TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Kalmer Keerup

Kasutaja õigustes töötava failiserveri veebiliidese teostus

Bakalaureusetöö (9 EAP)

Juhendaja(d): Tarmo Oja, MSc
Heili Orav, PhD

Tartu 2022

Kasutaja õigustes töötava failiserveri veebiliidese teostus

Lühikokkuvõte:

Levinud failiserveri teenuste kliendid on tihti tugevalt integreeritud operatsioonisüsteemidesse ja omavad erinevaid adresseerimisvorme, mis enamasti erinevate operatsioonisüsteemide vahel ei ühildu. Asukohtadele viitamine on aadressi vormide erinevustest tulenevalt keeruline ja jätab viitade mõistmise ja rakendamise kasutajale. Töö eesmärk oli luua universaalselt adresseeritav failiserveri liides, mis järgib süsteemset autentimist ja failide juurdepääsu reegleid. SIMU lahendus pakub autentimise ja POSIX juurdepääsuõiguste alusel veebiliidesega failiserveri rakendust.

Võtmesõnad:

süsteemiprogrammeerimine, failiserver, Rust, veebirakendus, SUID-rakendus

CERCS: P175 Informaatika, süsteemiteooria

Implementation of an user access control respecting file server

Abstract:

Common file server systems' clients are often tightly integrated to operation systems and can have different forms of addressing that are often not portable between different operating systems. Communicating locations is thus more complicated due-to different forms and thus the users need to be aware of different forms and how to make use of them. This thesis aims to create an universally addressable file server interface that uses system's authentication and file access rules. SIMU solution provides a system authentication and POSIX access control lists applying file server service over a HTTP interface.

Keywords:

systems programming, file server, Rust, web application, SUID-application

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Failiserverid võimaldavad jagada infovarasid erinevate inimeste ja arvutite vahel. Erinevates valdkondades on levinud vajadus infovarade jagamiseks, seega eksisteerib mitmeid lahendusi jagamise haldamiseks. Sellised lahendused on oma olemuselt erinevad ja toovad esile erinevaid nüansse nagu näiteks kiirus, seadistatavus või lihtsus. Failide pääsuhalduse rakendamine failiserveris muudab selle sisu kasutamise tavalisest keerulisemaks, kuna kasutajale nähtav osa failidest ja kaustadest sõltub tugevalt kasutajale määratud õigustest. Kui lisada sellele haldusele ka vajadus kasutada nende rakendamiseks operatsioonisüsteemi autentimise informatsiooni, väheneb sobivate lahenduste hulk kiiresti.

Töö eesmärk on luua veebibrauseri abil juurdepääsetav failiserveri liides, mis täielikult järgib süsteemisisest autentimist ja rakendab eelnevalt autenditud kasutaja õiguseid. Töö on ajendatud Cybernetica AS[1] vajadustest ning avaldatakse vabavarana ja bakalaureuse tööna Tartu Ülikooli arvutiteaduse instituudis.

Esimeses peatükis antakse ülevaade failiserverite liidestest, kirjeldatakse olemasolevate lahenduste omadusi ning vajadusi selliste liideste kasutamise lihtsustamiseks ja nendega seotud probleeme. Teises peatükis kirjeldatakse loodud lahenduse parema mõistmise tarvis seotud üldisi põhimõtteid ja meetodeid nagu näiteks turvastandardid ja väljavõtted POSIX-i (Portable Operating System Interface) standardist, mis on asjakohased failiserverite arendamisel. Kolmandas peatükis kirjeldatakse lahenduse teostuse detaile, täpsemalt nõudeid, arhitektuurilisi omadusi ja nende tagamaid ning muid implementatsiooni detaile. Neljandas peatükis tuuakse välja koormustesti tulemused ja näide sellisele süsteemile integratsiooni testi kirjutamisest ja selle kasutamisest. Lisades antakse sisupeatükke illustreerivaid pikemad koodinäiteid ja viide lähtekoodile.

[1] Cybernetica AS on Eestis asutatud infotehnoloogiline teadus- ja arendusettevõte, mille peamised tegevusalad on olnud andmevahetustehnoloogiad, digitaalne identiteet ja infoturve [Cyb].

1 Failiserveri liidestest

Infovaradega tegelevatel ettevõtetel on tavapärane luua selline infoturvapoliitika, mis sätestab täpsed reeglid infovarade kasutamiseks ja talletamiseks. Failiserverid enamasti on arendatud või kohaldatud neid reegleid ellu viima läbi failiserveris rakendatud piirangute, võimaldades näiteks juurdepääsu vaid vajaduse alusel.

Selliste piirangute valguses on raske leida võrguketaste lahendusi, mis oleks lisaks eelnevale ka mugav kasutada ja keskselt hallatav.

1.1 Teised võrguketaste lahendused

Sellest tulenevalt kasutatakse Cyberneticas peamiselt SMB (Server Message Block) või SSH (Secure Shell) protokolli abil ligipääsetavat failiserverit, millel on autentimine ja autoriseerimine liidestatud LDAP-protokolli (Lightweight Directory Access Protocol) toetava kataloogiteenuse vastu. SMB on failide ja printerite jagamise protokoll, tänaseks on peamiselt levinud selle CIFS (Common Internet File System) dialekt, mille on peamiselt välja arendanud Microsoft Windowsil implementeerimiseks [Mic20], kuid on kasutustes ka programmis Samba, mis on sarnane serverlahendus Unixile sarnanevatele süsteemidele, kus kasutatakse teadlikult CIFS-i asemel SMB-d, kuna tegemist on selle otsese järglasega ja seda peetakse pigem lihtsalt olemasoleva protokolli ümbernimetamiseks kui dialektiks [TE03].

SSH (Secure Shell) võimaldab teha üldisemaid turvalisi ühendusi kliendi ja serveri vahel, millel võib olla erinevaid otstarbeid, näiteks käsu käivitamine, failide edastus või interaktiivse konsooli avamine [BS01].

Läbi SMB või SSH protokolli toetavate võrguketta lahenduste suureks eeliseks on levinud liidestused erinevates töölaua operatsioonisüsteemides, kuid need liidestused on tihti pisut erinäolised üle operatsioonisüsteemide. Sellest tulenevalt puudub näiteks võimalus operatsioonisüsteemist või liidestuvast rakendusest sõltumata jagada viiteid erineva süs-

teemi või rakenduse kasutajale. Näitena Windowsi erinevatel variantidel eelistatakse UNC (Universal Naming Convention) süntaksil viiteid, st „`\\serverinimi\ kausta\teekond`" kuju [Mic20], samas macOSil eelistatakse „`smb://serverinimi/kausta/teekond`" mudelit [App]. Seega peab ühelt platvormilt teisele viidet jagades tihti kohandama ümber ka jagatava viite.

1.2 Varasemad lahendused

Varasemalt on ettevõttes kasutatud selleks ka sisemiselt võrguketta lihtsa juurdepääsu probleemi lahendavat lahendust, mis tegi lisaks SMB ja SSH kahele protokollile kataloogipuu ligipääsetavaks ka üle HTTP (HyperText Transfer Protocol), ehk siis läbi tavapärase veebibrauseri. Selle eeliseks on täielik platvormiagnostilisus ja lisandväärtuseks on veebibrauserisse lisatud liideste otse kasutatavus, nagu näiteks PDFi [Moz] või muu teksti-põhise dokumendi, näiteks Markdowni kuvajad [Vel]. Sarnaselt teistele ligipääsu protokollidele, kontrolliti ka selle kaudu ligipääsuõiguseid läbi LDAPi autentimise ja autoriseerimise. See lahendus oli realiseeritud Apache'i veebiserverisse liidestatud sihtotstarbeliselt programmeerimiskeeles Perl kirjutatud CGI-tüüpi (Common Gateway Interface) priviligeeritud rakendusena, kuid on tänaseks ettevõtte enda süsteemiadministraatorite hinnangul jäädavalt aegunud ja pole enam uuesti integreeritav. See aegunud rakendus täitis enda ülesandeid rohkem kui kümme aastat ja lõpetas töö juba 2021. aasta alguses. Samas vahendi aegumisega pole vajadus sellise lahenduse järgi kadunud, pigem ettevõtte hiljuti suurenenud kasvutempo tõttu on vajadus suurem kui varem.

1.3 Probleemi kirjeldus

Teiste ja varasemate lahenduste analüüsist tõstatub kaks tähtsat probleemi: platvormist sõltumatu universaalse adresseerimise süsteemi puudumine ja otse operatsioonisüsteemi vastu autentimist ja otse failisüsteemi vastu juurdepääsu kontrolli teostava mehhanismi

puudumine.

Platvormist sõltumatu universaalse adresseerimise lahendamiseks tuleb leida selline liides, millega saab olla kindel selle kasutatavuses erinevatel platvormidel ja et oleks nende vahel ka adresseerimise ühtsus. Liidese kasutatavuse ja otstarbeks sobivuse jaoks oleks kasulik, et adresseerimine või viitamine oleks mingil kujul keskne selle liidese olemusele. Sobivaks liideseks on näiteks HTTP, mis on kasutuses tänapäeva töölaua arvutites laialdaselt levinud veebibrauserites.

Otse operatsioonisüsteemi vastu autentimise ja otse failisüsteemi vastu juurdepääsu kontrolli teostamisel on mitu eelist. Otsekontrolli mudelil puudub alternatiivne vahesamm autentimise ja juurdepääsuõiguste info talletamisel, seega on kontrolli tulemus alati usaldusväärne ja puudub viivitus muudatuste rakendamisele. Sellise mudeliga on raske seadistada süsteemi tegema kontrolle valesti ja lihtsustub sellise failiserveri õigesti rakendamine.

2 Lahenduse teoreetilised alused

Järgnevalt kirjeldatakse lahenduse käigus esinevaid vahendeid ja mehhanisme üldise teoreetilise tausta mõistmiseks failiserverite liideste arendamisel.

2.1 Turvalisus

Kõige vankumatum mitte-funktsionaalne nõue on turvalisus — selle rakenduse kontekstis tähendab see peamiselt rangust pääsureeglite jälgimisel ja vastupidavust nii pahatahtlikule rünnakule kavatsusega pääseda ligi piiratud andmetele kui ka pahaaimamatule kasutajale. Seda ootust saab üldistada ka infoturbe kolmikuna, tuntud ka kui CIA triaad, mille osadeks on [And14]:

- konfidentsiaalsus, mis on privaatsusega seotud mõiste, kuid on vaid alamosa privaatsusest, sest määrab juurdepääsu võimalikkuse vaid reeglite alusel, mitte ei sõltu isiklikust huvist midagi konfidentsiaalseks pidada, nagu kehtib privaatsuse puhul;
- terviklikkus, mis on ootus teabe muutmisele ja kustutamisele rakendatud piirangutele ja reeglitele;
- käideldavus, mis on teabe soovi korral kätte saamise ootus, st infosüsteemi kasutamist ei piira teiste kasutajate, mh pahatahtlike kasutajate tegevus.

Veebiliidesel toimiva rakenduse kontekstis on lihtsaim lahendus nende ootuste täitmiseks võtta ette mõni tuntud infoturbealane standard, mida järgida juhisena. Veebirakenduste maailmas on OWASP (Open Web Application Security Project) üks suurimaid sihtasutusi, mis analüüsib hetkel levinud ja rohkem rünnatud haavatavusi [OWAa]. See asutus annab välja infoturbe standardit ASVS (Application Security Verification Standard), mis sisaldab üldistatud reeglistikku veebirakenduste turvaliseks programmeerimiseks ja tarnimiseks [OWAb]. Turbealaste omaduste kindlaks täitmiseks on eesmärgiks

käesoleva lahenduse valmistamisel järgida juhisena OWASP-i ASVS-i kolmanda taseme reegleid. Selle taseme reeglid on kõige piiravamad selles turbestandardis ja on hinnatud piisavaks näiteks kriitilisele infrastruktuurile ja sõjaväelisteks otstarveteks [OWAb].

2.2 POSIX-i failiõigused

POSIX-it toetavatel operatsioonisüsteemidel on failiõiguste ümber erinevaid standardiseeritud omadusi. Igal kasutajal on sellises süsteemis neli asja: kasutaja nimi, kasutaja arvuline ID, kodukataloog ja sisselogimisel avatav kest ning igal grupil on nimi, grupi arvuline ID ja kasutajate nimekiri, mis võimaldab ühel kasutajal olla rohkem kui ühes grupis samaaegselt [Ker10]. Failide õiguste peamine standardiseeritud mehhanism on faili omand, milles määratakse igale failile kolmes tüübis õigused, omanikule mõeldud õigused, faili grupile mõeldud õigused ja kõigile mõeldud õigused, kasutades kolme bitti, millega saab määrata õigust lugemiseks, kirjutamiseks ja käivitamiseks [Ker10]. Lisaks eelnevale on Linuxis kasutatav POSIX-i standardi laienduse mustandis spetsifitseeritud ACL-ide (Access Control Lists) süsteem, mis lubab lisada failidele ja kataloogidele ühe kasutaja või grupi kaupa kasutades sama kolme biti süsteemi, mis võimaldab kirjeldada keerulisemaid juurdepääsu reegleid, kui tavaline„omaniku-grupi-kõigi” õiguste mudel lubab [Ker10].

2.3 POSIX-i protsessimandaadid

POSIX-ile vastavates operatsioonisüsteemides nimetatakse töötavat programmi instantsi protsessiks, mida iseloomustavad näiteks binaari nimi, protsessi ID, jooksutamise koht, protsessi kasutaja ID, protsessi grupi ID ja protsessi lisandgrupid, millest kasutaja ja grupi ID ning lisandgrupid pärinevad protsessi käivitanud protsessilt, mis enamasti on võrdne protsessi alustaja kasutaja väärtustega [Ker10]. Vältimaks kõikide protsesside administraatorina jooksutamist, eksisteerib võimekus panna protsess alati käivituma

kindlates õigustes kasutades SUID (Set Owner User ID) märgendit binaaril, mille puhul protsess käivitatakse binaari faili omaniku kasutaja õigustes [Ker10]. Sellega saab võimaldada priviligeerimata kasutajatele süsteemi failides kontrollitud muudatusi, näiteks liigutades mängude edetabeli kasutaja „games” alla ja keelates teistel kirjutada, lubades nõnda ainult mängudel kirjutada edetabelitesse ja piirata sedasi edetabelite muutmist petmiseks. Protsessi mandaate saab muuta kahes olukorras: administraatori õigustest saab määrata vähem-priviligeeritud õigustesse ja SUID-i kasutamise korral jooksutava kasutaja ja SUID-iga saadud kasutajate vahel [Ker10].

3 Lahenduse kirjeldus

Antud peatükis kirjeldatakse lahenduse nõudeid, probleemide lahendusi ja implementatsiooni detaile.

3.1 Nõuded

Selleks, et valmistada ette lahendust, kirjeldati esmalt soovitud lahenduse omadusi. Siinse töö autor, juhendaja ja Cybernetica IT-talitus analüüsisid koos selleks varasema lahenduse omadusi. Omadusi kirjeldati funktsionaalsete ja mitte-funktsionaalsete nõuete vormis.

Funktsionaalsed nõuded on järgnevad:

1. Süsteem serveerib faile ja kataloogide kirjeldusi üle HTTP serverisiseste juurdepääsu reeglite alusel. Protokoll on implementeeritud piisava tasemeni, et garanteerida moodsate brauseritega töötamine.

2. Serverisiseseid juurdepääsu reegleid autentimiseks kontrollitakse kasutades PAM-i (Pluggable Authentication Modules) ja autoriseerimiseks kasutades failisüsteemi läbi Linuxi *kerneli*.

3. Vale autoriseerimise info esitamisel väljastatakse vastav veateade, mis ei võimalda määrata, milline osa sisendist oli vale.

4. Autentimise või autoriseeringu vea korral ei väljasta lahendus faili ega ka metaandmeid.

Mitte-funktsionaalsed nõuded on järgnevad:

1. Lahendus toimib moodsal Linuxi kernelil, alates versioonist 5.10 kuni värskeima versioonini, milleks on töö kirjutamise hetkel versioon 5.17.

2. Lahendus on vastupidav autentimise jõurünnete vastu.

3. Lahendus suudab serveerida minimaalselt viis edukat päringut sekundis.

4. Lahendus ei tarbi jõudeolekus üle ühe tuuma ega üle 500MiB mälu.

5. Lahendus ei tarbi faili vahendades rohkem operatiivmälu kui edastatav fail sisaldab täies mahus.

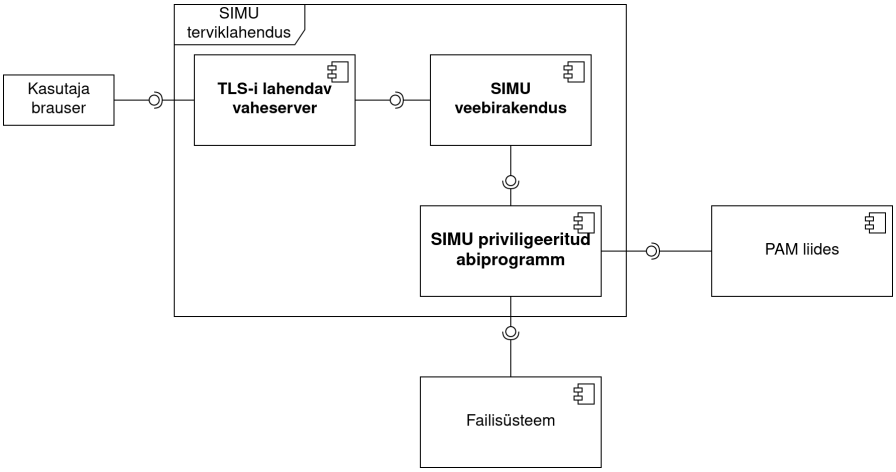
6. Lahendus vastab vähemalt OWASP-i ASVS-i kolmanda taseme nõuetele ja seda standardit kasutatakse arhitektuuriliste otsuste juhtimisel.

3.2 Teostus

Eelnevalt esitatud nõuete alusel loodi terviklahendus nimega SIMU[2]. Failiserveri lahendus koosneb kolmes osast: TLS-i (Transport Layer Security) rakendav vaheserver, SIMU veebirakendus ja SIMU priviligeeritud abiprogramm. SIMU rakenduse nime kasutatakse viidates korraga nii SIMU veebirakendusele kui ka SIMU priviligeeritud abiprogrammile, kuna need kaks komponenti jagavad palju koodi, on väga lähedalt seotud ja otstarbe täitmiseks lahutamatud. SIMU rakendus on testimise ja turvalisuse huvides tehtud kahes komponendis: veebirakendus serveerimaks HTTP protokolli ja privilegeeritud abiprogramm, mis võimaldab läbi lihtsa liidese autentida kasutaja ja avaldada ühe faili või kataloogi sisu. SIMU lahenduse arhitektuur on kujutatud joonisel 1 ja komponentide vahelist suhtlust kirjeldab järgnevusskeemi joonis 2.

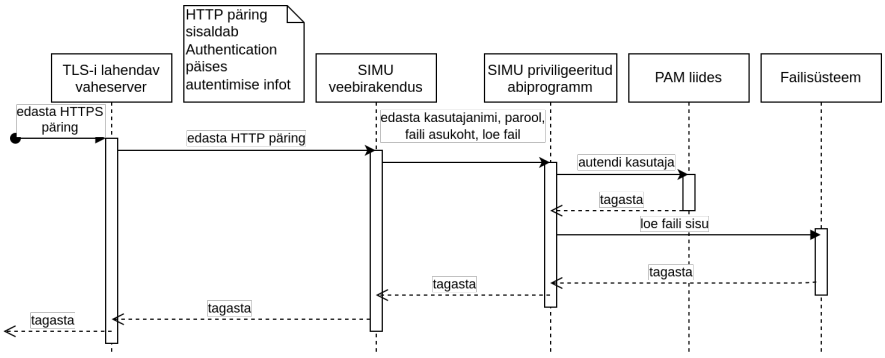
SIMU rakenduse kaheks jaotamise eelis turvalisuse tagamisel tuleneb sellest, et ainult abiprogramm peab töötama kõrgendatud õigustega, kuid samas on selle liides täielikult väljaspool avalikku ligipääsu ja võimaldab sisendina anda vaid kasutajanime, parooli, faili või kataloogi asukoha ja eeldatava vastuse tüübi. Sedasi töötleb vähemate õigustega komponent HTTP protokolli ise ja annab priviligeeritud abiprogrammile edasi vaid neli täpselt spetsifitseeritud sisendit. Tulenevalt HTTP protokolli mahukusest on selle

[2]Nimi on valitud juhuslikult lühidust ja hääldatavaust hinnates.



Joonis 1. SIMU lahenduse arhitektuur

tõlgendamine keerukas ja sellest tulenevalt on arendajal lihtne seda protokolli töödeldes kogemata vigu teha. Priviligeeritud abiprogrammis võivad sellised vead viia administraatori õigustes turvarikkeni. Lihtne liides annab eelise testimisel, kuna programm on väga sarnane käsule „su -c ’cat <faili asukoht>’ <kasutajanimi>”, kuid liigutab nende etteantavate muutujate liidese ainult sisendisse, vältimaks muutujate sisu protsessi tabelis nähtavaks muutumist.



Joonis 2. SIMU lahenduse järgnevusskeem

3.2.1 TLS-i lahendav vaheserver

TLS-i lahendav server on eraldiseisev SIMU rakendusest, kuna olemasolevad vaheserverid on süsteemiadministraatorile tuttavamad hallata ja võimekamad, kui oleks selle rakenduse osana võimalik luua mõistliku ajaga. SIMU lahenduse kasutaja brauser ühendub otse vaid vaheserverisse. Kliendi päringu sisu eeltöödeldakse ja vahendatakse pärast valideerimist SIMU veebirakendusele. Selline eraldatus võimaldab ka käia kaasas erinevate HTTP-i või TLS-i uuendustega, kuna sellised uuendused puudutavad vaid kliendi brauseri ja vaheserveri vahelist liidestust ega vaja SIMU rakenduse poolseid muudatusi. TLS-i protokolli implementeerimine on antud töö skoobist väljas, kuna tegemist on keerulise ja turvakriitilise protokolliga, millele on juba olemas laialt levinud lahendusi. Levinud vaheserverid on näiteks NGINX[3], Apache HTTP Server[4] või Caddy[5].

3.2.2 Veebirakendus

Veebirakendus on kirjutatud programmeerimiskeeles Rust, kasutades veebiraamistikku Actix. Raamistik on ehitatud Tokio koostööliste lõimede mudelile. Veebirakenduse peamine otstarve on eeltöödelda kasutaja päringut ja vahendada saadud autentimise info ja faili asukoht privilegeeritud abiprogrammile ning vahendada faili sisu või kataloogipuu päringu vastusena.

Veebirakendus serveerib vaid HTTP GET meetodiga päringuid, mis on mõeldud vaid andmete pärimiseks, ja standardi järgi peaks GET päring olema ohutu ja idempotentne, st päring ei tohiks teha muud peale andmete lugemise ja mitmekordne päring ei erine omadustelt ühekordsest päringust [Fie+99]. Veebirakendus kasutab autentimise info pärimiseks ja saamiseks *HTTP Basic* meetodit, millega antakse kasutajanimi ja parool päringu *Authorization* päises. Selle päise puudumisel saadab server kliendile

[3] `https://www.nginx.com/resources/glossary/nginx/`

[4] `https://httpd.apache.org/ABOUT_APACHE.html`

[5] `https://caddyserver.com/docs/`

teate, et kommunikatsiooniks on tarvis lisada sobiv autentimise info päringule. Sobiva päringu korral edastatakse selles sisaldunud autentimise info ja päritud faili asukoht privilegeeritud abiprogrammile, mis kontrollib kasutaja infot ja annab tema õigustele vastavalt vastuseks kas faili või kataloogi sisu või veateate faili puudumise või seotud õiguste puudumise kohta. Kataloogipuu ja veateadete lehed on stiliseeritavad veebilehe administraatori poolt, kuna nende esitlemisel kasutatakse veebilehe mallide süsteemi Handlebars.

Selle lähenemise puuduseks on *Authorization* päise brauseri-sisene käitlemine. Brauserite käitumine seoses kasutajanime ja parooli talletamisega brauseris pole standardiseeritud ja veebirakendusel puudub kontroll selle üle. Katsete tulemusel selgub, et nii Mozilla Firefox kui ka Google'i Chromiumi-baasil brauserid talletavad kasutajanime ja parooli kuni brauseri täieliku sulgemiseni. Seega brauser võib talletada seda parooli piiramatu aja ja veebirakendus ei saa nõuda uut kontrolli kinnitamaks, et veebibrauseri hetkene kasutaja teab seda informatsiooni, välistamaks näiteks arvuti kaotamisel kasutajanime ja parooli lekke ja rakenduse edasise kasutamise. Tulenevalt *Authorization* päise käitumise standardiseerimata olekust on keeruline ennustada kas ja kuidas võib see käitumine tulevikus muutuda.

3.2.3 Privilegeeritud abirakendus

Abirakendus on süsteemi alamosa, mille ülesanded jaotuvad järgmiselt: kasutaja andmete kontroll, tema õigustesse ümberasumine ja faili või kataloogipuu sisu edastamine. Abirakenduse priviligeeritus seisneb selle administraatori ehk Linuxi puhul kasutaja *root* õigustes töötamine. Selline priviligeeritus on saavutatav kas kogu SIMU rakendust administraatorina jooksutades või määrates kasutaja *root* abirakenduse binaari omanikuks ja määrates binaarile SUID õigusbitt, mis määrab sellest binaarist tekitatud protsessile kasutajaks binaari omaniku, ehk kavatsetud juhul kasutaja *root*. Nõnda on võimalik käivitada SIMU veebirakenduse osa vähempriviligeeritud kasutajana, aga kasutada priviligeeritud

abirakenduse kaudu talle endale kättesaamatuid vahendeid.

Kasutaja andmete kontroll seisneb serverilt saadud kasutajanime ja parooli kontrollimises vastu süsteemset autentimise moodulite kogumikku PAM, täpsemalt 'login' ehk sisselogimise teenuse vastu. PAM üldistab operatsioonisüsteemi autentimise liideseid võimaldamaks klientrakenduste kirjutajatel kasutada julgelt autentimisega seotud liideseid õigesti sõltumata operatsioonisüsteemist ja süsteemiadministraatori konfiguratsioonist [FR02]. See kontroll annab kindluse, et saadud info alusel oleks võimalik masinasse sisse logida ka väljaspool siinset süsteemi, näiteks SSH-ga, SMB-ga või füüsiliselt arvuti klaviatuuri taga olles. Selle kontrolli mitteläbimisel edastatakse veebirakendusele teade, et antud kasutaja pole autoriseeritud süsteemi kasutama ja see teade edastatakse kliendile. Kontrolli läbimisel alustab priviligeeritud komponent selle kasutaja õigustesse ümber asumist.

Kasutaja õiguste vahetamise algoritm on täpse järjekorraga, mis on välja selgitatud arendaja dokumentatsiooni alusel ja katsetuste käigus on see algoritm tehtud võimalikult lühikeseks. SIMU priviligeeritud abiprogrammis kasutatava õiguste vahetamise algoritmi sammud on järgnevad:

1. Programm leiab masinast sisendis soovitud nimega kasutaja UID (User Identifier), GID (Group Identifier) ja lisandgrupid.

2. Programm vahetab peamine grupp soovitud kasutaja grupiks kasutades eelnevalt leitud GID-d.

3. Programm tühjendab lisandgruppide nimekiri. Linuxil lisatakse lisandgruppide nimekirja tühjendamisel sinna protsessi peamine grupp tagasi.

4. Programm määrab eelnevalt leitud soovitud kasutaja lisandgrupid protsessi lisandgruppideks.

5. Programm määrab protsessi kasutajaks soovitud kasutaja, kasutades eelnevalt

leitud UID-d.

Peale viimast sammu on õigustesse ümberasumine lõppenud ja protsess võtnud täielikult omaks soovitud kasutaja õigused täpselt nii, nagu käitutakse kasutaja tavalisel sisse logimisel. Selline kasutaja õigustesse ümber asumine on pöördumatu käimas oleva protsessi piires. Koodinäide sellest protsessist käesoleva töö osana on nähtav joonisel 5.

Faili või kataloogipuu sisu edastamine on pärast kasutaja õigustesse asumist triviaalne ja seda teeb Linuxi tuum rakenduse eest, keelates ja lubades vastavalt ümber asutud kasutaja õigustele. Failid loetakse bait-baidilt standard-väljundisse. Kataloogid pakitakse rakenduse-spetsiifilisse binaarprotokolli, mille kodeerimise ja dekodeerimise lähtekoodi veebiserver ja abiprogramm jagavad. Binaarprotokoll on loodud teegi Bincode[6] abil, mis võimaldab Rustis kirjutatud andmestruktuurile tekitada mugavalt kodeerimise ja dekodeerimise mehhanismid. Kui abirakendus ei suuda leida soovitud teelt faili ega kataloogi või puuduvad vastavad õigused, siis tagastatakse veateade. Kui abirakendusele antud failitee ei viita ootuspäraselt failile, siis tagastatakse vastav veateade, et server saaks korrata päringut õige tüübiga.

[6] `https://github.com/bincode-org/bincode`

4 Testimine

Rakenduse loomise tähtsaks osaks on selle testimine nõuetele vastavuse osas kinnitamaks rakenduse vastavust ja omadusi nii käitumise kui ka jõudluse osas. SIMU rakendus vastab kõigile selles töös mainitud nõuetele peale ASVS-i kolmandale tasemele vastavuse, kuna selle testimise täielik läbiviimine ei mahu töö mahupiiridesse. Vähendamaks edasise arenduse käigus testimiskoormust kasutatakse integratsiooni teste, millega testitakse süsteemi omadusi automaatselt nõnda, et süsteemi kasutatakse mingis rakenduses ilma välise instrumenteerimiseta, võimalikult sarnaselt tegelikule tavakasutusele. Sedasi katsetatakse süsteemi toimimist täies mahus ja saavutatakse usaldus lahenduse toimimise suhtes automaatselt ilma täielikku katsekeskkonda kasutamata.

4.1 Koormustestid

Käesoleva töö osana loodud SIMU rakenduse koormustestide läbiviimiseks kasutatakse tööriista wrk [Glo]. Tegemist on moodsa mitmelõimelise tööriistaga sooritamaks jõudlusteste HTTP protokolli toetavate süsteemide vastu.

Koormusteste viidi läbi kasutades edukaid päringuid nii failide kui ka kataloogide vastu. Tulenevalt arhitektuuri omadustest, on neid tulemusi võrreldes võimalik eristada omadustest tulenevaid jõudluse aspekte. Tulemused on saavutatud testides protsessoril AMD Ryzen 5 3600, millel on 6 tuuma ja 12 lõime. Testitaval operatsioonisüsteemiks on vaikekonfiguratsioonis Debian 11.

Testides kataloogi sisu kuvamise kiirust, läbitakse HTTP päringu töötlus, alamprogrammi käivitamise, alamprogrammis autentimise ja õiguste vahetamise, kausta sisu veebiserverisse edastamise ja selle stiliseeritud veebilehte asetamise sammud. Ühe paralleelse kliendiga, st kõiki päringuid jadamisi esitades suudab server teenindada 130 päringut sekundis, keskmise latentsiga 7.68ms. Kaheteistkümne paralleelse kliendiga päringuid esitades suudab server teenindada 931 päringut sekundis, keskmise latentsiga

12.89ms, mille puhul saabub piirang protsessori ressursi puudusega. Sellest saab järeldada, et sellel masinal on abiprogrammi käivitamisel ülempiir umbes 900 korda sekundis ja et päringu latentsi alumine piir on 6ms, st sellest ei saa kiirem olla ükski edukas päring.

Testides suure faili laadimise kiirust, läbitakse kataloogi lugemisele sarnane protsess, kuid kataloogi sisu saatmise asemel saadetakse faile, mis on enamasti märkimisväärselt andmemahukam. Testitav fail on mahult 100 mebibitti. Ühe paralleelse kliendiga serveeritakse ühes sekundis 14.7 päringut, mis teeb andmete edastuskiiruseks umbkaudu 11.5 gigabitti sekundis. Kaheteistkümne paralleelse kliendiga serveeritakse 46.8 päringut sekundis, mis teeb andmete edastuskiiruseks umbes 36.6 gigabitti sekundis.

Testides serveri skaleeruvuse osas kasutati testimiseks kahe gibibaidist videofaili, mida esitati MPV[7] meediapleieriga otse läbi SIMU rakenduse. Joonisel 3 kujutatakse SIMU rakenduse komponente protsessinimistus kahe videot vaatava kliendiga. Antud väljavõttest selgub, et väljavõtte hetkel kasutatakse umbes 7852 kilobaiti mälu kliendi kohta ja umbes 0.75% ühest protsessori tuumast. Joonisel 4 kujutatakse sama olukorda kuue videot vaatava kliendiga. Sellest väljavõttest selgub, et ühe kliendi kohta kasutatakse umbes 4865 kilobaiti mälu ja umbes 0.35% ühest protsessori tuumast.

```
USER    PID    %CPU %MEM   VSZ  RSS TTY    STAT START   TIME COMMAND
nobody  20110  1.2  0.0 488652 9920 pts/0  Sl+  13:13   0:02 simu
liina   20333  0.3  0.0   3548 2860 pts/0  S+   13:14   0:00  \_ simu_suid_helper
keerup  20384  0.0  0.0   3548 2924 pts/0  S+   13:14   0:00  \_ simu_suid_helper
```

Joonis 3. Protsessinimistu väljavõte töötava SIMU lahenduse kasutamisel kahe kliendiga

[7] https://mpv.io/

```
USER    PID    %CPU %MEM   VSZ  RSS TTY    STAT START   TIME COMMAND
nobody  23276  0.7  0.0 757000 11512 pts/0 Sl+  13:30   0:04 simu
keerup  23351  0.0  0.0   3548  3016 pts/0 S+   13:30   0:00  \_ simu_suid_helper
liina   24510  0.3  0.0   3548  2948 pts/0 S+   13:38   0:00  \_ simu_suid_helper
liina   24535  0.2  0.0   3548  3020 pts/0 S+   13:38   0:00  \_ simu_suid_helper
liina   24901  0.3  0.0   3548  2932 pts/0 S+   13:38   0:00  \_ simu_suid_helper
liina   24786  0.3  0.0   3548  2932 pts/0 S+   13:38   0:00  \_ simu_suid_helper
liina   24825  0.3  0.0   3548  2832 pts/0 S+   13:38   0:00  \_ simu_suid_helper
```

Joonis 4. Protsessinimistu väljavõte töötava SIMU lahenduse kasutamisel kuue kliendiga

Nendest andmetest ei saa teha täpseid arvulisi järeldusi, kuid selgub, et skaleeruvus klientide kogusega on lineaarne või parem.

4.2 Integratsiooni testid

Tarkvara usaldusväärseks arendamiseks on tihti kasulik suuta automaatselt testida süsteemi või programmi enda terves ulatuseks koos selle võimalike interaktsioonide või liidestega. Selle lahenduse juures on integratsiooni testid implementeeritud läbi NixOSi süsteemi testide. NixOSi süsteemitestid võimaldavad kirjutada deklaratiivselt nii rakenduse ehitamise, kui ka virtuaalmasina konfigureerimise protsessi, kontrollides tugevalt selle loomisel väliseid sõltuvusi ja tehes selliste testide jooksutamise usaldusväärselt korratavaks sõltumata teste jooksutavast masinast [BD10]. Süsteemile vastav pakk ja integratsiooni test asuvad lähtekoodis failides „simu.nix” ja „test-vm.nix”. Praeguses seisus (mai 2022) integratsiooni test ei testi seda rakendust soovitusliku tarnega identselt, kuna SUID bitt priviligeeritud abiprogrammil on NixOSi süsteemi omadustest tulenevalt kujutatud tavatult, seega SUID bitt asendatakse testis terve SIMU rakenduse administraatorina jooksutamise abil.

Integratsiooni testis ehitatakse valmis kaks virtuaalmasinat: klient ja server. Serve-

risse paigaldatakse käesolev veebirakendus koos testimiseks tarviliku hulga failide ja kasutajakonto. Klientmasinas tehakse päringuid serveri vastu, mille puhul oodatakse nii sobiva kasutaja info korral edukat päringut, kui ka vale info või mitte-eksisteeriva faili korral veateadet. Kirjeldatud testi lähtekood on nähtav joonisel 6.

Kokkuvõte

Käesoleva töö käigus loodi failiserveri lahendus pääsemaks ligi sisule failiõiguseid austaval meetodil.

Juhendajale ja autorile teadaolevalt ei ole varasemat lahendust sellele probleemile ilma vajamata kompromisse või ümbritseva taristu loomist, mis mahult küündiks lähedale selle lahenduse omadustele. Teiste lahenduste puuduseks on enamasti puuduv failisüsteemi õiguste haldurina kasutamise meede. Alternatiivselt tuleks selle lahendamiseks konstrueerida siis mingi hulk failisüsteemi alusel loodud konfiguratsioonifaile, mis oleks eraldiseisev ja ei uueneks reaalajas.

Tänu lihtsakoelisusele ja kitsale otstarbele on loodud rakendus on lisaks kiiresti uuenemisele ka kergesti hallatav. Arhitektuuriliste otsuste mõjul on piiratud ründeala ja maksimaalselt kitsendatud kõrge ohuga programmi osi vähendamaks võimalusi arendaja vigadeks selle loomisel.

Lahendus on testitud kinnitamaks vastavust püstitatud nõuetele nii käitumiselt, kasutatavuselt kui ka jõudluselt. Testimise tulemuseks on lahenduse vastamine nii funktsionaalsetele kui ka mõistliku aja jooksul testitavatele mittefunktsionaalsetele nõuetele.

Täiendava arendusena on võimalik lisada rakendusse veelgi rohkem koostöölisi turvavahendeid, näiteks Linux seccomp [Edg15] või SELinux [LS01]. Lisaks on hetkeses lahenduses seni valideerimata muude operatsioonisüsteemide tugi väljaspool Linuxit ja puuduvad võimalused lisamaks muid HTTP-põhiseid autentimismehhanisme ja sessioonide haldamiseks.

Viidatud kirjandus

[And14] Jason Andress. The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice. Syngress, 2014.

[App] Apple. Network address formats and protocols on Mac. https://support.apple.com/guide/mac-help/network-address-formats-and-protocols-on-mac-mchlp1654/12.0/mac/12.0. (10.12.2021).

[BD10] S. van der Burg ja E. Dolstra. Automating System Tests Using Declarative Virtual Machines. *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)* (november 2010). https://edolstra.github.io/pubs/decvms-issre2010-submitted.pdf, lk. 181–190. https://doi.ieeecomputersociety.org/10.1109/ISSRE.2010.34.

[BS01] Daniel Barrett ja Richard E. Silverman. SSH, The Secure Shell: The Definitive Guide. O'Reilly, 2001.

[Cyb] Cybernetica. The architects of digital transformation. https://cyber.ee/company/our-difference/. (10.12.2021).

[Edg15] Jake Edge. A seccomp overview. https://lwn.net/Articles/656307/. (11.04.2022). September 2015.

[Fie+99] R. Fielding *et al. Hypertext Transfer Protocol – HTTP/1.1*. Tehniline raport. Juuni 1999. URL: https://doi.org/10.17487/rfc2616.

[FR02] Savio Fernandes ja K. L. M. Reddy. Securing Applications on Linux with PAM. *Linux J.* 2002.102 (oktoober 2002). https://www.linuxjournal.com/article/5940.

[Glo] Will Glozer. wrk - a HTTP benchmarking tool. https://github.com/wg/wrk/tree/a211dd5a7050b1f9e8a9870b95513060e72ac4a0. (11.04.2022).

[Ker10] Michael Kerrisk. The Linux Programming Interface. en. San Francisco, CA: No Starch Press, jaanuar 2010.

[LS01] Peter Loscocco ja Stephen Smalley. *Integrating Flexible Support for Security Policies into the Linux Operating System*. https://www.nsa.gov/portals/75/documents/resources/everyone/digital-media-center/publications/research-papers/flexible-support-for-security-policies-into-linux-feb2001-report.pdf. Veebruar 2001.

[Mic20] Microsoft. Common Internet File System (CIFS) Protocol. https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-CIFS/%5bMS-CIFS%5d.pdf. 2020.

[Moz] Mozilla. View PDF files in Firefox or choose another viewer. https://support.mozilla.org/en-US/kb/view-pdf-files-firefox-or-choose-another-viewer. (10.12.2021).

[OWAa] OWASP. About the OWASP Foundation. https://owasp.org/about/. (10.12.2021).

[OWAb] OWASP. OWASP Application Security Verification Standard. https://owasp.org/www-project-application-security-verification-standard/. (10.12.2021).

[TE03] Jay Ts ja David Eckstein Robert ja Collier-Brown. Using Samba, 2nd edition. https://www.samba.org/samba/docs/using_samba/ch01.html. O'Reilly, 2003.

[Vel] Simeon Velichkov. Markdown Viewer Google Chrome extension. https://chrome.google.com/webstore/detail/markdown-viewer/ckkdlimhmcjmikdlpkmbgfkaik (10.12.2021).

Lisad

I. Koodinäited

```
let pwent = unsafe { getpwnam(username.as_ptr()) }; // Find user
if pwent.is_null() {
    return -1; // libc is broken
}
let gid = unsafe { (*pwent).pw_gid };
let uid = unsafe { (*pwent).pw_uid };

unsafe {
    // This setgid comes first as otherwise current group is
    // later reinserted to supplemental groups.
    let ret = setgid(gid);
    if ret < 0 { return ret; }
    // Remove all current supplemental groups
    let ret = setgroups(0, std::ptr::null());
    if ret < 0 { return ret; }
    // Add all user's supplemental groups
    let ret = initgroups(username.as_ptr(), gid);
    if ret < 0 { return ret; }
    // Set to requested user ID
    let ret = setuid(uid);
    if ret < 0 { return ret; }
}
return 0;
```

Joonis 5. Priviligeeritud abiprogrammi õiguste vahetamise algoritmi implementatsioon

```
let
  pkgs = import <nixpkgs> {};
  simu = pkgs.callPackage ./simu.nix {};
  noGraphics = {
    virtualisation.graphics = false;
  };
in pkgs.nixosTest({
  name = "simu";

  nodes = {
    server = { config, pkgs, ... }: {
      imports = [ noGraphics ];
      networking.firewall.allowedTCPPorts = [ 8080 ];
      users = {
        mutableUsers = false;
        users.testaccount = {
          initialPassword = "testpassword";
          isNormalUser = true;
        };
      };

      systemd.services.simu = {
        serviceConfig = {
          ExecStart = "${simu}/bin/simu";
          Type = "simple";
          WorkingDirectory = "/data";
          User = "root";
        };
      };
      environment.systemPackages = [ simu pkgs.pam ];
    };
```

```
    client = { ... }: {
      imports = [ noGraphics ];
      environment.systemPackages = [ pkgs.curl ];
    };
  };

  testScript = ''
    # start client in parallel
    start_all()

    # prepare data for test
    server.succeed("mkdir /data")
    server.succeed("echo test > /data/test")
    server.succeed("chown testaccount:root /data/test")
    server.succeed("chmod 600 /data/test")

    # start service and wait until it's available
    server.succeed("systemctl start simu")
    server.wait_for_unit("simu")

    # attempt requests from the client vm
    client.succeed('curl --fail -o - ' \
        'testaccount:testpassword@server:8080/test')
    client.fail('curl --fail -o - ' \
        'testaccount:testpassword@server:8080/nonexistant')
    client.fail('curl --fail -o - ' \
        'notanaccount:testpassword@server:8080/nonexistant')
  '';
})
```

Joonis 6. NixOSil põhineva integratsioonitesti lähtekood

II. Lähtekood

Selle töö raames valminud terviklahenduse lähtekood on saadaval Githubis aadressil `http://github.com/kk-boop/simu`.

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Kalmer Keerup**,

*(*autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Kasutaja õigustes töötava failiserveri veebiliidese teostus,

   *(*lõputöö pealkiri)

   mille juhendajad on Tarmo Oja ja Heili Orav,

   *(*juhendaja nimi)

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Kalmer Keerup

10.05.2022