

$$E = U * I * \overline{T} \qquad (4.2)$$

Energijos suvartojimo sistemos paleidimui palyginimo diagrama pavaizduota 4.2 paveiksle.

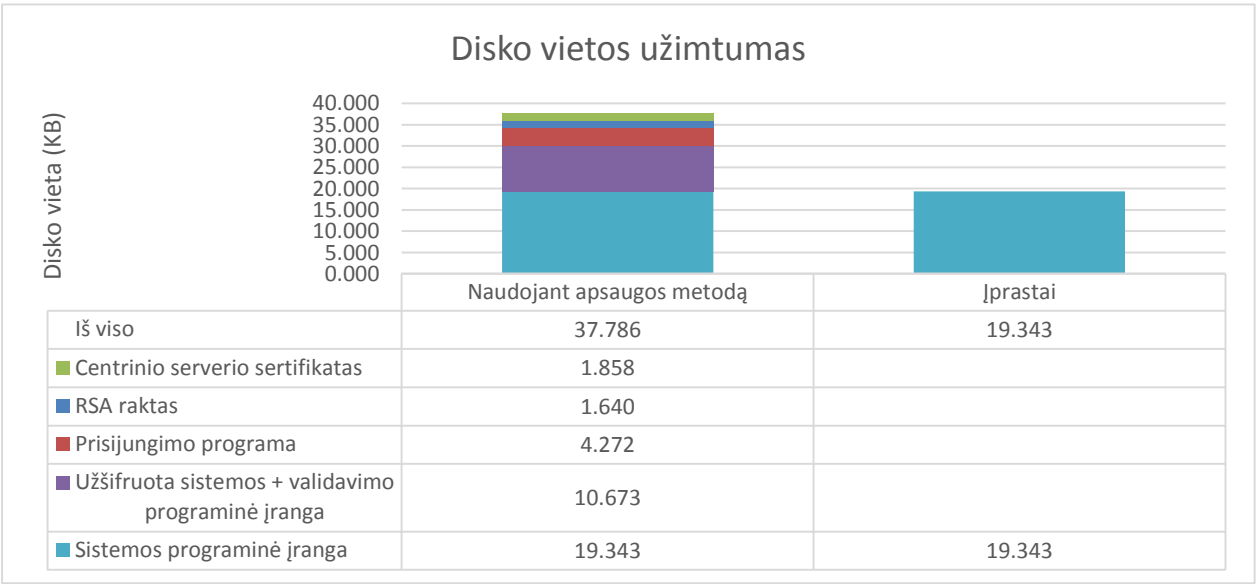


	Naudojant apsaugos metodą	Įprastai
Iš viso	1189.008	725.872
Autentifikavimas ir programinės įrengos paleidimas	303.882	
Autentifikavimo failo gavimas	45.088	
Komunikavimo rakto gavimas	63.335	
Kreipimasis į centrinį serverį	15.600	
Užsikrovimas	761.104	725.872

4.2 pav. Energijos suvartojimo sistemos paleidimui palyginimo diagrama

Iš 4.2 paveikslo matyti, kad energijos suvartojimas pasiskirstęs tolygiai kaip ir paleidimo laikas. Naudojant apsaugos metodą, energijos suvartojimas padidėjo ant 463,136 µWh.

4.3 paveiksle pavaizduota disko vietos užimtumo palyginimo diagrama.



	Naudojant apsaugos metodą	Įprastai
Iš viso	37.786	19.343
Centrinio serverio sertifikatas	1.858	
RSA raktas	1.640	
Prisijungimo programa	4.272	
Užšifruota sistemos + validavimo programinė įranga	10.673	
Sistemos programinė įranga	19.343	19.343

4.3 pav. Disko vietos užimtumo palyginimo diagrama

Iš 4.3 paveikslo matyti, kad sistemos disko vietos užimtumas padidėjo 18,443 KB. Tai beveik dvigubai daugiau nei nenaudojant apsaugos metodo. Didžiausias užimtumo padidėjimas atsiranda dėl

programinio kodo ir sistemos validavimo programinės įrangos užšifravimo (10,673 KB). Nors pati validavimo programinė įranga užima apie 2 KB.

4.2. Įterptinės sistemos programinio kodo apsaugos prototipo veikimo analizė

Buvo patikrinta, ar įmanoma apsaugotą programinę įrangą pasisavinti arba modifikuoti. Toliau aprašomi bandymai tą padaryti ir gauti rezultatai.

Bandymas modifikuoti įrenginyje esančius failus

Visi sistemos failai, reikalingi autentifikavimui ir programinio kodo apsaugai, saugomi namų kataloge. Todėl mėginta pakeisti ten esančius failus.

00000000f6167d08	88.223.48.3	Checksums not valid: /home	device	04/28/2016 - 20:39
00000000f6167d08	88.223.48.3	{"auth":"DcrJ8FGW15y4qhbaeru8MAs=","device_id":"00000000f6167d08"}	device	04/28/2016 - 20:39
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 20:39

4.4 pav. Bandymas modifikuoti sistemos failus

4.4 paveiksle pateiktas sistemos įvykių žurnalo užfiksuotas autentifikavimo ir validavimo rezultatas. Matome, kad namų katalogo kontrolinė suma neatitinka tos, kuri buvo įterptinės sistemos įrenginio paruošimo metu. Todėl programinio kodo dešifravimo kodai nėra siunčiami, o klaida užregistruota į žurnalą.

Bandymas prisijungti turint SSH privatųjį raktą

Šio bandymo metu, prisijungiama prie įrenginio naudojant SSH privatųjį raktą. Toks pats rezultatas būtų jungiantis ir su įsilaužėlio sukurtu vartotoju.

00000000f6167d08	88.223.48.3	Users not valid: edvinas	device	04/28/2016 - 20:58
00000000f6167d08	88.223.48.3	{"auth":"r9Gqk5i8OXsrRlo33sLbnFE=","device_id":"00000000f6167d08"}	device	04/28/2016 - 20:58
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 20:58

4.5 pav. Prisijungimas prie įrenginio per SSH prieigą

4.5 paveiksle paveiktas sistemos įvykių žurnalo rezultatas, su prisijungimo prie įrenginio naudojant SSH prieigą. Jei autentifikavimo metu prie sistemos buvo prisijungęs vartotojas, tai užregistruojama į sistemos įvykių žurnalą ir programinė įranga nedešifruojama.

Bandymas prijungti kitą įrenginį

Šio bandymo metu, prijungiamas papildomas įrenginys prie įterptinės sistemos įrenginio.

00000000f6167d08	88.223.48.3	Device not valid: Bus 001 Device 004: ID 125f:105b A-DATA Technology Co., Ltd.	device_auth_validate_result	04/29/2016 - 00:31
00000000f6167d08	88.223.48.3	{"auth":"pE8UDealApBLgH5gNrTxTzY=","device_id":"00000000f6167d08"}	device_auth	04/29/2016 - 00:31
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/29/2016 - 00:31

4.6 pav. Neleistino įrenginio prijungimas

4.6 paveiksle paveiktas sistemos įvykių žurnalo rezultatas, su prijungtu neleistinu įrenginiu. Jei autentifikavimo metu prie sistemos buvo prisijungtas papildomas įrenginys, kuris nėra leistinas, tai užregistruojama į sistemos įvykių žurnalą ir programinė įranga nedešifruojama.

Bandymas paleisti kenksmingą programinę įrangą

Sukurta kenksminga programinė įranga, kuri paleidžiama sistemai užsikrovus. Programinės įrangos tikslas persiųsti dešifruotą programinį kodą įsilaužėliui.

00000000f6167d08	88.223.48.3	Process not valid: /usr/bin/python /var/virus.py	device_auth_validate_result	04/28/2016 - 21:44
00000000f6167d08	88.223.48.3	{"auth":"pE8UDealApBLgH5gNrTxTzY=","device_id":"00000000f6167d08"}	device_auth	04/28/2016 - 21:44
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 21:44

4.7 pav. Įrenginyje paleistos kenksmingos programinės įrangos aptinimas

4.7 paveiksle paveiktas sistemos įvykių žurnalo rezultatas, su paleista kenksminga programine įranga. Autentifikavimo ir validavimo metu rastas procesas pavadinimu *„/usr/bin/python /var/virus.py*“. Programinė įranga nedešifruojama, o klaida užregistruota į įvykių žurnalą.

Bandymas sutrikdyti sistemos veiklą

Bandant sutrikdyti sistemos veiklą ir taip sukelti sistemos klaidą, norint išgauti slaptus duomenis, sulėtėjo sistemos veikimas. Centrinis serveris negavo reikiamų duomenų per numatytą laiką, todėl užregistruojama, kad pasibaigė sistemos autentifikavimo ir validavimo laikas. Tai pavaizduota 4.8 paveiksle.

00000000f6167d08	88.223.48.3	Timeout	device	04/28/2016 - 20:54
00000000f6167d08	88.223.48.3	{"auth":"wjcl2AF7bH29TqW8fxo8ySU=","device_id":"00000000f6167d08"}	device	04/28/2016 - 20:54
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 20:54

4.8 pav. Per ilgas atsakymo iš įrenginio laikas

Programinis kodas šiuo atveju taip pat nėra dešifruojamas.

IP adreso užblokavimas

4.9 paveiksle pateiktas įvykių žurnalas, su IP adreso užblokavimu. IP adresas užblokuojamas tada, kai pasiekiamas didžiausiais leistinas blogų kreipinių kiekis iš IP adreso (žr. 3.16 pav.). Užblokavus IP adresą iš jo nepriimami autentifikavimo prašymai. Tik sistemos administratorius gali atblokuoti IP adresą.

	88.223.48.3	IP address blocked	ip_blocked	04/28/2016 - 21:12
00000000f6167d08	88.223.48.3	{"auth":"Qhtf1yCk2w9uJlk058MgdR0=","device_id":"00000000f6167d08"}	device	04/28/2016 - 21:12
00000000f6167d08	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 21:12

4.9 pav. IP adreso blokavimas po nesėkmingo autentifikavimo

Įrenginio užblokavimas

4.10 paveiksle pateiktas įvykių žurnalas, su įrenginio užblokavimu. Įrenginys užblokuojamas tada, kai pasiekiamas didžiausiais leistinas blogų kreipinių kiekis iš įrenginio (žr. 3.16 pav.). Užblokavus įrenginį iš jo nepriimami autentifikavimo prašymai. Tik sistemos administratorius gali atblokuoti įrenginį.

00000000f6167d08	88.223.48.3	Device blocked	device	04/28/2016 - 17:31
	88.223.48.3	{"device_id":"00000000f6167d08"}	device	04/28/2016 - 17:31

4.10 pav. Įrenginyje blokavimas po nesėkmingo autentifikavimo

Mėginimas išgauti programinį kodą iš veikiančios sistemos

Sėkmingai autentifikavus sistemą ir dešifravus programinį kodą galima mėginti perimti, jau dešifruotą programinį kodą. Tai galima mėginti padaryti mėginant prijungti kitą įrenginį arba prisijungti prie įrenginio naudojant SSH prieigą.

Prijungus kitą įrenginį arba prisijungus per SSH prieigą programinio kodo vykdymas sustabdomas. Taip pat sunaikinami duomenys esantys virtualiam RAM diske. Duomenys sunaikinami neatstatomai, todėl jų nuskaityti ir atkurti nepavyksta.

Užkrovimo nepaleidus ataka

Sukurtas metodas neapsaugo nuo užkrovimo nepaleidus atakos. Kol sistema pajungta į elektros šaltinį, tol vykdoma programinė įranga, kurios metu tikrinama, ar nevykdomas įsilaužimas. Bet jei panaudojama užkrovimo nepaleidus ataka, galima pasidaryti RAM atminties kopiją. Turint RAM atminties kopiją, panaudojus apgrąžos inžineriją, galima mėginti atkurti programinį kodą.

Tai praktiškai atlikti nepavyko, nes reikalaujama daugiau laiko RAM atminties kopijos analizei. Bei reikalinga programinė įranga RAM atminties nuskaitymui. Bet tai teoriškai padaryti įmanoma.

4.3. Išvados

Atlikus prototipo analizę paaiškėjo, kad sistemos paleidimo laikas pailgėjo 16,508 s. Taigi, sistema sulėtėjo 63,8 proc. Sistemai pradėjus darbą, programinės įrangos vykdymo laikas nepasikeitė. Panaudoti keli įsilaužimo būdai ir patikrinta, kaip veikia įrenginio autentifikavimas. Nerasta operacinė

sistema, šifruojanti duomenis esančius RAM atmintyje, kuri tiktų „Raspberry Pi“ įrenginiui. Todėl įterptinės sistemos nepavyko apsaugoti nuo paleidimo neužkrovus atakos.

Pilnai apsaugoti programinio kodo esančio „Raspberry Pi“ architektūros įterptinėje sistemoje nepavyko. Norint tai padaryti, reikia pritaikyti arba sukurti operacinę sistemą, kuri būtų atspari paleidimo neužkrovus atakai. Arba, turint daugiau išteklių naudoti brangesnį arba savo sukurtą įrenginį.

5. IŠVADOS

- ✓ Atlikus įterptinių sistemų / architektūrų ir jų saugos priemonių analizę, nustatytos dažniausios įterptinių sistemų saugumo silpnosios vietos ir apsaugos priemonės;
- ✓ Atlikus duomenų ir programų šifravimo / dešifravimo algoritmų analizę nustatyta, kad:
  - ✓ Geriausiai programinio kodo šifravimui / dešifravimui tinka RC4 simetrinis šifravimo algoritmas;
  - ✓ Geriausiai slapto rakto perdavimui tinka RSA viešojo rakto šifravimo algoritmas;
  - ✓ Geriausiai tinkanti maišos funkcija kontrolinių sumų skaičiavimui yra SHA-1.
- ✓ Sudaryta įterptinių sistemų programinės įrangos saugos priemonių struktūra;
- ✓ Sudarytas įterptinių sistemų programinės įrangos apsaugos algoritmas;
- ✓ Sudaryta programinės įrangos saugos užtikrinimo bei diegimo metodika;
- ✓ Atlikta praktinė įterptinės sistemos programinės įrangos saugumo prototipo realizacija;
- ✓ Atliktas praktinis įterptinės sistemos programinės įrangos saugumo prototipo tyrimas;
- ✓ Darbo metu nepavyko rasti apsaugos priemonių nuo paleidimo neužkrovus atakos, tinkančių „Raspberry Pi“ architektūros įterptinei sistemai. Todėl siūlomi šie sprendimai:
  - ✓ Sukurti arba pritaikyti operacinę sistemą, kuri būtų atspari paleidimo neužkrovus atakai;
  - ✓ Sukurti papildomą techninę įrangą, kuri padėtų apsisaugoti nuo paleidimo neužkrovus atakos;
  - ✓ Norint didesnės apsaugos naudoti kitą arba sukurti savo įrenginį, kuris šiai atakai būtų atsparus.

6. LITERATŪRA

[1] B. Filipovič ir O. Schimmel, „Protecting embedded system against product piracy,“ [Tinkle]. Available: http://www.aisec.fraunhofer.de/content/dam/aisec/Dokumente/Publikationen/Studien_TechReports/englisch/Whitepaper_ProductProtection.pdf. [Kreiptasi 15 11 2014].

[2] „Arduino, Raspberry Pi, BeagleBone and PcDuino,“ [Tinkle]. Available: http://randomnerdtutorials.com/arduino-vs-raspberry-pi-vs-beaglebone-vs-pcduino. [Kreiptasi 10 12 2014].

[3] „The Raspberry Pi has Its Potentials and Perils,“ [Tinkle]. Available: http://www.trendmicro.com/vinfo/us/security/news/internet-of-things/raspberry-pi-has-its-potentials-and-perils. [Kreiptasi 05 12 2014].

[4] „ARM1176JZF-S Technical Reference Manual,“ [Tinkle]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0301h/ddi0301h_arm1176jzfs_r0p7_trm.pdf. [Kreiptasi 8 12 2014].

[5] „BeagleBone Black,“ [Tinkle]. Available: http://beagleboard.org/Products/BeagleBone%20Black. [Kreiptasi 12 01 2015].

[6] D. Kleidermacher ir M. Kleidermacher, „Embedded system complexity,“ įtraukta *Embedded systems security practical methods for safe and secure software*, 2012, p. 4.

[7] „Embedded systems security - an overview,“ [Tinkle]. Available: http://www.ecs.umass.edu/ece/wolf/pubs/daes2008.pdf. [Kreiptasi 12 01 2015].

[8] „Using public keys for SSH authentication,“ [Tinkle]. Available: http://the.earth.li/~sgtatham/putty/0.55/htmldoc/Chapter8.html. [Kreiptasi 10 01 2015].

[9] „5 steps to secure embedded software,“ [Tinkle]. Available: http://embedded-computing.com/articles/5-steps-secure-embedded-software/. [Kreiptasi 12 08 2015].

[10] „Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy,“ [Tinkle]. Available: http://www.cse.psu.edu/~pdm12/cse597g-f15/readings/cse597g-embedded_systems.pdf. [Kreiptasi 12 08 2015].

[11] „The Ongoing Threat of Cold Boot Attacks,“ [Tinkle]. Available: https://www.technologyreview.com/s/530186/the-ongoing-threat-of-cold-boot-attacks/. [Kreiptasi 12 05 2015].

[12] A. Miller, Performance Investigation of Various Cryptographic Techniques in Embedded Systems, 2010.

[13] A. a. J. B. Petzoldt, A Multivariate Signature Scheme with an almost cyclic public key, 2009.

[14] „Kriptografija - šifravimas ir dešifravimas,“ [Tinkle]. Available: http://www.straipsniai.lt/Kriptografija/puslapis/8203. [Kreiptasi 12 01 2015].

[15] „Hash function,“ [Tinkle]. Available: https://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html. [Kreiptasi 05 01 2015].

[16] „Setting up a (reasonably) secure home web-server with Raspberry Pi,“ [Tinkle]. Available: https://mattwilcox.net/web-development/setting-up-a-secure-home-web-server-with-raspberry-pi. [Kreiptasi 12 06 2015].

[17] „Setting up a (reasonably) secure home web-server with Raspberry Pi,“ [Tinkle]. Available: https://mattwilcox.net/web-development/setting-up-a-secure-home-web-server-with-raspberry-pi. [Kreiptasi 10 04 2016].

[18] „How To Set Up SSH Keys,“ [Tinkle]. Available: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2. [Kreiptasi 10 04 2016].

[19] „Pyminifier,“ [Tinkle]. Available: https://liftoff.github.io/pyminifier/. [Kreiptasi 10 04 2016].

[20] „Create a RAM disk in Linux,“ [Tinkle]. Available: https://www.jamescoyle.net/how-to/943-create-a-ram-disk-in-linux. [Kreiptasi 10 04 2016].

[21] [Tinkle]. Available: https://www.howtoforge.com/tutorial/how-to-encrypt-a-linux-partition-with-dm-crypt-luks/. [Kreiptasi 10 04 2016].

[22] „Linux and Unix shred command,“ [Tinkle]. Available: http://www.computerhope.com/unix/shred.htm. [Kreiptasi 10 04 2016].

[23] [Tinkle]. Available: https://www.drupal.org/project/aes. [Kreiptasi 10 04 2016].

[24] „Free SSL Certificate for 90 days,“ [Tinkle]. Available: https://www.comodo.com/e-commerce/ssl-certificates/free-ssl-certificate.php. [Kreiptasi 19 04 2015].

[25] NIST - National Institute of Standards and Technology, 1999.

[26] B. Schneier, Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C, 2005.

[27] A. Plotnikov, Logical cryptoanalysis on the example of the cryptosystem DES, 2010.

[28] G. Schorcht, Lecture notes: IT-Security, 2010.

[29] W. Stallings, Cryptography and Network Security: Principles and Practice (5th Edition), 2010.

[30] D. D. Coleman, D. A. Westcott, B. E. Harkins ir S. M. Jackman, Certified Wireless Security Professional Official Study Guide.

7. PRIEDAI

7.1. Duomenų bazės konfigūravimo instrukcija

Norint pagerinti MySQL saugumą reikia sutvarkyti duomenų bazės saugumo konfigūraciją. Pagrindinė užduotis yra sumažinti MySQL pažeidžiamumą.

7.1.1. Tolimosios prieigos suvaržymas

Norint suvaržyti MySQL nuo tinklo lizdo atidarymo, reikia pridėti tokius parametrus [mysqld] sekcijoje „my.cnf“ arba „my.ini“:

```
skip-networking
```

Failas yra „/etc/my.cnf“ arba „/etc/mysql/my.cnf“ Linux‘ų operacinėje sistemoje.

Ši eilutė sutrukdo prisijungimo prie sistemos iniciavimą MySQL paleidimo metu. Reikia pastebėti, kad naudojantis vietiniu prisijungimu galima prisijungti prie MySQL serverio.

Tada reikia MySQL leisti vykdyti tik vietinio tinklo („localhost“) užklausas. Tai galima padaryti pridedant tokią eilutę [mysqld] sekcijoje, kuri yra „my.cnf“:

```
bind-address=127.0.0.1
```

7.1.2. „LOCAL INFILE“ naudojimo apribojimas

Sekantis žingsnis yra apriboti „LOAD DATA LOCAL INFILE“ komandos naudojimą, kuris padės išvengti neautorizuoto vietinių failų skaitymo. Tai ypatingai svarbu, kai PHP aplikacijoje yra randamas naujas SQL įskiepio pažeidžiamumas. Be to, tam tikrais atvejais, „LOCAL INFILE“ komanda gali būti panaudota gauti prieigą prie kitų operacinėje sistemoje esančių failų, tarkim „/etc/passwd“, panaudojant tokią eilutę:

```
mysql> LOAD DATA LOCAL INFILE '/etc/passwd' INTO TABLE table1
```

Arba paprasčiau:

```
mysql> SELECT load_file("/etc/passwd")
```

Norint apriboti „LOCAL INFILE“ naudojimą, sekantys parametrai turi būti pridėti [mysqld] sekcijoje į MySQL konfigūracijos failą:

```
set-variable=local-infile=0
```

7.1.3. Šakninio vartotojo prisijungimo vardo bei slaptažodžio pakeitimas

Numatytasis administratoriaus prisijungimo vardas MySQL serveryje yra „root“. Programišiai dažnai bando įgyti prieigą prie šio vartotojo. Norint padaryti, kad šis procesas būtų sunkesnis, reikia pakeisti „root“ vartotojo vardą į kokį nors kitą ir suteikti ilgą, sudėtingą slaptažodį iš raidžių ir skaičių.

Norint pakeisti administratoriaus prisijungimo vardą, reikia panaudoti pervadinimo komandą MySQL konsolėje:

```
mysql> RENAME USER root TO new_user;
```

MySQL „RENAME USER“ komanda pasirodė MySQL 5.0.2. versijoje. Jeigu yra naudojama senesnė MySQL versija, tuomet reikėtų panaudoti kitas komandas. Norint pakeisti vartotojo vardą:

```
mysql> use mysql;
mysql> update user set user="new_user" where user="root"; mysql> flush privileges;
```

Norint pakeisti vartotojo slaptažodį, naudojama tokia komandinė eilutė:

```
mysql> SET PASSWORD FOR 'username'@'%hostname' = PASSWORD('newpass');
```

Taip pat yra galimybė pakeisti slaptažodį naudojant "mysqladmin":

```
shell> mysqladmin -u username -p password newpass
```

7.1.4. „Test“ duomenų bazės pašalinimas

Įdiegiant MySQL kartu yra sukuriama „test“ duomenų bazė. Prie jos gali prieiti anoniminiai vartotojai, tai naudojama daugelyje atakų. Norint pašalinti šią duomenų bazę, reikia naudoti tokią komandą:

```
mysql> drop database test;
```

Arba naudoti „mysqladmin“ komandą:

```
shell> mysqladmin -u username -p drop test
```

7.1.5. Anoniminių ir nenaudojamų vartotojų šalinimas

Įdiegiant MySQL duomenų bazę, jau būna sukurta keletas anoniminių vartotojų be slaptažodžių. Dėl to, bet kas gali prisijungti prie duomenų bazės. Norint tai patikrinti, reikia suvesti:

```
mysql> select * from mysql.user where user="";
```

Saugioje sistemoje, neturėtų būti gaunamas joks atsakymas. Kitas būdas atlikti tą patį:

```
mysql> SHOW GRANTS FOR "@'localhost';
mysql> SHOW GRANTS FOR "@'myhost';
```

Jei dotacijos (grants) egzistuoja, tuomet bet kas gali prisijungti prie duomenų bazės ir mažiausiai, ką gali padaryti, tai pasinaudoti „test“ duomenų baze. Tai patikrinti galima taip:

```
shell> mysql -u test
```

Norint pašalinti šitą vartotoją, reikia įvykdyti tokią komandą:

```
mysql> DROP USER "";
```

„DROP USER“ komanda yra palaikoma nuo MySQL 5.0 versijos. Jei naudojama senesnė versija, tuomet reikėtų naudoti:

```
mysql> use mysql;
mysql> DELETE FROM user WHERE user="";
mysql> flush privileges;
```

7.1.6. Žemesnės sistemos privilegijos; duomenų bazės saugumo padidinimas su „Role Based Access Control“

Bendra duomenų bazių saugumo rekomendacija – sumažinti prieigos teises suteiktas įvairioms šalims. MySQL nėra išskirtinė. Įprastai, kai programuotojai dirba, jie naudoja maksimalias prieigos teises prie sistemos. Ir apie turimas teises mąsto mažiau, nei kad jiems atrodo. Dėl to gali iškilti didelė rizika duomenų bazei.

Norint apsaugoti duomenų bazę, reikia įsitikinti, kad, iš tikrųjų, failų direktorija, kurioje yra MySQL duomenų bazė, priklauso „mysql“ vartotojui ir „mysql“ grupei:

```
shell>ls -l /var/lib/mysql
```

Taip pat, reikia patikrinti, kad tik vartotojai „mysql“ ir „root“ turi priėjimą prie „directory/var/lib/mysql“.

Mysql, kuris yra /usr/bin/ kataloge, turi būti valdomas „root“ arba specifinės sistemos „mysql“ vartotojo. Kiti vartotojai neturi turėti galimybės redaguoti šių failų:

```
shell>ls -l /usr/bin/my*
```

7.1.7. Žemesnės duomenų bazės privilegijos

Daugumoje, yra administratoriaus vartotojas (pervadintas „root“) ir vienas arba daugiau vartotojų, kurie kartu egzistuoja duomenų bazėje. Įprastai „root“ nieko nedaro su duomenimis duomenų bazėje, vietoj to, jis palaiko serverį ir jo lenteles, suteikdamas arba panaikindamas prieigos teises ir kt.

Iš kitos pusės, keletas vartotojų ID yra naudojami pasiekti duomenis. Pavyzdžiui, vartotojo ID yra priskirtas prie žiniatinklio serverio tam, kad vykdytų „select\update\insert\delete“ užklausas ir kitas saugomas procedūras. Daugeliu atvejų, daugiau jokių vartotojų nebereikia, tačiau tik sistemos administratorius gali žinoti taikomosios programos poreikius.

Tik administratoriaus paskyra turi turėti SUPER / PROCESS /FILE privilegijas ir prieigą prie mysql duomenų bazės. Paprastai, gera idėja būna sumažinti administratoriaus privilegijas prieiti prie duomenų.

Reikia peržiūrėti likusių vartotojų privilegijas ir įsitikinti, kad jos nustatytos tinkamai. Tai galima padaryti taip:

```
mysql> use mysql;
[Identify users]
mysql> select * from users;
[List grants of all users]
mysql> show grants for ‘root’@’localhost’;
```

Anksčiau paminėtas patvirtinimas turi būti atliktas kiekvienam vartotojui. Reikia pastebėti, kad tik vartotojai, kuriems tikrai reikia šakninių privilegijų turi jas turėti.

Kita įdomi privilegija yra „SHOW DATABASES“. Numatyta, kad šią komandą gali naudoti bet kas, kas turi prieigą prie MySQL komandinės eilutės. Jie gali naudoti šią privilegiją renkant informaciją (pavyzdžiui, gaudami duomenų bazės vardus) prieš užpuldami DB, pavyzdžiui tam, kad pavogtų duomenis. Norint to išvengti yra rekomenduojama įvykdyti žemiau aprašytą procedūrą.

Pridėti „--skip-show-database“ pradiniame MySQL scenarijuje arba pridėti šitą eilutę MySQL konfigūracijos faile.

Suteikti SHOW DATABASES privilegiją tik tiems vartotojams, kuriems administratorius leidžia naudotis šia komanda.

Norint panaikinti "SHOW DATABASES" komandos naudojimą, reikia pridėti sekantį parametrą į [mysqld] sekciją, kuri yra „/etc/my.cnf“:

[mysqld] skip-show-database

7.1.8. Registravimo įjungimas

Jei duomenų bazės serveris nevykdo daugelio užklausų, rekomenduojama įjungti operacijos registravimą, tai galima padaryti pridedant eilutę [mysqld] sekcijoje „/etc/my.cnf“ faile:

[mysqld] log =/var/log/mylogfile

Tai nėra rekomenduojama sunkiosios MySQL produkcijos serveriams, kadangi tai sukelia didelį krūvį serveriui. Be to, reikia įsitikinti, kad tik „root“ ir „mysql“ id turi prieigą prie registro failų.

7.1.9. Klaidų registras

Reikia įsitikinti, kad tik „root“ ir „mysql“ turi prieigą prie registracijos failo „hostname.err“. Šis failas yra mysql duomenų direktorijoje. Jame yra labai svarbi informacija, tokia kaip slaptažodžiai, adresai, lentelių pavadinimai, saugomų procedūrų pavadinimai ir kodo dalys. Tai gali būti panaudota informacijos surinkimui. Ir kai kuriais atvejais atakuotojams gali būti suteikta informacija, kuri reikalinga norint panaudoti duomenų bazę, įrenginį, kuriame yra sudiegta duomenų bazė, arba duomenis, kurie yra įrenginio viduje.

7.1.10. MySQL registras

Reikia įsitikinti, kad tik „root“ ir „mysql“ turi prieigą prie registro failo „*logfileXY“. Šis failas yra saugomas mysql duomenų direktorijoje.

7.1.11. Šaknies direktorijos keitimas

Chroot - tai (Unix operacinėse sistemose) operacija, kuri pakeičia esamą šakninį katalogą dabartiniam procesui ir jo vaikams. Programa, kuri iš naujo įsišaknija į kitą katalogą, negali pasiekti arba pervadinti failų, kurie nėra tame kataloge ir toks katalogas yra vadinamas šaknies kalėjimu („chroot jail“).

Naudojant šaknies katalogo pakeitimo aplinką, prieigos prie MySQL procesų (ir vaikinių procesų) nustatymas gali būti apribotas, taip didinant serverio saugumą.

Reikia įsitikinti, kad numatytasis katalogas yra aptariamoje aplinkoje. Tai atrodo maždaug taip: „/chroot/mysql“. Be to, norint panaudoti duomenų bazės administravimo įrankius, reikia pakeisti sekančius parametrus MySQL konfigūracijos faile, [client] sekcijoje:

[client] socket = /chroot/mysql/tmp/mysql.sock

Dėl pastarosios eilutės nebereikia kiekvieną kartą pateikti mysql, mysqladmin, mysqldump ir kt. komandų, pasitelkiant „--socket=/chroot/mysql/tmp/mysql.sock“ parametrus, kai naudojami šie įrankiai.

7.1.12. Istorijos šalinimas

Per įdiegimo procedūras, yra pateikiama labai daug svarbios informacijos, kuri gali padėti užpuolikui įsibrauti į duomenų bazę. Ši informacija yra laikoma serverio istorijoje ir yra naudinga kuomet per įdiegimo procesą įvyksta kažkas blogo. Analizuodamas istorinius failus, administratorius gali suprasti, kur įvyko klaida ir tai pataisyti. Tačiau šie failai nebėra reikalingi po įdiegimo proceso.

Reikia pašalinti MySQL istorinių failų turinį („~/.mysql_history“), kuriame yra saugomos visos vykdytos SQL komandos (ypatingai slaptažodžiai, kurie laikomi kaip paprastas tekstas):

```
cat /dev/null > ~/.mysql_history
```