Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Kenno Kallastu

CTF-pi – CTF ülesanded IoT turvalisuse õpetamiseks

Bakalaureusetöö (12 EAP)

Arvutitehnika eriala

Juhendaja:

Alo Peets, MSc

Tartu 2025

Resümee/Abstract

CTF-pi – CTF ülesanded IoT turvalisuse õpetamiseks

Lõputöö tulemusena valmis 38 praktilist CTF stiilis küberturvalisuse harjutust Raspberry Pi-le. CTF-pi on vabavaraline platvorm, mis simuleerib IoT ökosüsteemi, keskendudes seal levinud tarkvarakomponentidele ja nende teadaolevatele turvanõrkustele. Töö seletab detailsemalt loodud ülesannete sisu kaheksast alamkategooriast, milleks on: üldiste teadmiste kontroll, nõrgad paroolid, nõrk krüpteering, füüsiline turve, nõrgad veebirakendused, võrgundus, aegunud tarkvara, ebaturvaline API.

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Märksõnad: CTF, Raspberry Pi, küberturvalisus, turvalisus, ülesanne

CTF-pi – CTF challenges to teach IoT security

For this thesis, a total of 38 practical CTF-style cybersecurity exercises were developed for the Raspberry Pi platform. CTF-pi is completely free and open source. It simulates an IoT ecosystem by utilizing commonly used software components and their known security vulnerabilities. This thesis will cover the making of challenges, which are from eight different categories: general knowledge, weak credentials, weak encryption, physical access, vulnerable web applications, networking, outdated software, insecure API.

CERCS: T120 Systems engineering, computer technology; P170 Computer science, numerical analysis, systems, control

Keywords: CTF, Raspberry Pi, cyber security, security, challenge

Sisukord

Lühendid ja mõisted

Raspi, Pi – Raspberry Pi, kompaktne arvutiplatvorm koos kõigi peamiste tavaarvuti juurde kuuluvate komponentidega.

IoT - *Internet of Things*, füüsiliste seadmete võrgustik, millesse on paigaldatud elektroonika, tarkvara, sensorid, täiturseadmed ning interneti ligipääs[1].

CTF – *Capture The Flag*, eetilistele häkkeritele mõeldud õppevahend, kus läbi tahtliku süsteeminõrkuse tuleb jõuda peidetud sõnumini ehk lipuni.

CTF-pi – lõputööks valminud CTF ülesannete kogum.

Host, hostima – *host, hosting*, informatsiooni allikana käituv seade[2], pakutav teenus võib olla veebilehe, andmebaasi jms näol. Antud töös peetakse hostiks Raspberry Pi-d.

Port – TCP/IP ja UDP võrkude loogilise ühenduse lõpp-punkt[2]. Erineval pordil asetseb erinev teenus, näiteks 80 on HTTP liikluse jaoks, 22 on SSH protokolli jaoks.

GPIO – *General-Purpose Input/Output,* on mikrokontrolleril tahtlikult kasutajale vabaks ning kasutatavaks jäetud viigud. GPIO viikusid juhitakse tarkvara kaudu[3].

API – *Application Programming Interface,* programmiliides, kus üks tarkvara saab teise tarkvara (või operatsioonisüsteemi) teenuseid kasutada, seda läbi määratud reeglistiku[2].

Dockerfile – fail Dockeri konteinerite koostamiseks. Faili sisuks on reeglistik, mille rea haaval töötlemise lõpptulemusena saab toimiva ja originaalse rakenduse keskkonna.

---

[1] https://sonaveeb.ee/search/unif/dlall/kfs/IoT/3/eng

[2] http://www.vallaste.ee/

[3] https://en.wikipedia.org/wiki/General-purpose_input/output

1. Sissejuhatus

Küberohud on viimasel ajal üha aktuaalsem teema, seda kinnitab USA mittetulundusühingu *Identity Theft Resource Center* (ITRC) aastaraport[2], mille kohaselt oli aastal 2023 ligi 73% rohkem andmelekkeid kui aasta 2021 tipp punktis. Küberohtude teadvustamiseks on üle maailma viidud läbi palju kampaaniaid, mille peamine sõnum on hoolikalt jälgida saabuvaid e-maile, kasutada tugevaid paroole, mitte vajutada tundmata linkidele, jms. Need soovitused on kahtlemata asjakohased, kuna statistika ja praktika näitavad, et suur osa küberrünnakutest saavad alguse just sellistest inimlikest eksimustest nagu kahtlastele linkidele klõpsamine või nõrkade paroolide kasutamine[3].

Küberohtude käsitlemisel keskendutakse sageli just eelnevalt nimetatud interneti või võrgupõhistele rünnakutele, kuid oluline on mõista, et ohuks võib olla ka füüsiline ligipääs seadmele. Igasugune järelvalveta arvuti, tehases juhtimiskonsool, avalikus ruumis IoT seade võib kujutada endast tõsist turvariski. Sellistele seadmetele füüsiline juurdepääs võib anda ründajale lihtsa ligipääsu privaatsele kui ka salastatud väärtuslikule informatsioonile[4]. Arvestades, et IoT-seadmete arv maailmas kasvab kiiresti, suureneb samas tempos ka nende rünnatavus[5]. Eriti haavatavad on turvamata ja regulaarselt uuendamata IoT-seadmed, mis kujutavad endast järjest tõsisemaid riske nii üksikisikute kui ka kriitilise taristu jaoks.

1.1 Töö eesmärk

Antud bakalaureusetöö käsitleb IoT seadmete küberturbe ohtusid. Selleks on tööks kasutatud Raspberry Pi 5-te: kompaktset seadet, mida saab kasutada nagu tavalist arvutit ja on võimekuse poolest võrreldav tavaarvutiga. Eriliseks ja eelistatuks muudab Pi tema kompaktsus ja odavus. Samuti on Pi-l head liidesed riistvaraga, eelkõige GPIO, kuhu on mugav ühendada erinevaid sensoreid, nuppe jms. Raspberry Pi vabavara kogukond on suur ja lai ning toetatud on paljud populaarsemad programmeerimiskeeled, muutes tema ka väga heaks IoT seadmeks kui ka tavaarvuti alternatiiviks[6].

Kuna tegu on populaarse valikuga nii hobikorras tegelejate kui ka firmade poolt, siis on laiem kasutajaskond vägagi hea sihtmärk küberkurjategijatele, seetõttu on ka väga oluline jälgida häid küberturbe tavasid.

Bakalaureusetöö annab IoT valdkonnast huvitatud või valdkonnaga aktiivselt tegeleval inimesel võimaluse ise häkker olla. Sihilikult nõrkade konfiguratsioonidega Raspberry Pi seadistus on võimalik alla laadida ja kasutajatel tuleb juhendi järgi süsteemi sisse murdes peidetud sõnumid leida. Selline praktiline läbi tegemine peaks näitama kasutajale, kui kaugele tegelikult ühe lihtsa mööda vaatamise tõttu võib ründaja jõuda, mis omakorda võiks õpetada kasutajatel sellistele vigadele oma projektides rohkem tähelepanu pöörama. Samuti võiks seade olla huvipakkuv ka küberturbe entusiastidele, pakkudes ülesandeid ka tavaarvuti nõrkuste kohta.

Järgnevas peatükis sukeldutakse sügavamale IoT olemusse, turvalisusesse ning selle vajalikkusesse. Seejärel käiakse läbi viis, kuidas küberturbe õppematerjal loodi ja kuidas igaüks saab sarnase projekti luua, eesmärgiga laiemale IoT kogukonnale paremat küberturvalisust treenida. Käiakse läbi ka valminud ülesannete lahendusideed ning viimaseks valminud projekti testiprotseduur.

2. Teoreetiline ülevaade

IoT (ingl *Internet of Things)* ehk asjade internet[4], on võrku ühendatud ja omavahel seotud ning suhtlema pandud seadmed. Kõige klassikalisem näide pärineb kodu automaatikast, kus tavaliselt on sensor, mis mõõdab päevavalgust, valvekaamera, programmeeritavad valgustid, automatiseeritud väravad jne. Töö toimub seadmelt seadmele, näiteks valguse sensor tuvastab valguse vähenemist ja teavitab sellest sissesõidu väravat, et öö saabudes värav sulgeda. Samuti leidub rohkelt rakendusi tehaseautomaatikasse, robootikasse ja muusse sarnasesse. IoT-d võib kohata ka õppeasutustes ning parem on üldsegi küsida: kus IoT-d pole[1]?

IoT arenduseks on variante mitmeid ning oleneb suuresti lõppeesmärgist, mida täpsemalt seade tegema peab, mis on tema maksumus, ülalpidamise kulu, jõudlus, riistvara liidesed jms. Samas on turul kättesaadavad ka rohkem hõlmavad ja üldisemad arendusplaadid, andes võimaluse enne päristoote disainimist prototüüpida oma lahendust. Muidugi pole keelatud ka arendusplaati rakendada produktsioonis, seda üsnagi kättesaadava maksumuse ning arvestatava jõudluse juures.

Kõige enam soovitatakse arenduseks: ESP32, STM32, Arduino ning Raspberry Pi tooteid[11, 12, 13]. Töö jaoks sai neist välja valitud just Raspberry Pi, täpsemalt mudel 5. Pi hiilgab just oma lihtsuse poolest, programmeerimiskeelena kasutatakse enim Pythonit[5], mitte C-d või Arduino programmeerimiskeelt, mis algajale esialgu keerulised võivad tunduda. Lisaks, erinevalt teistest, kasutab Pi ka operatsioonisüsteemi Raspberry Pi OS, mis võimaldab seadet kasutada kui tavalist Linuxil põhinevat arvutit.

Nii Pi lihtsus ning võimekus kui ka tema üldjuhul madal hind[14] muudavad IoT kättesaadavamaks algajale kui ka ettevõtetele, suurendades tõenäosust, et osa populatsioonist on vähemalt teadlik Raspberry Pi olemasolust või isegi juba omavad teda.

[4] https://sonaveeb.ee/search/unif/dlall/kfs/IoT/3/eng

[5] https://www.deepseadev.com/en/blog/programming-language-raspberry-pi/

2.1 IoT nõrkused

Tihti visatakse nalja IoT seadmete ning nende nõrkuste kohta: „*The S in IoT stands for Security*“, nalja idee on see, et lühendis IoT pole „S“ tähte, seega ta pole ka turvaline[27].

Tüüpiliseks põhjuseks IoT seadmete ebaturvalisusele on üldiselt kas: 1) küberturbele üldse mitte tähelepanu pööramine, olgu selleks näiteks finantsilised põhjused, kuna head küberturbe tavad nõuavad lisainvesteeringuid, või 2) inseneri enda teadmatusest[27]. Kuna IoT seadmed on mõeldud mingi kindla tööülesande täitmiseks, ehk jõudlus, suurus, tarkvara saavad olla vaid kindlate tingimustega, ei pruugi lihtsalt olla võimalus turvalisust seadmesse integreerida[27].

IoT ehk asjade internet ning automaatika on tänapäeval laialdaselt tuntud ja levinud mõisted. Kes meist ei sooviks nutitelefonist juhitavat robottolmuimejat, häälkäsklustele reageerivaid valgusteid või koguni isesõitvat autot? Tehnoloogia peamine eesmärk on muuta inimeste igapäevaelu lihtsamaks ja mugavamaks, kuid siinkohal tekib õigustatud küsimus – kui lihtsaks on mõistlik elu muuta?

Kuigi hääleassistendid nagu Amazon Alexa[6] on leidnud koha paljudes kodudes ning sulandunud sujuvalt igapäevasesse internetivõrku, võib mõne seadme, näiteks veekeedukannu, varustamine püsiva internetiühendusega tekitada juba kahtlusi. Kas tõepoolest on vajalik, et taoline lihtne kodumasin oleks ööpäevaringselt ja seitse päeva nädalas ühendatud internetti? Sellised küsimused suunavad meid mõtlema mitte ainult mugavuse, vaid ka turvalisuse ja privaatsuse üle.

Esmapilgul võib tunduda ühe veekannu häkkimine tähenduseta, kuid nüüdseks on ründaja juba esimese suure sammu teinud, ligipääsu turvamata seadmele saanud. Kõige lihtsamat sorti rünne, mida hetkel teha võimalik oleks kontrollimatult veekannu juhtida, võimalusel tema elektroonikat kahjustada. Sarnane rünne, küll mitte veekannude vastu, vaid Iraani tuumaprogrammi peatamiseks on tänapäevani üheks suurimaks ja keerukamaks küberoperatsiooniks nimetatud Stuxnet[29].

[6] https://alexa.amazon.com/

Suure tõenäosusega küll ründaja vaid seadme hävitamisega ei piirdu, potentsiaal jõuda ohvri, ehk seadme ostnud kliendi isiklike andmeteni või teiste seadmeteni on palju väärtuslikum. Sai mainitud veekannu interneti ühenduvust, kui see seade oleks hetkel ka internetis, siis on ründajal võimalus proovida külgrünnet (ingl *lateral movement*)[7], külgründe eesmärk on liikuda ühest vähema privileegiga seadmest mõnda teise kõrgema privileegiga seadmesse. Ründaja võib selleks kasutada võrgu skaneerimist ning avastada mõne teise haavatava seadme, vigase kohaliku võrgu seadistuse puhul on läbi veekannu võimalik ka ruuterisse kahjulikke pakette saata[31]. IoT seadmeid nagu IP kaamerad, ruuterid, printerid sihilikult rünnanud ja hiljem neid ummistusrünnete sooritamiseks kasutanud Mirai[32] on üks näidetest, mis ebaturvaliste IoT seadmetega juhtuda võib.

2.1.1 Mirai bottnett

Mirai on uss-tüüpi kahjurvara[33], ehk on programm, mis paljuneb ise, ilma kasutaja sekkumiseta ning levib peamiselt võrgu kaudu[8]. Kahjurvaraga nakatunud seadet võidakse kasutada rämpsposti saatmiseks, seadme pealt krabatakse tundlikku informatsiooni, võidakse avada ka tagauksi hiljem seadme peal edasi toimetamiseks, kuid ka koondada seadmed pahalase alla suure hulgana, moodustades seeläbi *botneti*[35]. Bottnett või ka robotvõrk[9] allub pahalaste käsklustele ning on vägagi efektiivne sooritamaks ummistusründeid (ingl *Distributed Denial of Service - DDoS*), kuna seadmed pärinevad geograafiliselt erinevatest asukohtadest on ummistusründe ohvril väga keeruline rünnakut peatada. Mirai bottneti suurim kasutus oligi just ummistusrünnete sooritamine.

Mirai bottneti märgid hakkasid ilmnema 2016. aasta augusti lõpul[37] ning ei läinud kaua kuni hakkasid toimuma esimesed rünnakud. Küberturbe teemalisi artikleid kirjutav KrebsonSecurity kirjutas oma postituses, et olevat pihta saanud suure ummistusründega,

[7] https://akit.cyber.ee/term/9376-lateral-movement-2

[8] https://akit.cyber.ee/term/553-worm

[9] https://akit.cyber.ee/term/117-botnet-1

andmemahtude poolest pea kaks korda suuremaga kui varasemalt. Seadmed ise olid vähese jõudlusega, kuid nende äärmiselt suur kogus oli see, mis ründe nii edukaks tegid[38].

Kohati oli Mirai kahjurvarasse nakatunud 200 000 – 300 000 IoT seadet, IP kaamerad, ruuterid, printerid jms[32]. Mirai paljunemisloogika ei ole keeruline, kahjurvara saatis suurel hulgal TCP SYN pakette kuni mõni seade sellele vastas. Vastuse saades kahjurvara hakkas proovima tuntud kasutajanimede ja paroolide kombinatsioone, eduka sisselogimise korral saadeti teade Mirai looja serverisse. Seejärel käivitus järgmine programm, mis tuvastas seadme arhitektuuri ja laadis alla tegeliku pahavara[33].

2.1.2 Hiina robotkoer

Unitree Go1 on Hiina päritoluga robotkoer, mis on varustatud kaamerate ja sensoritega ning kasutab tehisaru tuvastamaks inimesi ja on võimeline tema omanikuga kaasas käima ja takistustest hoiduma[36].

Uurijad avastasid robotis tagaukse, mis võimaldas igaühel veebi API kaudu nii roboti asukohta kui ka kaamera pilti vaadata, ilma autentimiseta[34]. Robotil oli ka Raspberry Pi kasutaja ja parool vaikesätetena, ning kui toote soetaja neid ei muutnud, oli ründajal võimalik ka robotit juhtida[34].

Pole teada, kas tegu oli tahtlikult paigaldatud tagauksega või oli see toote disainimisel tehtud viga. Tänaseks on tagauks Go1 robotist täielikult eemaldatud, seda Unitree enda väitel[34].

2.2 CTF

CTF (ingl *capture the flag*) ülesanded on küberturbes hea ja laialtlevinud viis praktiliste oskuste arendamiseks. CTF ülesande eesmärk on leida tahtlikult seadistatud turvanõrkus, see turvanõrkus ära kasutada ja seejärel jõutakse mingi peidetud lipuni, milleks üldjuhul on tekstifail. Tekstifaili sees on juhised kas edasisteks sammudeks või on seal kirjas vastus, tavaline kuju on näiteks CTF{salas6na123}. Kui CTFi lahendaja sellise tekstirea ehk lipu üles leiab, loetakse ülesanne sooritatuks. Sellise mängulisuse lisamine võib muidu igavana tunduva küberturbe õppimise muuta vägagi põnevaks ning tagasisidestavaks.

CTF tüüpi ülesannete kohta korraldatakse ka hulganisti turniire, kuhu eetilised häkkerid tulevad kohale või distantsilt sooritavad, eesmärgiga oma küberturvalisuse oskused proovile panna. Olenevalt reeglitest lahendatakse ülesandeid tiimides ning vahetevahel ka üksinda. Võidab see, kellel kõige rohkem punkte või lippe on kogutud. Selliseid üritusi korraldatakse ka Eestis, näiteks üks selline on Cyber Battle, mida korraldatakse igal aastal[15].

Lisaks reaalselt toimuvatele üritustele on iga kell võimalik lahendada harjutusi iseseisvalt veebiplatvormidel, kus eksisteerivad globaalsed edetabelid ning iga edukas sooritus tõstab kasutaja taset ning avab uusi saavutusi. Mõned suurimad sellise teenuse pakkujad on: https://tryhackme.com/, https://www.hackthebox.com/, https://overthewire.org/.

2.3 Võrdlus olemasolevate lahendustega

Eristuste tegemiseks on loodud alljärgnev Tabel 1. Tabelis on neli erinevat CTF ülesande pakkujat, kaasaarvatud lõputööks loodud CTF-pi. Veebipõhised lahendused on TryHackMe, OverTheWire. Allalaetavate virtuaalmasinate lahenduse pakkujaks on valitud VulnHub[16]. Tabeli koostamiseks on tutvutud kõigi pakkujate sisuga ning antud lõpphinnang ca kümne erineva ülesande järgi.

Tabel 1. CTF-pi võrdlus alternatiividega.

	Raspberry Pi CTF[10]	**VulnHub**[11]	**TryHackMe**[12]	**OverTheWire**[13]
Tasuline	Ei, avatud lähtekood, kuid riistvara maksab	Ei	Preemium avab rohkem materjale ning avab kiiremad masinad ning mugavuse[14]	Ei
Võrguühendus vajalik	Vajadusel saab ilma	Vajadusel saab ka ilma	Jah	Jah
Salvestusruumi kasutus	Keskmine, kujutised ja teegid	Suur, iga virtuaalmasin on eraldi ülesanne	Puudub, tasuta kasutajatel vaja oma tööriistu	Madal, tööriistad
Riistvara ülesanded	Jah	Ei	Ei	Ei
Ülesannete hulk	Väike (esialgu)	Suur	Väga suur	Pigem suur
Ülesande juhendid	Jah	Pigem ei	*Walkthrough* tüüpi ülesannetel	Pigem ei
Platvormil avatud lähtekood	Jah	.ova Faili sisu saab vaadata	Ei	Ei
Ise koostatavad ülesanded	Jah	Jah	Jah	Ei
Punktid, edetabelid	Ei	Ei	Jah	Jah, lisaseadistusega[15]
Aktiivsed uuendused	Pigem jah	Ei	Jah	Ei
Muudetavus	Väga suur	Suur	Ei	Ei

[10] https://github.com/kennolot/CTF-pi

[11] https://www.vulnhub.com/

[12] https://tryhackme.com/

[13] https://overthewire.org/wargames/

[14] https://tryhackme.com/pricing

[15] https://overthewire.org/information/wechall.html

Enda valduses	Jah	Jah	Ei	Ei
Käivituskiirus	Kiire	Keskmine	Tasuta versioonil aeglane	Kiireim
Ülesannete endi platvorm	Linux	Linux või Windows	Linux või Windows	Linux

CTF-pi – on parim valik kui kasutaja eelistab seadmega palju ringi liikuda ja ise ülesandeid hostida. Odavuse mõistes on see samuti parim valik (juhul kui kasutajal juba arvutit pole), 11.mai 2025 seisuga saab Farnellist[16] Raspberry Pi mudel 3B umbes 30 euroga. Kui kasutaja soovib ise rohkem süvitsi toimuvasse sisse minna on CTF-pi jällegi võitja, kuna kõigi ülesannete käivitusprotseduur on avalikult GitHubis. Samuti on CTF-pi ainus, millel saab lahendada riistvara ülesandeid. Üldiselt sobib CTF-pi kasutajale, kes soovivad täielikku kontrolli oma keskkonna üle.

VulnHub – on hea valik Raspberry Pi puudumisel, kuid seevastu arvuti olemasolul. VulnHubi peamine tugevus võrreldes CTF-pi'ga on tema tunduvalt suurem ülesannete hulk. VulnHubis võib aimata olevat 700 erinevat ülesannet, mis on kommuuni poolt talle aastate jooksul koostatud.

TryHackMe – tõenäoliselt parim kasutajale, kes ise ei oska või ei taha ise asju üles seada, vaid tahab kiiresti ülesannetega pihta hakata. Mugavuse ja viimistletuse poolest on TryHackMe selge võitja. Kasutajad saavad kõike CTF seonduvat teha kasutades ainult oma veebibrauserit. Samuti on TryHackMe kommuun pakutud neljast suurim ning vahetevahel toimuvad ka päris auhindadele võistlused. Seevastu tuleks märkida, et tema kvaliteedi ja mugavuse jaoks oleks tarvis igakuine tellimus aktiveerida, tasuta mudelil on aeglasemad käivitusajad ning osad ülesanded pole ligipääsetavad.

OverTheWire – võib lugeda TryHackMe ja VulnHubi vahepealseks. OverTheWire pakkuja hostib ise ülesandeid, mis on kättesaadavad SSH ühendusega. Alla laadida muud peale SSH kliendi ei tule, kuigi mõne ülesande lahendamiseks on uut tarkvara vaja installida enda

[16] https://ee.farnell.com/raspberry-pi/raspberrypi3-modb-1gb/sbc-raspberry-pi-3-mod-b-1gb-ram/dp/2525225

arvutisse. Otseselt lahendusi ei paku, kuid aastatega on palju videomaterjale ning blogisid kirjutatud kasutajate poolt. Hea valik kui ei soovita maksta, kuid samuti ka mitte ise ülesandeid hostida ja seadistusi teha.

Tabeli, ega ka arutelu tulemustele tuginedes ei saa tuua välja kindlat võitjat, mis igale kasutajale hästi sobiks. Parim platvorm langeb väga kasutajate endi peale. Näiteks on kasutajaid, kelle rahalised võimekused on väiksed ka Raspberry Pi ostmiseks. Seevastu võib olla ka kasutajaid, kelle internetiühendus jääb kesiseks ja eelistab võrguühenduseta lahendada.

Kokkuvõtlikult võib öelda, et CTF-pi on parim valik kasutajale, kes soovib täielikku kontrolli oma õpikeskkonna üle, on valmis ise ülesandeid seadistama ning eelistab liikuvust ja riistvarapõhiseid harjutusi. VulnHub sobib hästi neile, kel Raspberry Pi puudub, kuid on olemas arvuti ning soov laiema ülesannete valiku järele. TryHackMe on parim valik kasutajatele, kes eelistavad mugavust ja kiiret alustamist ilma tehniliste seadistusteta, olles siiski seotud kuutasuga. OverTheWire jääb funktsionaalsuselt nende vahele, pakkudes tasuta ligipääsu ülesannetele ilma vajaduseta neid ise hostida, sobides hästi õppuritele, kes ei soovi maksta ega tehniliste detailidega süvitsi minna.

3. Kasutatud töövahendid

Töö viidi läbi Raspberry Pi platvormil, kasutades ülesannete käivitamiseks võimalikult palju Dockerit. Kõik loodud ülesanded koos vastavate juhenditega on avalikult kättesaadavad GitHubis. Riistvarapõhiste ülesannete teostamiseks kasutati lisaks Raspberry Pi Pico mikrokontrollerit, mõningaid ühenduskaableid ning multimeetrit.

3.1 Raspberry Pi 5

Töö jaoks kasutati RaspBerry Pi 5 8GB mudelit koos korpuse ja jahutusega. Seade ise ilmus aastal 2023 ja tema tehnilised näitajad on järgmised [8]:

- Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 protsessor,
- VideoCore VII GPU,
- HDMI võimalus,
- LPDDR4X-4267 SDRAM 8GB (olenevalt mudelist),
- 802.11ac Wi-Fi,
- Bluetooth 5.0 / Bluetooth Low Energy (BLE),
- MicroSD kaarti ava,
- 2 × USB 3.0,
- 2 × USB 2.0,
- Gigabit Ethernet,
- 2 × 4-lane MIPI camera/display transceivers,
- PCIe 2.0 x1 interface.

3.2 Raspberry Pi Pico

Riistvara turvaaukude ärakasutamiseks kasutati Raspberry Pi Pico 1-te RP2040 mikrokontrolleriga. Micro-USB B port võimaldab Pico kaudu suhelda Raspberry Pi GPIO

viikude ning peaarvuti vahel[18]. Tema programmeerimine toimus Pythonil põhineval MicroPythonil[17], mis on spetsiaalselt mikroprotsessorite programmeerimiseks mõeldud keel.

3.3 Docker

Töös on rohkesti kasutatud Dockeri[18] ning alamtööriista *Docker Compose*[19] võimalusi. Docker võimaldab luua hosti süsteemist eraldiseisvaid konteinereid, lubades konteineril, mis töötab arvutis number 1, töötada ka arvutis number 2[19]. Iga kord kui konteiner käivitatakse, läbib Docker kompileerimis protsessi, lugedes juhised, kas Dockerfile-st ja/või *compose.yaml* failist või juba olemasolevast kujutisest (ingl *image*), lokaalse keskkonna sõltuvused või eripärad enam rolli ei mängi, sest sõltuvused (ingl *dependencies*) on määratud Dockeri koodiga. Kõik see kiirendab märkimisväärselt tarkvara valmimist ning vähendab hilisemat veahaldust, kus kõik saavad sama tarkvara jooksutada, mis originaalne autor. Docker on vägagi tuntud tööriist arendajate ja *DevOps* inseneride seas[20].

Küberturbe kohapealt tähendab keskkond, mis on isoleeritud hostist, et nõrk tarkvara ei jookse otse riistvara pealt (ingl *bare metal*), vaid oma eraldi konteineris, kuigi seeläbi võimaldades hostil ja konteineril faile jagada ning porte ühendada[20].

Manuaalselt käskude sisestamine konteineri allalaadimiseks, seejärel ehitamiseks ning seejärel veel käivitamiseks koos lisa argumentidega võib muutuda kasutajale tülikaks ja võibolla isegi arusaamatuks. Töö CTF ülesande käivitamine peaks olema kasutajale hästi lihtne, seega parandamaks eelnevat olukorda, oleks mõistlik kasutada Dockerfile võimalust. Dockerfile'i sisse kirjutatud koodiga on võimalik kujutis juba automaatselt ise luua, kuid Dockeri *build* ja *run* käsk tulevad ikka käsitsi sisestada[21, 22]. Samuti iga CTF ülesande eripärast võib käskude

[17] https://micropython.org/

[18] https://www.docker.com/

[19] https://docs.docker.com/compose/

[20] https://sematext.com/glossary/docker/

vahel tekkida erinevusi, ehk ei ole ühtlast mustrit, mida kasutaja meeles saaks pidada. Samuti võivad lisaargumendid ja muu ära reeta osa lahendusest. Näiteks tuleb kasutajal port avalikustada *run* käsus, aga tegelik ülesanne ootabki lahenduseks just selle pordi leidmist. Lisaks toob omakorda keerukust juurde mitme konteineri korraga jooksutamine ning nende omavaheline koostöö.

Ühtlase käskude mustri kui ka mitme konteineri jooksutamise lihtsustamiseks sai CTF ülesannete jaoks kasutusele võetud *Docker Compose*. *Docker Compose* on tavalise Dockeri installatsiooniga kaasaskäiv lisatööriist, mis kõige paremini paistab silma mitme konteineri jooksutamisel, kuid samas saab ehitamise ja jooksutamise ning kõik muu vajaliku sooritada ühe *docker compose up* käsuga. Kogu jooksva süsteemi saab kinni panna ka ühe käsuga: *docker compose down*. *Compose* kasutamine toob ainult vajaduse kasutada *compose.yaml* faili, mille sees on jooksutatavad teenused, nende kujutise e *image* lähtekoht, pordid, köide (ingl *volume*) jt[23].

3.4 GitHub

CTF-pi on täielikult avatud lähtekoodiga ning paigutades projekti GitHubi, saab iga huviline oma harus uue ülesande luua, seejärel küsida haru lisamist põhiharusse (ingl main branch). See on ideaalne, sest ei piira projekti ainult ühe isiku ehk autori külge ning projekti populaarsemaks saamisel võimaldab ülesannete koguse märkimisväärset kasvu.

GitHub[21] on üks tuntumaid pilvepõhiseid tarkvaraarendus platvorme, kasutajad saavad oma koodi üles laadida repositooriumitesse, kus teised arendajad seda endale laadida saavad, kuid erinevalt tavalisest faili üleslaadimis teenusest saab igaüks ka koodi muudatusi teha ning vigasid tõstatada.

Tehnoloogias suure tõenäosusega ei näe kõiki probleeme ette, samuti ei saa ette näha ka tulevikku, mis uued riistvarad võivad tekkida või mis teek kuskilt ära kaob. GitHubis *Issues*

[21] https://github.com/

sektsioonis saavad kasutajad probleemide korral teema algatada, misjärel arendaja vea kinnitab ja ära parandab.

3.5 Markdown

CTF-pi lähtekoodi hoiustamiseks valiti platvormiks GitHub, mis toetab tekstide vormindamiseks peamiselt Markdown keelt. Arvestades, et GitHubi Markdown on hästi dokumenteeritud[28] ja laialdaselt kasutusel, osutus see parimaks valikuks ka töö juhendite ja dokumentatsiooni koostamiseks.

Markdown on tasuta ja avatud lähtekoodiga märgistuskeel faililaiendiga .md, mida saab kasutada dokumentide, e-kirjade, kui ka märkmete loomiseks. See märgistuskeel toimib igal tuntud platvormil, nagu Windows, Linux, kui ka Android[24]. Markdowni populaarsust toetab ka selle lihtsus ja laialdane tugi – lisaks GitHubile kasutatakse seda näiteks ka Redditis[24]. Tänu kasutusmugavusele ja platvormiülesele ühilduvusele on Markdownil suur potentsiaalne kasutajaskond, mis omakorda vähendab tõenäosust, et kasutajad loobuvad CTF-pi kasutamisest tehniliste piirangute tõttu.

GitHub võimaldab vaikimisi visualiseerida Markdowni projekti lehel, kui fail on nimega *README.md*. See tähendab, et ülesannete lahendajal pole vaja eraldi juhendit avada, GitHub laeb kogu stiili ja pildid automaatselt sisse ja kuvab selle kasutajatele.

GitHubi Markdown toetab HTMLi `<details>` elementi, millega saab lahenduste ning vihjete sektsiooni peita, ning nähtavaks teha ainult peale klikkides. Samuti on ära kasutatud Markdowni pealkirjade kui ka koodiploki sektsioone, muutes jaotuse arusaadavaks ning lubades koodijuppide kopeerimist.

3.6 Raspberry Pi Connect

Põhiline ligipääs Raspberry Pi arenduseseks toimus kasutades Raspberry enda loojate tarkvara Raspberry Pi Connect. Seadistus on lihtne, tuleb installida Pi-le *rpi-connect*, siis luua keskne kasutaja, Raspberry Pi peal autentida, seejärel veebis klientarvutil autentida[22].

Tulemuseks on võimalik veebibrauserist sooritada ekraanijagamist, kui ka *remote shell* abil käske Pi-le anda, likvideerides vajaduse füüsilise klaviatuuri, ekraani ja hiire ühenduse jaoks. Connecti või muu sarnase kasutamine on soovitatav ka CTF-pi ülesannete lahendajal.

Kuigi 9. mai 2025 seisuga on Raspberry Pi Connect veel beetaversioonis, ei kohatud töö käigus ühtegi tehnilist probleemi.

3.7 Töövahendite koosmõju

Töö käigus kasutati mitmesuguseid töövahendeid, kuid põhiliselt keskendudes Raspberry Pi platvormile ja konteinerpõhisele ülesannete haldusele Dockeriga. Ülesanded loodi Raspberry Pi 5 seadmel ning nende töökindluse ja kaasaskantavuse tagamiseks kasutati *Docker Compose'i*. Riistvarapõhiste ülesannete jaoks kaasati Raspberry Pi Pico mikrokontroller koos vajalike komponentidega. Kõik loodud ülesanded ja juhendid on avaldatud GitHubis, kus neid hoitakse avatud lähtekoodina, võimaldades kogukonnal ülesandeid täiendada ja vigasid parandada. Dokumentatsioon koostati Markdowni abil, mis tagab laialdase platvormitoe ja kasutusmugavuse. Lisaks kasutati Raspberry Pi Connect teenust, mis võimaldas ligipääsu Raspberry Pi-le kaugühenduse teel, lihtsustades nii arendust kui ka ülesannete lahendamist. Kombineerituna moodustavad need tööriistad paindliku, skaleeruva ja kogukonnasõbraliku arenduskeskkonna.

[22] https://www.raspberrypi.com/documentation/services/connect.html

4. Õppematerjalide loomine

Töö eesmärgiks oli teha õppematerjalid küberturbe tavade kinnistamiseks. Selleks loodi 8 kategooriat CTF ülesandeid, mis põhinevad kõige sagedamini esinevatest IoT seadme nõrkustest ja vigadest[9]. Kõik ülesanded on jagatud eraldi moodulitesse ning on eraldiseisvad teistest. Ülesandeid on oodatud lahendama järjest, alustades Linuxi käsurea harjutamisest kuni Dockeri tutvustamiseni, et edaspidi probleeme vältida. Ülejäänud kategooriad katavad nii võrguturvet, krüpteeringut, tarkvara nõrkusi, füüsilist turvet. Kogenud kasutajatele on loodud tabel, kust on võimalik valida meelepärane ülesanne. Samuti on iga mooduli juures ka juhend ning võimalusel lisatud ülesande oodatud lahenduse kood, kõik on kättesaadav lihtsasti leitavate failidena GitHub keskkonnas[23].

4.1 Juhendi nõuded

- Juhendid peavad olema inglise keeles, et tagada projekti rahvusvahelisus.
- Juhendid peavad olema jagatud moodulitesse ning olema eraldiseisvad teistest ülesannetest.
- Juhendil peaks olema pealkiri, stsenaariumi kirjeldus, eeltingimused, eesmärk ning olenevalt ülesandest kas vihjete plokk või lahendus.
- Juhendid peaksid olema arusaadavad ka küberturbega vähem kokku puutunud inimesele.
- Juhendil peab olema juhis ülesande käivitamiseks ning hiljem peatamiseks.
- Juhendil peab olema ülesande raskusaste märgitud.
- Peab olema eraldi juhend õpetamaks huvilisi ise ülesandeid koostama.

[23] https://github.com/kennolot/CTF-pi

4.2 Struktuur

CTF-pi iga ülesanne asub omale määratud eraldiseisvas repositooriumi kaustas. Juurkataloogis on jaotatud ülesanded oma kategooriatesse, näiteks krüpteeringu ülesanded kategooriasse „2.weak_encryption“ või aegunud tarkvara kategooriasse „6.outdated_software“. Kasutatud on nummerdatud loendust ning tühikuvaba teksti, tekitades korrektse ülesannete järjestuse ning lubades kergesti juba Raspberry Pi peale kloonitud repositooriumis tabeldusklahviga (ingl *tab*) ringi liikuda.

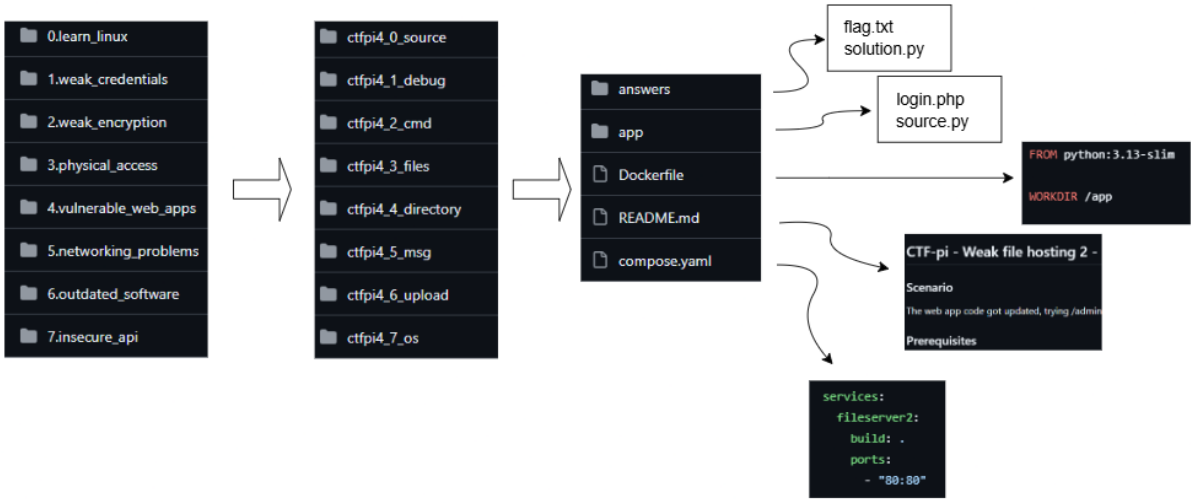
Peale juurkaustast meelepärase kategooria valimist avaneb ülesannete endi kataloog, iga ülesanne on märgistatud unikaalse nummerduse ning nimega kujul: kategooria number, näiteks 4, ülesande unikaalne number (id), lühike nimi.

Seejärel ollakse jõudnud päris ülesande sisuni. See on koht, kuhu kasutaja peab enda Raspberry Pi terminalis liikuma ning kust käivitatakse ülesanne *docker compose* käsuga. Sisu koosneb *answers* kataloogist, mis sisaldab „flag.txt“ faili, lahenduskoodi, kuid vajadusel ka abistavaid või ajutisi faile. *Answers* sisu ei ole mõeldud eelnevalt vaatamiseks kuna reedab ära lahenduse või vastuse, kataloog on mõeldud vaatamiseks vaid peale iseseisvalt lipu leidmist. Kui peaks selguma et vastused ei klapi, siis tuleks kasutajal edasi otsida, et lipud samastuksid.

Järgmine kataloog sisaldab lähtekoodi, üldiselt *app/* või ka *src/*. Selle teekond antakse ka ette Dockerile ning on vajalik alusprogrammi käivitamiseks. Enamjaolt asetseb siin Pythoni skript, kuid on ka juhtumeid, kus kasutatakse PHP kui ka Bashi koodi. Lähtekoodi vaatamine on rangelt määratud juhendi poolt, on ülesandeid, kus on kindlasti vaja lähtekoodi vaadata, kuid ka ülesandeid, kus selle vaatamine pole lubatud.

Järgnevad Dockerfile ning *compose.yaml* failid on mõeldud ülesande ehitamiseks ning käivitamiseks. Nende failide sisu vaatamine on peaaegu alati keelatud, kuna reedavad avatud pordid, tarkvara versioonid või ka peidetud lipufaili asukoha. Muidugi ei ole võimalik kasutaja piilumist piirata, kuna kõik toimub lokaalse seadme peal ning reaalse turvalisuse vaatepunktist on väga oluline enne tundmatu faili käivitamist tema sisus veenduda. CTF-pi põhi *README.md* failis on ka hoiatus toodud, et kui on soov Dockerfile või *compose.yaml* sisus veenduda, tuleks arvestada ülesande lahenduse teadasaamisega.

Viimasena tuleb tõenäoliselt kõige olulisem fail, *README.md*. Selle lugemine on rangelt soovituslik, kuna toob välja kõik eeltingimused enne ülesande käivitamist, annab kätte esialgsed juhised, kust ülesannet lahendama asuda, kuidas ülesanne käivitada ja hiljem peatada ning lõpuks toob detailse lahenduskäigu või mitmed vihjeid lahendaja abistamiseks. Kogu eelnevalt kirjeldatud struktuuri aitab visualiseerida Joonis 1.



Joonis 1. CTF-pi näidisstruktuur.

Ülesannete eripäradest tulenevalt on olukordi, kus ülesannet ei olegi Dockeriga võimalik ehitada, selliseks on näiteks riistvara ülesanded. Sellisel juhul on kasutajatele antud juhised, mis teegid endale laadida ning mis skriptid manuaalselt käivitada.

4.3 Ülesannete lühikirjeldus

CTF-pi koosneb kaheksa kategooria ülesannetest. Kategooriate valikul on kasutatud mitmeid veebi artikleid enam levinud IoT turvavigagade kohta[9, 25, 26] ning lisaks meenutades arvutitehnika eriala erinevate ainete loodud lahendustele ja sealt potentsiaalselt tulenevatele turvavigadele. Samuti on analüüsitud mitte IoT spetsiifilisi küberrünnakuid[3, 5]. Ülesannete ideede jaoks sai vähesel määral kasutatud tehisaru ja Tartu Ülikooli andmeturve kursuse materjale.

4.3.1 Üldised Linuxi teadmised

Peatüki eesmärk on kõik lahendajad viia nö ühele lehele oskuste mõttes. Esimeses ülesandes tehakse selgeks käsud *cd, ls, find* ja *grep,* mille eduka kasutamise korral on kasutaja leidnud saja tekstifaili seest rekursiivselt peidetud lipu.

Teises ülesandes tuleb kasutajatel mitme eksitava ja peidetud faili seest Linuxi terminaliga, kui ka GitHub repositooriumist leida lipp.

Kolmandas ülesandes meenutatakse kasutajale Pythonit, kuna Python on peamine programmeerimiskeel CTF-pi puhul (83.7%) ning valdav osa programmeerimisülesannetest sooritatakse Pythonis. Ülesandes genereeritakse ennekõike igaühele erinev räsi, seejärel tuleb kasutajal avastada räsi sisse peidetud kodeeritud sõnum, kirjutades selleks enda Pythoni skript.

Neljandas ülesandes võetakse ette Docker. Umbes 60-70% ülesannetest on Dockeril põhinevad, seetõttu on ka suur osa tutvustavat osa just Dockerile pühendatud. Ülesandes tuleb jooksvast Dockeri konteinerist kätte saada peidetud fail. Eesmärgi saavutamiseks tuleb uurida *interactive shell* võimalust. Ülesande näidislahendust kujutab Joonis 2.

```
ctfpi@raspberrypi:~/CTF-pi/0.learn_linux/ctfpi0_3_dckrdbg $ docker run --rm -it
my-ctfpi-image /bin/sh
/ #
/ # hostname
ad767d1d7144
/ # ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
/ # ls /tmp
flag.txt
/ #
```

Joonis 2. Näidislahendus Dockerit sissejuhatavale ülesandele.

Viiendas ülesandes tutvustatakse *docker compose* käsku ja tutvustatakse projekti ehitamiseks vajalike failide sisu nagu Dockerfile ja *compose.yaml*.

Kuuendas ülesandes tuleb eelnevalt õpitud Dockeri teadmisi rakendada ja ise täiendada programmi. Kasutajate eest on tehtud Dockerfile ja *compose.yaml*, samuti on ka üks pool Pythoni skriptist kirjutatud. Ülesandes on üks Dockeri konteiner saatja ja teine konteiner vastuvõtja, saatja kirjutab pidevalt lippu tekstifaili ning kasutajal tuleb implementeerida teine pool, mis seda lippu vastu võtab.

Seitsmendas ülesandes tutvustatakse *race condition* olukorda, kuna võis täiesti juhtuda, et ka kuuendas ülesandes veaohtlik kood kirjutati ise seda teadmata. Kasutajatele on ette loodud kood, kus tekib kahe konteineri vahel *race condition*. Mõlemad konteinerid loevad JSON formaadis loenduri väärtust ja kirjutavad teda üle. Tulemuseks juhtus katsetades umbes 30 sekundi tagant olukord, kus üks konteiner parasjagu üle kirjutab, kuid teine loeb tühja loenduri väärtust. Kasutaja ülesanne on vea olemusest aru saada ja parandada *race condition* omale meeldival viisil. Soovitatud viis ja lahendus kasutavad Pythoni `fnctl.flock()` lukustamist.

Peatüki viimases ülesandes ehk kaheksandas juhitakse tähelepanu asjaolule, et ka Docker pole täiesti ohutu. Ülesandes tuleb Dockeri konteineri seest pääseda hostsüsteemi ja hosti *answers* kaustast „flag.txt“ fail kätte saada.

4.3.2 Nõrgad paroolid

Kategooria esimeses ülesandes käivitab kasutaja Bash skripti, mis genereerib igal uuel käivitusel juhusliku Linuxi kasutaja lokaalse masina peal. Implementeeritud juhuslikku genereerimist näitab järgnev kood.

```
size=${#password[@]}
index=$(($RANDOM % $size))
randvar=${password[$index]}
```

Kasutaja ei tohiks Bash skripti sisu esialgu vaadata ning peaks proovima teha *su weakaccount1*, küsitakse parooli, mida kasutaja ei tea. Kasutaja peab kas automatiseerides või käsitsi paroole

proovides *weakaccount1* sisse saama logitud. Kui sisselogimine õnnestub, leiab kasutaja lipu, mis on *weakaccount1* kodukaustas.

Teises ülesandes käivitatakse Dockeri abil ametlik Redis *image*. Kasutaja peab internetist järgi uurides leidma vaikimisi seadistatud port numbri ning andmebaasist leidma peidetud võtme. Uurimise käigus võiks kasutaja avastada, et vaikimisi Redis parooli ei küsi[24]. Kasutajal tuleb leida moodus, kuidas Redis andmebaasile käske anda ning leida sealt võti. Võtme välja kutsudes tagastab andmebaas lipu.

Kolmandas ülesandes jooksutatakse Flask veebirakendus, mis peidab endas sisselogimis lehte. Kasutajatel tuleb esiteks see leht Raspberry Pi poolt hostitud (kohalikus võrgus) üles leida kasutades *hostname -I* käsku või *nmap* teadmisi. Seejärel soovitatakse kasutajal käsitsi lihtsaid paroole proovida, enam see aga edukat tulemust ei anna. Selles ülesandes pole kasutusel teadatuntud parool nagu: „pi“, „password“, „admin“, vaid hoopis suvaline 3 väiketähega kombinatsioon. Tuleb avastada, et leht kasutab POST päringut. Kasutaja ülesandeks on ise kirjutada Pythonis jõu rünnet (ingl *brute force*) sooritav skript, lubatud on ka ise leida juba toimiv tööriist. Programm peab proovima kõiki a-z kombinatsioone kuni aaa-zzz.

4.3.3 Nõrk krüpteering

Nõrga krüpteeringu esimene ülesanne polegi tegelikult üldse krüpteering, vaid kodeering[25]. Kasutatakse Base64 kodeeringut, mis küll esmapilgul näeb loetamatu välja, kuid tegelikkuses on väga lihtne tema algne sisu välja tuletada. Edukas Base64 dekodeerimine paljastab lipu.

Teises ülesandes on kolm räsi, mis kõik peavad olema dekodeeritud, et täielik lipp saada. Esimene neist on jällegi Base64, teine aga SHA1, ning kolmas MD5. Neist kaks viimast, ehk SHA1 ja MD5 seevastu pole otse lahti tuletatavad. Asjaolu halvendab veel see, et SHA1, kui

[24] https://hub.docker.com/_/redis

[25] https://stackoverflow.com/questions/4070693/what-is-the-purpose-of-base-64-encoding-and-why-it-used-in-http-basic-authentica

ka MD5 on ühesuunalised, seega otsest viisi nende lahti murdmiseks pole[26]. Küll aga on võimalik anda ise sisend näiteks MD5 räsimisfunktsioonile, saadud tulemus ehk räsi koos antud sisendiga salvestada ning hiljem suurte räsi tabelitega võrrelda. Kui on teada näiteks, et sisend "abc" väljastab räsi "900150983cd24fb0d6963f7d28e17f72" ja sellist samasugust räsi kuskil kohatakse, võib olla kindel, et selleks on kasutatud sisendit "abc". Ülesandes võib ise dekrüpteerija kirjutada, kuid lubatud on kasutada ka veebipõhiseid dekrüpteerijaid[27].

Kolmandas ülesandes esitleb ennast MQTT ehk *Message Queuing Telemetry Transport.* MQTT on masinast masinasse suhtlemiseks mõeldud protokoll ning on väga levinud viis IoT seadmete omavaheliseks suhtluseks[17]. Kasutajal tuleb luua MQTT klient, mis „teiselt IoT seadmelt" krüpteeritud andmeid sisse loeb. Krüpteering toimub aga AESiga, mis on vähemalt 2025. aastal üheks tugevamaks krüpteeringuks, tema lahtimurdmine võtab teoreetiliselt miljardeid aastaid[10]. Küll aga on arendaja vea teinud ja oma võtme lekitanud koodi. Kasutaja peab kirjutama Pythoni koodi, mis ühendub MQTT teema külge ja loeb teemast krüpteeritud sõnumi. Krüpteeritud sõnum on võimalik dekrüpteerida lekkinud võtmete abil.

Neljas ülesanne tutvustab külgkanalrünnet. Kasutaja peab kirjutama koodi, mis tuvastab ajalisi erinevusi sisselogimisprotseduuride juures. Sisselogimisaken käib parooli läbi tähe haaval ning iga õige tähe puhul laeb lehekülg kauem, kasutaja kood peab proovima igat tähte kuni õige leidmiseni, seejärel indeksit suurendama.

4.3.4 Füüsiline turvalisus

Füüsilise turvalisuse ülesanded vajavad füüsiliselt ligipääsu seadmele, ehk antud juhul Raspberry Pi-le.

Esimesed kolm ülesannet juhendavat kasutajat kas ise Raspberry Pi Picost ehitama või kasutama olemasolevat BadUSB seadet. Ülesannete lahendaja kasutab spetsiaalset

[26] https://www.reddit.com/r/cryptography/comments/1ai7kfk/decoding_md5_hash/

[27] https://10015.io/tools/md5-encrypt-decrypt

DuckyScripti[28], mis käivitub, kui ohverseadme USB porti BadUSB seade sisestatakse. DuckyScript tema kõige klassikalisemal kasutusjuhul trükib ohverseadmesse käske, ohverseade ise näeb teda kui klaviatuuri[29]. CTF-pi riistvara esimeses ülesandes peab BadUSB sisestades Raspberry Pi-sse, kui ohverseadmesse, käivitama ennekõike terminali ja sinna käskusid sisestades lipufaili kätte saama. Teises ülesandes tuleb ohverseadme töölauast kuvatõmmis teha ja salvestada, kõike seda automatiseeritult vaid USB pulga laadse seadme sisestamisel. Kolmas ülesanne annab kasutajale võimaluse uurida DuckyScripti projekte, mida teised loonud on.

Neljandas ülesandes vihjatakse kasutajale, et Pi küljes olevat märgatud kaableid UARTile vastavatele viikudele. Kasutaja ülesandeks on ohverseadme UART väljundile ühendada Raspberry Pi Pico, mis üle UARTi saadetud signaali sisse loeb. Kasutaja peab leidma õiged viigud ning edastuskiirused iseseisvalt. Samuti tuleb kasutajal luua MicroPythoni kood andmete vastuvõtmiseks Picol.

Viiendas ülesandes on juba valmis loodud GPIO viikudel põhinev koodilukk, kus iga viik simuleerib kui reaalsel koodilukul ühe PIN numbri vajutamist. Koodilukk on aga ühe suure loogikaveaga, nimelt seatakse kõrgeks või aktiivseks ainult oodatud PIN numbri viik, seega õige numbri vajutusel loetakse ta sisse, kuid kõige muu puhul viiku lihtsalt ignoreeritakse, sest pole seadistatud sisendiks. Kasutaja võib kasutada multimeetrit või ka Picot, et tuvastada aktiivses olekus viik ja PIN kood ära arvata. Kasutaja peab ka eksimuse teel GPIO viikude eripärasid tundma õppima, aktiivne olek võib olla nii kõrge kui madal, samuti mõni GPIO viik on spetsiaalse kasutusega ning tema väljund võib viia eksiteele.

Kuuendas ülesandes aga sai eelnev viga parandatud ja nüüd on kõik viigud aktiivsed. Selles ülesandes tuleb kasutada Picot, kasutaja kirjutab koodi, mis proovib läbi kõik viigud lootuses üks hetk „uksekood“ ära arvata.

[28] https://docs.hak5.org/hak5-usb-rubber-ducky/duckyscript-tm-quick-reference

[29] https://en.wikipedia.org/wiki/BadUSB

4.3.5 Haavatavad veebirakendused

Nõrgad veebirakendused on hostitud Raspberry Pi-ga kohalikus võrgus.

Esimeses ülesandes on olemasolev Apachel[30] ja PHP-l põhinev sisselogimis veebileht.

```
<?php
if (isset($_GET['source'])) {
    highlight_file(__FILE__);
    exit;
}

$flag = file_get_contents('/flag.txt');

$message = "Enter the password to retrieve the flag.";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['password']) && $_POST['password'] ===
'admin123') {
        $message = "Flag: " . $flag;
    } else {
        $message = "Wrong password!";
    }
}
?>
```

Ülesande PHP koodis on kaks viga, üks on sisseprogrammeeritud parool „admin123“, ning teine on `highlight_file(__FILE__)`, mille abil on võimalik veebirakenduse lähtekoodile ligi pääseda. Seega kasutajal ei tule muud teha kui aadressiribale õiged parameetrid anda ning lähtekoodi kirjutatud parool saab nähtavaks.

Teisest ülesandest alates on kasutatud Flaski[31]. Teises ülesandes on arendaja sisse unustanud silumisrežiimi (ingl *debug*). Reaalses tarkvaras võib silumisrežiim ründajale tarkvara sees toimuvat reeta. Selles ülesandes tuleb režiim avastada ja sinna on peidetud lipp.

[30] https://httpd.apache.org/

[31] https://flask.palletsprojects.com/en/stable/

Kolmandas ülesandes on eksitud tavanõrkuse CWE-78 vastu[7]. Kasutaja avastab ohtliku funktsiooni `os.popen`, mis on Pythoni `os` teegi funktsioon. `Popen`i kasutatakse GET päringust antud IP aadressi ühenduvuse proovimiseks, kujul: `os.popen(f"ping -c 1 {target}")`. Tegu on süsteemikäsuga, ning käivitatakse *ping* kasutaja sisendi põhjal, mis loodetavasti on IP aadress. Samas ei toimu sisendi kontrolli ning kasutaja saab sisestada ";" sümboli, mis lõpetab eelmise käsu. Kasutaja saab peale ";" sisestada ükskõik millise käsu ja sisuliselt füüsilisel masinal nagu terminaliski toiminguid teha. Siin ülesandes tuleb proovida *;ls* ja *;cat* käske lipu leidmiseks olles *www-data* kasutaja.

Neljandas ülesandes pannakse käima algeline failiserver, millele üle võrgu saab ligi pääseda, sinna faile üles laadida ning samuti ka faile alla laadida. Ülesandes on HTMLi lähtekoodi peidetud vihje, mis viitab administraatori alamkataloogile. Alamkataloog autentimist ei nõua ning sinna on peidetud lipp.

Viiendas tuleb leida üles peidetud alamkataloog, kuna kataloog ise pole ettearvatava nimega tuleks kasutada selleks tööriista nagu *gobuster*[32]. Kasutajal tuleks itereerida läbi erinevate alamkataloogi nimede, kuni leitakse õige.

Kuuendas on arendaja jätnud Javascripti käsu `console.log()`, mis kurdab kahtlase kausta puudumise kohta. See kaust aga sisaldab peidetud lippu. Kasutajal tuleb leida *Developer Tools* tööriist ja veakood.

Seitsmendas ülesandes on failiserveri arendaja loonud esmapilgul mugavust lisava funktsionaalsuse, nimelt kuvatakse failisisu otse veebiserveris, enam ei tule üleslaetud faile alla laadida, et neid lugeda. Kahjuks aga arendaja kasutab jällegi süsteemikäsku nagu *cat*. Suvalise faili üleslaadimine tekitab uue failitee veebiserverisse, ründaja saab teha näiteks */uploads/test.txt;rm sensitive_file.txt*. Test.txt sisu küll kuvatakse, kuid käsule järgnev *rm sensitive_file.txt* eemaldab lokaalsest failisüsteemist tundlikke andmetega faili. Ülesanne ise ootab peidetud lipu leidmist.

[32] https://github.com/OJ/gobuster

Kaheksandas ning kategooria viimases ülesandes võib kasutaja avastada koodis lõigu, mis käivitab .py faililaiendiga skripte automaatselt kui üles laetakse. Tegu on aga failiserveriga, ning igaüks saab .py faile, mille sisu kuidagi ei kontrollita, üles laadida. Tulemuseks käivitub tundmatu Pythoni skript serveris. Kasutajal tuleb kirjutada Pythoni skript, mis üles laadides leiab lipu failiserverist.

4.3.6 Võrgundus

Võrgu ülesanded toimuvad samas võrgus, kuhu Raspberry Pi ühendatud on. Üldiselt eeldatakse kasutaja lokaalset koduvõrku, kuhu ka teised seadmed nagu sülearvutid ja nutitelefonid ligi pääsevad.

Esimeses ülesandes tuleb Wiresharki[33] või muu sarnase tööriista abil monitoorida pakette. Kasutaja võiks avastada ükskõik millise seadme peal (ühendatud samasse võrku mis Pi), et UDP port 1337-le saadetakse pidevalt pakette. Filtreerides võiks leida, et paketti on peidetud lipp.

Teises ülesandes pannakse Dockeri abiga käima mitu erineva teenust. Käivitub nii võlts veebirakendus, võlts SSH, võlts FTP ning veel üks võõras teenus. Kõik teenused kasutavad porte avatud ja aktiivses olekus. Kasutajal tuleks *nmap*[34] tööriistaga skaneerida kõiki võrgu porte, et leida üles lipp. Tuleb leida *nmap*i õiged argumendid ning katse-eksituse meetodil võiks lõpuks silma paista *Unknown* väärtusega port, juhul, kui *nmap* käivitati oodatud lippudega. Kasutajal tuleks, leida mida tundmatu teenus teeb ning temast lipp kätte saada.

Kolmandas ülesandes tuleb kasutajal käituda kui ründaja ise. On loodud veebileht, mis kasutab HTTP-d, et korraldada sisselogimist. HTTP aga ei tee mingit krüpteeringut ning kasutaja ehk ründaja peab sooritama MITM (ingl *man in the middle*) rünnaku. Ülesandes on loodud Pythoni skript, mis simuleerib sisselogivaid kasutajaid, ülesande lahendajal tuleb madala privileegiga

[33] https://www.wireshark.org/

[34] https://nmap.org/

sisselogijate seast avastada administraatori kasutajanimi ja parool. Boonusena on antud ülesanne, kus tuleb luua sisselogimise andmeid talletav server.

4.3.7 Aegunud tarkvara

Aegunud tarkvara ülesanded põhinevad tänapäeval päriselt kasutusel olevate tehnoloogiate nõrkustel.

Esimene ülesanne käivitab Grafana[35] aegunud versiooni. Antud aegunud versioon on aga haavatav CVE-2019-13068'le[36]. See keskmise ohuga turvanõrkus lubab kasutada HTMLi injektsiooni, ehk kohas, kus HTML koodi lugeda ei tohiks, seda siiski loetakse.

Teises Grafana ülesandes on aga ohutase juba kõrge või lausa kriitiline. Aegunud Grafana versioon võimaldab ründajal kataloogihüppeid[37] sooritada, ning pääseda kasvõi */etc/shadow* failile ligi. Reaalses elus annaks selline võimalus ründajal potentsiaalse administraatori kasutaja enda valdusesse saada. Samuti on Grafana andmebaasid ning SSH võtmed ligipääsetavad. Kusagil tundlike andmete sees on lipp.

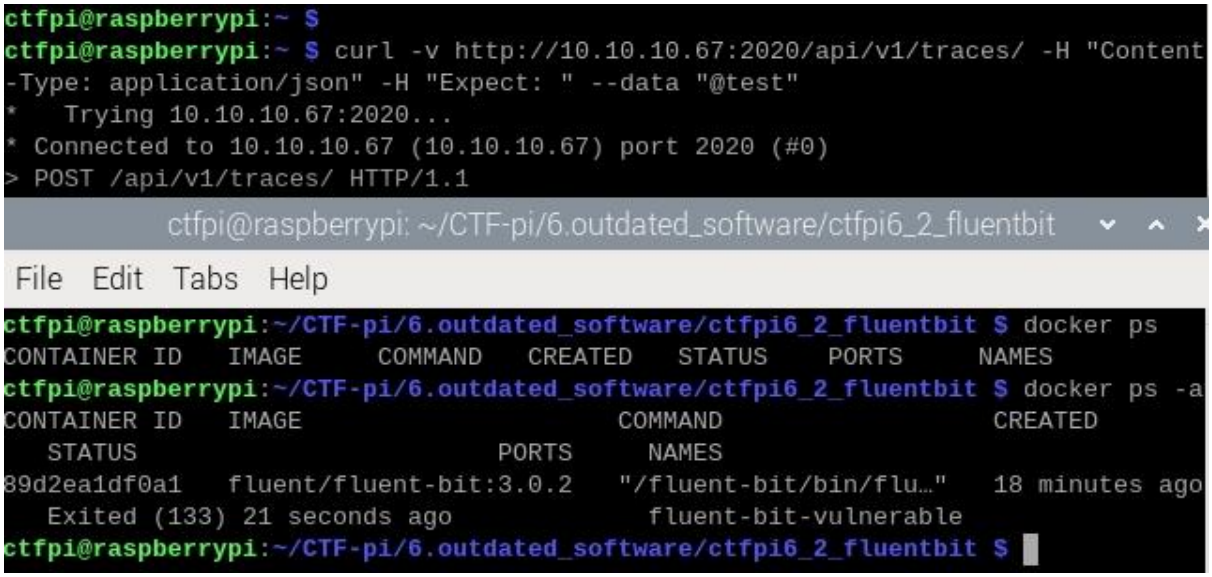
Kolmas ülesanne on ehitatud Fluentbitile[38]. Aegunud Fluentbiti versioon lubab kasutajal ületada puhvri suuruse ning seetõttu rakendus krahhida. Kasutaja sooritab sisuliselt ummistusründe, kuna rakenduse krahhimine kas vähendab käideldavust või üldse peatab teenuste töö. Näidislahendus on toodud Joonis 3.

[35] https://hub.docker.com/r/grafana/grafana

[36] https://www.exploit-db.com/exploits/51073

[37] https://akit.cyber.ee/term/39

[38] https://fluentbit.io/



Joonis 3. Ülemises terminalis käivitatud eksploit, päring kasutab sisendiks faili „test“, mis sisaldab JSON formaadis puhvri jaoks liiga suurt sisendit. Alumises aknas on näha krahhinud Fluentbit peale eksploidi käivitamist.

4.3.8 Ebaturvaline API

API ülesannete esimeses osas tuleb kasutajal luua enda veebihaak[39] (ingl *webhook*). Veebihaagi olemus lubab igaühel ligipääsuga teda kutsuda. Antud ülesanne on ehitatud HomeAssistant[40] tarkvarale, kus veebihaak tekitab HomeAssistanti logidesse teate. Kasutajale tuuakse näiteid teistest potentsiaalsetest veebihaagi ohtudest, HomeAssistant võimalustele lähtudes.

Teises ülesandes on jällegi kasutusel HomeAssistant. Selles ülesandes enam ei luua veebihaak, vaid reaalne REST API[41]. Kasutajad käivitavad Pythoni skripti, mis prindib välja kõik API vastused, olenevalt seadistusest võivad seal olevad väärtused sisaldada geograafilisi koordinaate, sensorite olekut jms.

[39] https://akit.cyber.ee/term/14312-webhook

[40] https://www.home-assistant.io/

[41] https://www.ibm.com/think/topics/rest-apis

Kõige viimane ülesanne on eriline ja rohkem lisatud kui kogu CTF-pi täiendusena. Raspberry Pi või tavalise arvuti pealt jooksutatav *OWASP Juice Shop*[42] käivitab veebilehe, mis meenutab kui e-poodi. Tegelikkuses on e-poodi sisse integreeritud suurel määral turvanõrkusi, mida kasutajad ilma abi ja juhenditeta leidma peavad.

[42] https://hub.docker.com/r/bkimminich/juice-shop

5. Testimine

Lahenduse kvaliteedi tagamiseks ning korrektse käitumise veendumiseks on väga oluline tehtut testida. Töö valmis täielikult iseseisvalt ning juhendid ja kood on kirjutatud kindla nägemuse ning käekirjaga, mis teisele isikule ei pruugi arusaadav olla. Tagamaks, et töö oleks üheti mõistetav ja töötaks kvaliteetselt, sai tehtud detailselt iseseisvat testimist kui ka arendusega mitte kokkupuutunud katsetajatega testimist.

5.1 Iseseisev

Iga ülesanne CTF-pi repositooriumist sai juhendi abil läbi proovitud Raspberry Pi 5 peal. Teststsenaarium nägi välja järgnev:

1) Lugedes CTF-pi põhilist *README.md* faili, mõelda, kas kasutajale on piisavalt selgeks tehtud, mida edasi ette võtta.

2) Iga ülesande juures lugeda läbi juhend ja selgitada, kas sammud, mida kasutaja tegema peab, on arusaadavad.

3) Veenduda, et juhend toob välja kõik eeldused ning nende olemasolul ülesanne käivitub riketeta.

4) Ülesanne, kus on toimiv konteiner või aktiivselt käiv rakendus, veenduda, et ei toimu krahhe ning rakendus püsib käivas olekus.

5) Juhendis toodud lahenduskäigule või vihjetele toetudes proovida jõuda lipuni.

6) Peale lahendust kaaluda üle kasutajate seadmete turvalisus ning veenduda, et nõrgad seadistused kaoksid peale ülesande lahendamist. Juhendis on eraldi sektsioon, mida tuleb teha.

5.1.1 Tulemused

Iseseisva testimise tulemusel võib üldjoontes öelda, et arendatud lahendused toimisid ootuspäraselt ning suuremaid funktsionaalseid või tehnilisi tõrkeid ei ilmnenud. Testimisel tuvastatud vead olid pigem väiksema mõjuga ega mõjutanud süsteemi üldist toimimist olulisel määral. Alljärgnevalt on välja toodud peamised avastatud puudused koos nende lühikirjeldusega:

- Eeltingimuste ja eesmärkide sõnastus oli segane ning kohati oli teave laialivalguv.
    - Lahenduseks sai tehtud juhendi jaoks üks kindel mall, mille järgi kõigi ülesannete *README* failid said ühtlaseks viidud.
- Juhendis puudus selgelt ülesande raskust näitav indikaator.
    - Sai lisatud raskusastme näitaja 1-5, kus 1 on väga lihtne, 5 väga raske.
- Dockeri registrist pärinevatel kujutisfailidel oli Raspberry Pi 5 mäluprobleeme, mistõttu rakendus krahhis kohe käivitamisel.
    - Sai lisatud peamisse *README*, kui ka vea avalduva ülesande juurde parandusjuhis.
- Oli ülesandeid, mis sisaldas *hard coded* Dockeri kujutisi, kuigi polnud vajalik.
    - *Compose.yaml* ja Dockerfile said muudetud.
- Nmap ülesanne tekitas tundlike andmetega kausta, mistõttu kasutajad võisid kogemata gitiga selle lekitada.
    - Sai lisatud .gitignore vastava kausta jaoks.
- Mõni kujutis kasutas sama tarkvara, aga erinevaid versioone, kuigi selleks polnud põhjust ja asjata raisati mälu, näiteks Python3.13-slim ja Python3.12-slim.
    - Kujutised said viidud samale versioonile ülesannete vahel.

5.2 Tartu Häkkerikoda

Saamaks tagasisidet projektiga varem mitte kokkupuutunud isikutelt ning võimalikult erineva taustaga, sai Tartu Häkkerikojaga kokku lepitud CTF teemaline üritus[30]. Üritusel tehtud fotod on vaadatavad Lisa I. Häkkerikoda on Tartus Delta õppehoone lähedal asuv sõbralik ja IT-st huvituv kommuun. Üritusel võttis osa 10 inimest, kelle seas oli nii noori kui vanemaid ning kelle taust erines märgatavalt, oli vägagi kogenud lahendajaid, kui ka Raspberry Pi ja CTF-ga vähem kokkupuutunud inimesi.

Sündmus nägi välja selline:

1) osalejad lülitasid Raspberry Pi-d sisse ja tuli teha esmane seadistus.

2) Seejärel tuli installida Docker, juhised olid kuvatud projektori abil tahvlile.

3) Kloonida CTF-pi repositoorium Pi-le.

4) Alustada lahendamist esimesest ülesannete plokist.

Kümnest osalejast viis olid füüsiliselt klaviatuuri, hiire ning monitoriga Raspberry Pi külge ühendatud.

Teised viis osalejat kasutasid SSH ühendust. SSH kliendi said osalejad valida omale meelepärase ning mingeid piiranguid ei määratud. Kasutati laialdaselt Linuxi terminalist OpenSSH-d, kuid eriti mugavalt lahendati ka VSCode-i SSH laiendusega.

Kuna ürituse aeg oli piiratud ja CTF-pi ülesandeid on palju, siis ei oleks olnud võimalik ühe korraga kõiki ülesandeid ära lahendada. Selleks jäid välja keerulisema seadistusega ülesanded nagu riistvara kategooria omad. Üritus algas mitteametlikult kell 18, kuid üles seadmise ajakulu tõttu algus natuke hilines. Üritus kestis kokku umbes 3 tundi, kuni viimane osaleja lahedamise lõpetas.

Testimisel sai hinnatud osalejate edasijõudmist ja juhendite arusaadavust, samuti andsid osalejad ise aktiivselt tagasisidet ning ka lõpparvamuse projektist.

5.2.1 Tulemused

- Riistvara probleemide taha ei jäänud ükski osaleja, kõik said Raspberry Pi ilusti käima ja internetti.
- Esimene katsumus oli Dockeri installimisega, kuna käske oli vaja sisestada 3 ning ekraanil kuvades ei jõudnud kõik järge pidada. Kulus umbes 30 minutit kuni kõigil toimiv Docker oli.
    - Lahenduseks sai CTF-pi repositooriumi põhilisse *README* faili pandud juhised Dockeri installeerimiseks.
- Repositooriumini jõudmine oli raskendatud, osalejad pidid ekraanilt maha kirjutama.
    - Tegu on vähetuntud ja uue repositooriumiga, seega otsingumootoritest temani hetkel ei jõua, siin on ainus lahendus CTF-pi populariseerimine.
    - Edaspidiste ürituste korraldamisel võiks luua lühilingi või eelnevalt repositoorium avada seadmetel.
- Tundus, et osalejad jäid segadusse, kuidas asuda kloonitud repositooriumil päriselt ülesannete kallale.
    - Paranduseks sai repositooriumi *README* failis lahendamisjuhised ülespoole tõstetud, eelnevalt olid need all.
- Ülesande enda manuaalides olid vihjed liiga kergesti nähtavad.
    - Sai kasutatud HTMLi `<details>` elementi, et peale klikkides alles vihjed avaneksid, muidu on peidetud.
- Küsiti, kas Raspberry Pi olemasoluta on võimalik CTF-pi ülesandeid teha.
    - Esimeses versioonis polnud võimalik, CTF-pi sai aga uuenduse ning kõik kategooriad peale füüsilise turbe on käivitatavad Linuxi virtuaalmasina või arvuti peal.
        - Testimist sooritati Lubuntu[43] virtuaalmasina peal.

[43] https://lubuntu.me/

- Originaalselt olid ülesanded nimetatud kujul: „ctfpix_y“, kus „x“ on kategooria number ja „y“ ülesande unikaalne number. See aga ei lubanud peale vaadates ülesande sisust aru saada ning kogu ülesanne tuli failiteelt avada.
    - Lahenduseks sai ülesande nimi „ctfpix_y“ muudetud „ctfpix_y_lühinimi“ kujule.
- Kogenud kasutajatel, kes ei soovinud lihtsamaid harjutusi teha, oli raskem navigeerida repositooriumi kaustade vahel, tuli kogu failitee läbi käsitsi käia.
    - Põhilisse *README* faili sai lisatud Markdown tabel, mis sisaldab otselinki iga ülesande sisuni. Täpsemalt on seda näha Lisa II.

Ka Häkkerikoja testimisega suuri vigu ei avastatud, osalejad olid rahul tehtud tööga ning andsid positiivse hinnangu. Häkkerikoja eestvedaja tänas omaltpoolt ja luges ürituse edukaks.

Mitmed osalejad tõdesid, et on varem CTF-pi ülesannete konfiguratsioonivigadega kokku puutunud ning, et lahendatud ülesanded panid mõtlema enda küberturvalisuse peale. Täpselt see oligi CTF-pi algeesmärk, et tähelepanu saaks juhitud turvalisuse poolele.

Mitmed osalejad ütlesid, et plaanivad kodus ülejäänud tegemata ülesanded ka läbi teha, kaasaarvatud riistvara ülesanded. Lubati ka probleemide korral GitHubis *Issue* tõstatada.

Väga kogenud lahendaja pakkus kõigi ülesannete lahendamise ajakuluks 8-10 tundi koos vihjete ja lahenduste abiga.

Umbkaudselt võiks Raspberry Pi-ga võõral ja küberturbe algajal kuluda 40 tundi. See tulemus on saadud arvestades Häkkerikoja keskmist lahendamistempot, kuid samas võttes arvesse ka kui palju aega kulus ülesande enda tegemiseks (autori poolt). Ülesande valmistamine hõlmas ideede kogumist, uurimustööd, arendust, testimist ning ka juhendi kirjutamist. Kuna ülesannete sooritajad ideid koguma, juhendeid kirjutama, otseselt arendama ei pea, siis on tundide mõistes arvutusest need osad välja jäetud.

6. Tulevikuplaanid

CTF-pi on kogu maailmale avatud avastamiseks ning esimene suurim lootus ja eesmärk oleks projekti rohkem populariseerida. Sellisel Dockeri kujul ja Pi-le keskenduvat CTF ülesannete kogumit ei ole kerge tänapäeval veel leida.

Tõenäoliselt võiks veel üritusi Häkkerikojas või ka mujal korraldada, et saavutada suuremat populaarsust. Samuti pole välistatud CTF-pi tutvustamine *online* küberturbe kogukondadele.

Kuigi 2025. aastal ülikoolis ühtegi kursust pole, kus CTF-pi teemasid õpetada, sai andmeturbe õppejõuga arutatud CTF-pi kasutamist kui lisaülesannete ja tagavarana kursuse praktikumides. Selle võimaluse lihtsustamiseks on kõik võimalikud ülesanded ainult Raspberry Pi peal toimivalt kujult viidud Linuxil toimivale kujule, see võimaldab tudengitel ka näiteks Ubuntu virtuaalmasinas CTF-pi ülesandeid lahendada.

Hetkel keskendub CTF-pi iseõppimisele ning vabalt vastuste andmisele. Kui tahta kasutada teda turniiridel või õppetöö hindamisel tuleks lipufailid kasutaja valdusest ära võtta. Üks võimalik lahendus on Dockeri registrisse[44] laadida valmis kujutis. Küll aga on võimalik Dockeri kujutiste sisu ikkagi natuke lisatööd tehes näha[45].

[44] https://hub.docker.com/

[45] https://stackoverflow.com/questions/44769315/how-to-see-docker-image-contents

7. Kokkuvõte

Küberohtude arv ei näita märke vaibumisest ning üks üldjuhul möödavaadatud osa ründesihtmärkidest on IoT seadmed. IoT on saanud tavakasutajale palju kättesaadavamaks ning internetist on võimalik läbi proovida väga suurel hulgal erinevaid projekte. Laiem kasutajaskond aga toob endaga kaasa suurema ründepinna potentsiaalsetele häkkeritele. IoT küberturbe kinnistamiseks sai loodud CTF-pi.

Antud bakalaureusetöö (CTF-pi) käsitleb tüüpilisi asjade interneti seadmete turvanõrkusi ning esitab praktilise lahenduse nende õpetamiseks. Välja arendatud CTF-pi on Raspberry Pi platvormil toimiv, avatud lähtekoodiga harjutuste kogum, mis võimaldab kasutajatel simuleerida rünnakustsenaariume ning omandada teadmisi küberturbe praktilistest aspektidest. Ülesanded põhinevad CTF (ingl *Capture the Flag*) formaadil ja hõlmavad nii tark- kui ka riistvarapõhiseid haavatavusi.

CTF-pi koondab 38 ülesannet kaheksas kategoorias: üldised teadmised, nõrgad paroolid, nõrk krüpteering, füüsiline turvalisus, haavatavad veebirakendused, võrgundus, aegunud tarkvara ja ebaturvalised API-liidesed. Platvormi loomisel kasutati kaasaegseid tööriistu, sh Docker, GitHub ja Markdown, mis võimaldavad harjutuste kiiret levitamist ja taaskasutust.

Süsteemi testiti nii iseseisvalt kui ka ürituse vormis Tartu Häkkerikojas, kus osalesid erineva tasemega küberturbehuvilised. Testide põhjal hinnati platvormi kasutusmugavust, juhendite mõistetavust ja tehnilist töökindlust. Tagasiside põhjal tehti mitmeid täiustusi, sealhulgas ühtlustati dokumentatsiooni struktuur, täpsustati ülesannete raskusastmeid ja lisati funktsionaalsus ülesannete läbimiseks ka virtuaalses Linuxi keskkonnas.

Töö peamine panus seisneb ligipääsetava ja kohandatava õppelahenduse loomises, mis võimaldab nii algajal kui ka kogenud kasutajal oma ülesandeid luua. CTF-pi aitab mõtestada IoT seadmete turvariske ja kujundada tõhusamaid arusaamu küberturbe parimatest tavadest juba arendusprotsessi varases faasis.

Kasutatud kirjandus

[1] Tebbenkamp, M, *IoT is Everywhere.* https://www.cosn.org/iot-is-everywhere/ (vaadatud 13. mai 2025).

[2] ITRC, Identity Theft Resource Center, *Annual Data Breach Report.* https://www.idtheftcenter.org/publication/2023-data-breach-report/ (vaadatud 6. detsember 2024).

[3] Fortinet, *Types of Cyber Attacks.* https://www.fortinet.com/resources/cyberglossary/types-of-cyber-attacks (vaadatud 1. november 2024).

[4] Physical Security and Cybersecurity: How They Work Together, *LenelS2.* https://www.lenels2.com/en/news/insights/Physical_and_Cybersecurity.html (vaadatud 1. november 2024).

[5] Baker, K. (2024), CrowdStrike, *12 Most Common Types of Cyberattacks.* https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/common-cyberattacks/#11.-IoT-Based-Attacks (vaadatud 1. november 2024).

[6] YoungWonks, *Why is Raspberry Pi Used for Iot Devices?* https://www.youngwonks.com/blog/why-is-raspberry-pi-used-for-iot-devices (vaadatud 3. november 2024).

[7] Common Weakness Enumeration, *CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')* https://cwe.mitre.org/data/definitions/78.html (vaadatud 12.mai 2025).

[8] Raspberry Pi 5 tooteleht (2024). https://www.raspberrypi.com/products/raspberry-pi-5/ (vaadatud 3. november 2024).

[9] Fortinet, *Top IoT Device Vulnerabilities.* https://www.fortinet.com/resources/cyberglossary/iot-device-vulnerabilities (vaadatud 5. mai 2025).

[10] Kime, C, Lafferty, M (2024). *Strong Encryption Explained: 6 Encryption Best Practices* https://www.esecurityplanet.com/networks/strong-encryption/ (vaadatud 12.mai 2025).

[11] Talking IoT. *The Top 10 IoT Development Boards for 2024.* https://talkingiot.io/the-top-10-iot-development-boards-for-2024/ (vaadatud 4. mai 2025).

[12] Zonetronik artikkel. *Top 10 affordable programmable development boards*. https://www.zonetronik.com/en/top-10-affordable-programmable-development-boards/ (vaadatud 4. mai 2025).

[13] Nabto artikkel. *The Best Audio Development Boards for IoT.* https://www.nabto.com/the-best-audio-development-boards-for-iot/ (vaadatud 4. mai 2025).

[14] Geerling, J. (2024), *When did Raspberry Pi get so expensive?* https://www.jeffgeerling.com/blog/2024/when-did-raspberry-pi-get-so-expensive (vaadatud 4. mai 2025).

[15] CTFtech sündmuse leht. *Telia Cyber Battle of Nordic-Baltic 2024 – bootcamps & qualification rounds*. https://ctftech.com/telia-cyber-battle-of-nordic-baltic-2024-cybercation/ (vaadatud 4. mai 2025).

[16] Vulnhub, *about.* https://www.vulnhub.com/about/ (vaadatud 4. mai 2025).

[17] HiveMQ, *MQTT Essentials*. https://www.hivemq.com/mqtt/ (vaadatud 12.mai 2025).

[18] Raspberry Pi Pico, andmeleht. https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf (vaadatud 4. mai 2025).

[19] DockerDocs, *What is Docker?* https://docs.docker.com/get-started/docker-overview/ (vaadatud 4. mai 2025).

[20] Walker, J. (2024), *How to Expose a Docker Port – Tutorial & Examples* https://spacelift.io/blog/docker-expose-port (vaadatud 4. mai 2025).

[21] DockerDocs, *Build and push the image.* https://docs.docker.com/get-started/introduction/build-and-push-first-image/#build-and-push-the-image (vaadatud 4. mai 2025).

[22] DockerDocs, *Build and push your first image*. https://docs.docker.com/get-started/docker-concepts/running-containers/publishing-ports/ (vaadatud 4. mai 2025).

[23] DockerDocs, *Docker Compose Quickstart.* https://docs.docker.com/compose/gettingstarted/ (vaadatud 4. mai 2025).

[24] Markdown Guide, *Getting Started.* https://www.markdownguide.org/getting-started (vaadatud 4. mai 2025).

[25] Wilson, B (2024), *IoT Security: Avoid These 5 Mistakes.* https://www.devprojournal.com/technology-trends/internet-of-things/iot-security-avoid-these-5-mistakes/ (vaadatud 5. mai 2025).

[26] Kovacevic, A (2025), *Biggest IoT Data Security Mistakes That Make Businesses Vulnerable.* https://www.iotforall.com/biggest-iot-data-mistakes (vaadatud 5. mai 2025).

[27] Owl Cyber Defense, *The "S" in IIoT Stands for "Security".* PDF dokument(nõuab registreerimist). https://owlcyberdefense.com/resource/the-s-in-iiot-stands-for-security/ (vaadatud 5. mai 2025).

[28] GitHub, *Basic writing and formatting syntax.* https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax (vaadatud 11.mai 2025).

[29] Kushner, D (2013), *The Real Story of Stuxnet*. https://spectrum.ieee.org/the-real-story-of-stuxnet (vaadatud 5. mai 2025).

[30] Häkkerikoda, Raspberry Pi & CTF üritus. https://hakkerikoda.ee/et/event/ctf-pi (vaadatud 9.mai 2025).

[31] asimily, *Identifying Lateral Movement on Your Device Network.* https://asimily.com/blog/identifying-lateral-movement-on-your-device-network (vaadatud 5. mai 2025).

[32] Antonakakis, M. jt (2017), *Understanding the Mirai Botnet,* Usenix, 1, ISBN: 978-1-931971-40-9

[33] Antonakakis, M. jt (2017), *Understanding the Mirai Botnet,* Usenix, 2, ISBN: 978-1-931971-40-9

[34] Sabin, S(2025), *Chinese robotics manufacturer left backdoor in product.* https://www.axios.com/2025/04/01/threat-spotlight-backdoor-in-chinese-robots-future-of-cybersecurity (vaadatud 9.mai 2025).

[35] Fortinet, *Worm Virus Definition.* https://www.fortinet.com/resources/cyberglossary/worm-virus (vaadatud 9. mai 2025).

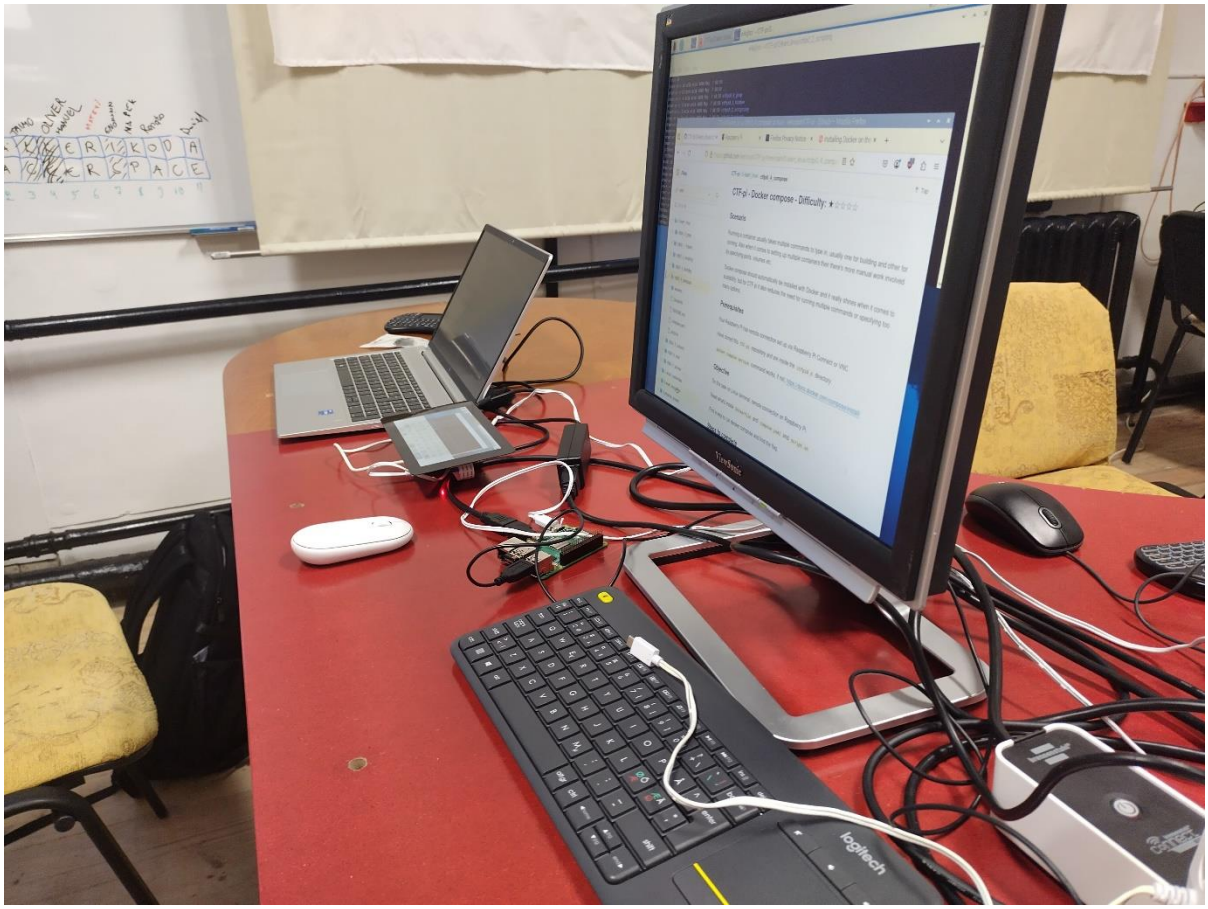
[36] Unitree, Go1 tooteleht. https://shop.unitree.com/products/unitreeyushutechnologydog-artificial-intelligence-companion-bionic-companion-intelligent-robot-go1-quadruped-robot-dog (vaadatud 9.mai 2025).

[37] The MalwareMustDie Blog, *MMD-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled.* https://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html (vaadatud 9. mai 2025).

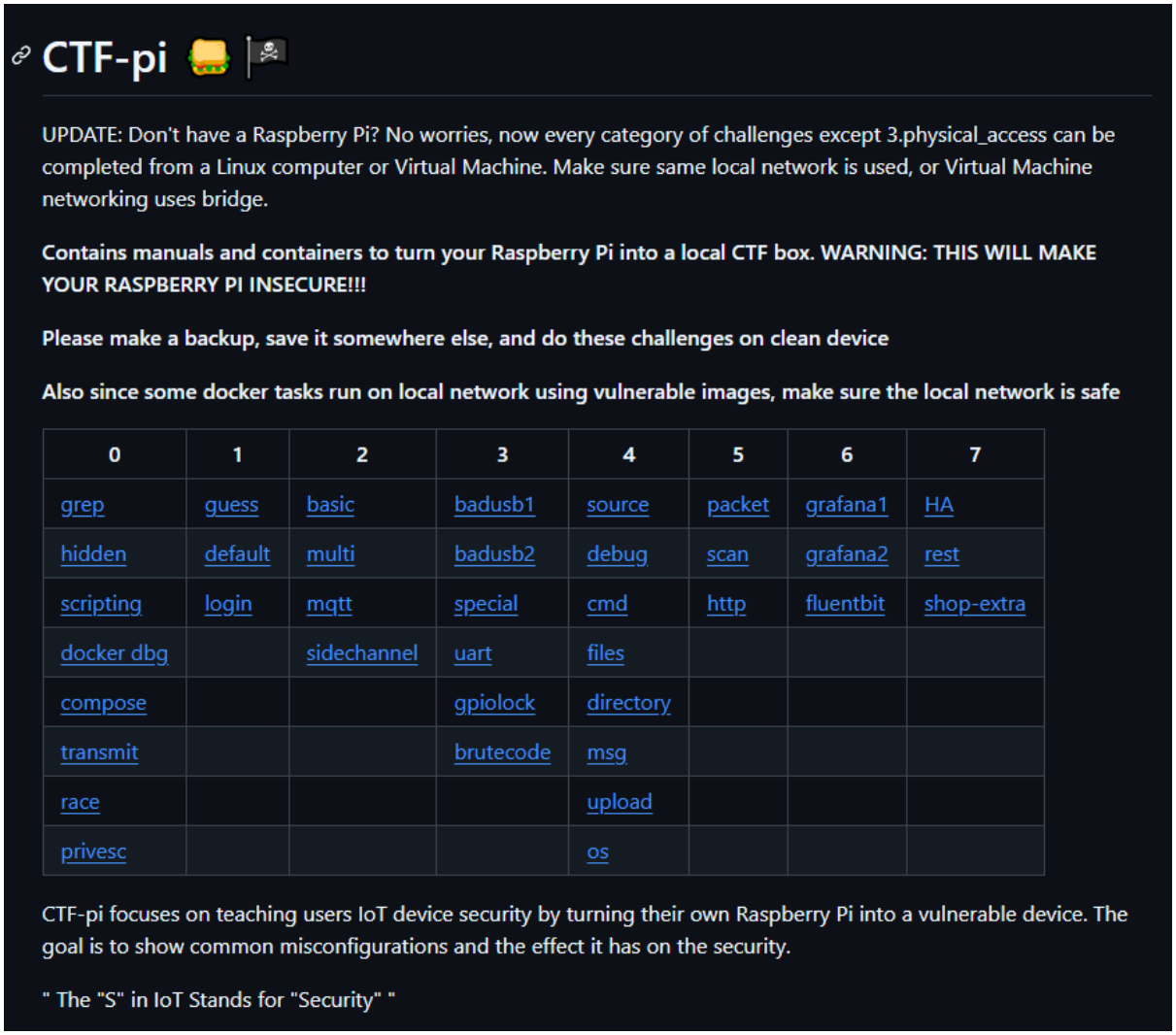
[38] B. Krebs. *Krebsonsecurity hit with record DDoS.* https://krebsonsecurity.com/2016/09/krebsonsecurity-hitwith-record-ddos/ (vaadatud 9.mai 2025).

Lisad

Lisa I Häkkerikoja ürituse fotod


logitech

Lisa II CTF-pi repositooriumi kuvatõmmis



CTF-pi

UPDATE: Don't have a Raspberry Pi? No worries, now every category of challenges except 3.physical_access can be completed from a Linux computer or Virtual Machine. Make sure same local network is used, or Virtual Machine networking uses bridge.

Contains manuals and containers to turn your Raspberry Pi into a local CTF box. WARNING: THIS WILL MAKE YOUR RASPBERRY PI INSECURE!!!

Please make a backup, save it somewhere else, and do these challenges on clean device

Also since some docker tasks run on local network using vulnerable images, make sure the local network is safe

0	1	2	3	4	5	6	7
grep	guess	basic	badusb1	source	packet	grafana1	HA
hidden	default	multi	badusb2	debug	scan	grafana2	rest
scripting	login	mqtt	special	cmd	http	fluentbit	shop-extra
docker dbg		sidechannel	uart	files			
compose			gpiolock	directory			
transmit			brutecode	msg			
race				upload			
privesc				os			

CTF-pi focuses on teaching users IoT device security by turning their own Raspberry Pi into a vulnerable device. The goal is to show common misconfigurations and the effect it has on the security.

" The "S" in IoT Stands for "Security" "

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Kenno Kallastu

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose CTF-pi – CTF ülesanded IoT turvalisuse õpetamiseks,

   mille juhendaja(d) on Alo Peets,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, 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.

Kenno Kallastu

20.05.2025