TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Laura Ernits 164128IAPB

EBATURVALISE JAVA VEEBIRAKENDUSE PARANDAMISE PRAKTIKUM VIRTUAALLABORIS

Bakalaureusetöö

Juhendaja: Sten Mäses
MSc
Kaasjuhendaja: Ago Luberg
MSc

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Laura Ernits

21.05.2019

Annotatsioon

Käesoleva lõputöö eesmärgiks oli luua praktikum Tallinna Tehnikaülikooli tudengitele Küberturbe aluste (ITI0103) õppeaines, mis võimaldaks tudengitel saada praktilisi kogemusi veebirakenduste turvalisusest. Teoreetilise põhjana kasutati OWASP-i projekti Top 10, valides sealt välja 5 turvaauku, mille parandamist kontrolliti.

Praktikumi lahendati kauglaborite süsteemis, kus asus turvaaukudega veebirakendus. Lahendaja ülesandeks oli parandada turvaaugud, muutes rakenduse lähtekoodi. Tulemusi kontrolliti automaatselt, kasutades GitLabi pidevat integratsiooni.

Praktikum viidi läbi planeeritud mahus. Tulemustest selgus, et praktikumi lahendanud tudengid hindasid seda võrdlemisi huvitavaks ja kasulikuks, kuid keeruliseks. Edaspidise tegevusena soovitatakse praktikumi korraldamist teise või kolmanda kursuse tudengitele, kellele ülesanne on võimetekohasem.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 10 peatükki, 9 joonist, 4 tabelit.

Abstract

Patching Insecure Java Web Application in a Hands-On Virtual Lab

The aim of this thesis was to develop a practical lab for students of the Tallinn University of Technology in the Foundations of Cyber Security (ITI0103) course which would enable students to gain practical experience of the web application security. The OWASP Top 10 project was used as the theoretical basis, out of which 5 vulnerabilities were chosen to be represented in the lab.

The lab was solved in a distance laboratory system using a vulnerable web application. The task of the student was to fix the vulnerabilities by changing the source code of the application. The results were tested using GitLab's continuous integration pipeline.

The lab was conducted as planned. The results showed that the students who solved the lab, evaluated it as relatively interesting and useful, but at the same time complicated. It is proposed to conduct the lab for second or third year students, who will find it more suitable.

The thesis is in Estonian and contains 32 pages of text, 10 chapters, 9 figures, 4 tables.

Lühendite ja mõistete sõnastik

Ad hoc	*Spetsiaalselt just selleks otstarbeks või juhuks* [1]
API	*Application Programming Interface*, rakendusliides
GDPR	*General Data Protection Regulation*, isikuandmete kaitse üldmäärus[1]
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel
HTTP	*HyperText Transfer Protocol, hüperteksti edastuse protokoll* [2]
NPM	*Node Package Manager*, Paketihaldur Node.js teekide jaoks[2]
ORM	*Object-relation mapping, objekt-relatsioonvastendus* [2]
OWASP	*Open Web Application Security Project*, avatud veebirakenduse turvalisuse projekt
REST	*Representational State Transfer, esitusoleku siire* [2]
RIA	Riigi Infosüsteemide Amet[3]
URI	*Uniform resource identifier, ühtne ressursi-identifikaator* [2]
XML	*Extensible Markup Language*, laiendatav märgistuskeel
XSS	*Cross-site scripting, murdskriptimine* [2]
XXE	*XML external entity attack, XML välisolemrünne* [2]

[1] https://eugdpr.org

[2] https://github.com/npm/cli

[3] https://www.ria.ee

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Hetkeseisuga on Tallinna Tehnikaülikooli informaatika õppekavas (IAIB17/17) küberturvalisuse osakaal võrdlemisi väike. Bakalaureuse mahust (180 EAP-d) vaid 2% hõlmab Küberturbe aluste (ITI0103) kursus, mille eesmärk on anda põgus ülevaade küberturbe valdkonnast ning tutvustada sellega seonduvaid põhimõtteid ja mõisteid [3]. Kursuse maht on 3 EAP-d ja seda viib 2019. aasta kevadsemestril läbi MSc Sten Mäses.

Kuivõrd paljud tudengid alustavad tööd veebiarendusmeeskondades, on äärmiselt oluline tutvustada tudengitele ka veebirakendustega seonduvat turvalisust. Antud lõputöö eesmärk on luua praktikum veebirakenduste turvalisuse teemal, mis hõlmaks ühe hindelise osa Küberturbe aluste ainemahust ning annaks praktilise kogemuse valdkonnas, mida senini Tallinna Tehnikaülikooli informaatika õppekavas vaid põgusalt käsitleti.

Küberturvalisuse valdkonna olulisus peitub paljuski nii eetilistes kui rahalistes küsimustes. Kui eksitakse isikuandmete kaitse üldmääruse (GDPR – ingl *General Data Protection Regulation*) vastu, võib sellega kaasneda lisaks mainelisele kahjule ka rahaline trahv. Sellest tulenevalt on äärmiselt oluline käsitleda turvalisust kui disaini alustala, mitte üritada valminud koodi hiljem turvaliseks muuta.

Paljud olemasolevad veebiturbe õppimise platvormid pakuvad vigade tuvastamise lähenemist. Selle praktikumi eesmärk on rõhuda eelkõige turvavigade parandamisele muutes rakenduse lähtekoodi, mistõttu on vead koos nende esiletoomiseks vajalike juhistega tööjuhendis välja toodud. Oluline on tutvustada kriitilisi turvavigu, et suunata tudengeid edaspidises töös sellelaadsete vigade teket ennetama.

Töö alguses antakse ülevaade teoreetilistest alustest, seejärel kirjeldatakse valminud praktikumi tehnilist põhja ja tööprotsessi. Töö lõpus esitatakse praktikumi tulemused koos järeldustega.

2 Kirjanduse ülevaade

Riigi Infosüsteemi Ameti (RIA) küberturvalisuse aastakokkuvõttest selgub, et iga aastaga registreeritakse järjest rohkem tarkvara turvaauke. 2018. aastal pöörduti RIA poole seoses andmeid või infosüsteeme mõjutanud intsidentidega 17 440 korda [4]. See on ligikaudu 63% enam kui seda tehti 2017. aastal, mis näitab, et küberturvalisus on jätkuvalt oluline teemavaldkond, millega tuleb igapäevaselt tegeleda.

2.1 OWASP

Üks tuntumaid veebirakenduste turvavigu analüüsiv projekt on avatud veebirakenduse turvalisuse projekt (OWASP – ingl *Open Web Application Security Project*). See on mittetulunduslik organisatsioon, mis asutati 1. detsembril 2001. aastal Ameerika Ühendriikides, eesmärgiga võimaldada organisatsioonidel luua, arendada, omandada, hallata ja hooldada rakendusi, mida saab usaldada [5]. OWASP-l on eesmärkide saavutamiseks mitmeid erinevaid projekte. Selles töös käsitletakse täpsemalt OWASP Top 10, rakenduste turvalisuse kontrollimise standardit ja ennetavate kontrollide nimekirja.

2.2 OWASP-i Top 10

OWASP-i Top 10 on projekt, mis koondab endasse 10 kõige kriitilisemat turvariski veebirakendustes [6]. 2017. aasta raporti väljavõttest selgub, et need 10 turvariski on järgnevad:

A1. *Süstimisrünne – süstimisrünne (näiteks SQL, NoSQL, LDAP) tekib kui rakendusse saadetakse ebausaldusväärseid andmeid käsu või päringu osana. Sellised andmed võivad käivitada soovimatuid käske või anda juurdepääsu andmetele ilma piisava loata* [6].

A2. *Autentimisvead – autentimise ja seanssi haldusega seotud funktsionaalsus on tihtipeale arendatud vigaselt, andes ründajale võimaluse manipuleerida*

paroolide, võtmete või seansi identifikaatoriga [2] ning kasutada muid rakenduse vigu, et jäljendada teiste kasutajate identiteeti ajutiselt või püsivalt [6].

A3. *Tundliku info kaitseta jätmine – paljud veebirakendused ja rakendusliidesed (API-d – ingl Application Programming Interface) ei kaitse korrektselt tundlikke andmeid, nagu näiteks finants-, tervishoiu- ja isikuandmeid. Ründaja võib varastada või muuta nõrgalt kaitstud andmeid, et teostada näiteks krediitkaardipettusi, identiteedivargusi või muid kuritegusid. Tundlikud andmed nõuavad täiendavat kaitset krüpteerimisel ja andmete edastusel. Eriti tähelepanelik tuleb olla veebilehitseja teel informatsiooni vahendamisel* [6].

A4. *XML välisolemrünne (XXE – ingl XML external entity attack) – paljud vanemad või ebakvaliteetselt konfigureeritud XML-l (laiendatav märgistuskeel – ingl Extensible Markup Language) baseeruvad veebilehed lubavad XML dokumentides välisolemiviiteid. Välisolemiviiteid saab kasutada avalikustamaks sisemisi faile, kasutades URI (ühtne ressursi-identifikaator – ingl Uniform resource identifier* [2]*) käsitlejat, sisemisteks failijagamisteks, sisemise pordi skaneerimiseks, koodi kaugkäituseks ja teenusetõkestuseks* [6].

A5. *Puudulik pääsu reguleerimine – autenditud kasutajatele seatud piirangud ei ole sageli õigesti rakendatud. Ründajad saavad neid vigu ära kasutada selleks, et pääseda ligi lubamatule funktsionaalsusele ja/või andmetele, näiteks ligipääs teiste kasutajate kontodele, tundlike failide vaatamine, teiste kasutajate andmete muutmine, juurdepääsuõiguste muutmine jne* [6].

A6. *Vigased turvaseaded – vigased turvaseadmed on kõige sagedamini esinev probleem. See on tavaliselt ebaturvaliste vaikeseadete, mittetäielike või ad hoc (spetsiaalselt selleks juhuks mõeldud [1]) konfiguratsioonide, avatud pilvtalletuse, vigaselt konfigureeritud HTTP (hüperteksti edastuse protokoll – ingl HyperText Transfer Protocol* [2]*) päiste ja tundlike andmeid sisaldavate veasõnumite tulemus. Kõik operatsioonisüsteemid, raamistikud, teegid ja rakendused peavad olema turvaliselt konfigureeritud ning õigeaegselt paigaldatud ja uuendatud* [6].

A7. *Murdskriptimine (XSS – ingl Cross-site scripting) – XSS-i puudused ilmnevad igakord kui rakendus kasutab ebausaldusväärseid andmeid uuel veebilehel ilma*

nõuetekohase valideerimiseta või skripte/HTML-i (hüperteksti märgistuskeel - ingl Hypertext Markup Language) elemente eemaldamata. Samuti kas siis, kui rakendus uuendab veebilehte kasutaja poolt esitatud andmetega, kasutades veebilehitseja API-t, mis võib luua HTML-i või JavaScripti. XSS võimaldab ründajal jooksutada skripte ohvri veebilehitsejas, mis võib kaaperdada kasutaja seanssi, moonutada veebilehti või suunata kasutaja pahatahtlikele veebilehtedele [6].

A8. *Ebaturvaline objektimine – ebaturvaline objektimine (objekti struktuuri ennistus biti- või baidijadast* [2]*) viib tihti koodi kaugkäituseni. Isegi kui objektimise puudused ei vii koodi kaugkäituseni, saab neid kasutada rünnakute, sealhulgas kordusrünnakute, süstimisrünnakute ja privileegide suurendamise rünnakute tegemiseks* [6].

A9. *Teadaolevalt auklike komponentide kasutamine – komponendid (teegid, raamistikud ja tarkvaramoodulid) töötavad samade privileegidega nagu rakendus. Kui haavatav komponent on kasutusel, võib selline rünnak hõlbustada tõsist andmete kaotust või serveri ülevõtmist. Rakendused ja kasutajaliidesed, mis kasutavad teadaolevalt auklike komponente, võivad õõnestada rakenduse kaitset ja võimaldada mitmesuguseid rünnakuid ja mõjutusi* [6].

A10. *Ebapiisav logimine ja seire – ebapiisav logimine ja seire, koos puuduva või ebaefektiivse integratsiooniga vahejuhtumitele reageerimisega, lubab ründajal süsteeme edasi rünnata, säilitada olukordi, pöörduda enamategi süsteemide poole ning teha väljavõtteid andmetest, moonutada või hävitada andmeid. Enamik rikkumistega seotud uuringuid näitavad, et rikkumise avastamiseks kulub tavaliselt üle 200 päeva ja tüüpiliselt avastavad rikkumisi välised osapooled, mitte ei leita neid logidest või seirest* [6].

2.3 OWASP-i rakenduste turvalisuse kontrollimise standard

„Rakenduste turvalisuse kontrollimise standard (OWASP ASVS - ingl *Application Security Verification Standard*) annab aluse veebirakenduste tehniliste turvakontrollide testimiseks ja haldab arendajatele mõeldud turvalise tarkvaraarenduse nõuete nimekirja [7]“. Nõuded töötati välja, pidades silmas järgmisi eesmärke:

- *Kasuta mõõdupulgana – eesmärgiks oli pakkuda rakenduste arendajatele ja tooteomanikele mõõdupuu, mis võimaldab hinnata nende rakenduste turvalisuse taset* [7].

- *Kasuta juhisena – eesmärgiks oli anda arendajatele juhiseid, kuidas arendada veebirakendusi nii, et oleks täidetud rakenduse turvanõuded* [7].

- *Kasuta hankimise ajal – eesmärgiks oli anda alus rakenduse turvanõuete täpsustamiseks lepingutes* [7].

Turvalise veebirakenduse loomise nõuete nimekiri jaguneb allpool esitatud kategooriatesse. Kõik eelnevalt mainitud OWASP Top 10 kriitilisemat turvaviga saavad kaetud, võttes arvesse OWASP ASVS nimekirja. See tähendab, et kui oleks korrektselt rakendatud turvanõuded vastavalt OWASP ASVS standardile, poleks esinenud OWASP Top 10 turvavigu. Turvalise veebirakenduse nõuded on:

V1. *Arhitektuuri, disaini ja ohtude modelleerimise nõuded* [8]

V2. *Autentimise verifitseerimise nõuded* [8]

V3. *Seansi juhtimise verifitseerimise nõuded* [8]

V4. *Pääsuõiguste verifitseerimise nõuded* [8]

V5. *Valideerimise, saniteerimise ja kodeerimise nõuded* [8]

V6. *Salvestatud krüptograafia verifitseerimise nõuded* [8]

V7. *Vigade käsitlemise ja logide verifitseerimise nõuded* [8]

V8. *Andmekaitse verifitseerimise nõuded* [8]

V9. *Teabevahetuse verifitseerimise nõuded* [8]

V10. *Ründekoodi verifitseerimise nõuded* [8]

V11. *Äriloogika verifitseerimise nõuded* [8]

V12. *Failide ja ressursside verifitseerimise nõuded* [8]

V13. *API-de ja veebiteenuste verifitseerimise nõuded* [8]

V14. *Konfiguratsiooni verifitseerimise nõuded* [8]

2.4 OWASP-i ennetavate kontrollide nimekiri

OWASP ennetavate kontrollide nimekiri (ingl *Proactive Controls*) on nimekiri turvatehnikatest, mis peaksid olema kaasatud igasse tarkvaraarenduseprojekti. OWASP- i ennetavate kontrollide nimekiri on esitatud tähtsuse järjekorras järgnevalt (esimene on kõige tähtsam):

C1. *Turvanõuete defineerimine – turvanõuded tuletatakse tööstusharu standarditest, seadustest ja varasemast kogemusest. Need on aluseks rakenduse kontrollitud turvafunktsionaalsusele, et tagada turvaomaduste täitmist. Korrektse turvanõuete defineerimise protsess koosneb neljast osast. Protsessi osadeks on: turvanõuete ja funktsionaalsuse valimine, dokumenteerimine, rakendamine ja korrektse rakendamise valideerimine* [9].

C2. *Turvaliste raamistike ja teekide kasutamine – turvaliste raamistike ja teekide kasutamine aitab ära hoida turvalisusega seotud disaini- ja rakendusvead. Kui arendaja üritab otsast peale ise arendada kogu funktsionaalsuse, ei pruugi tal olla piisavalt teadmisi, aega või rahalisi ressursse, et arendada turvanõuetekohane rakendus* [9].

C3. *Turvaline andmebaasi ligipääs – turvalise andmebaasi ligipääsu rakendades on oluline silmas pidada järgnevaid valdkondi: turvalised päringud, konfiguratsioon, autentimine ja suhtlus* [9].

C4. *Andmete kodeerimine ja elementide eemaldamine – andmete kodeerimine ja elementide eemaldamine on kaitsemeetodid, mis on mõeldud süstimisrünnakute tõkestamiseks. Kodeerimine tähendab erimärgi muutmist sellisele kujule, et see pole enam interpretaatorile ohtlik (näiteks tõlkides „<“ märgi „<“ tekstiks HTML-lehel). Elementide eemaldamine tähendab, et enne tähte/teksti lisatakse erimärk, et interpretaator ei tõlgendaks sisu vääralt. Näiteks lisades „\“ märgi enne jutumärke, et see tõlgitakse kui tekst mitte kui teksti lõpusümbol* [9].

C5. *Sisendparameetrite valideerimine – sisendparameetrite valideerimine on programmeerimistehnika, mis tagab, et tarkvarasüsteemi on võimalik sisestada ainult nõuetekohaselt vormindatud andmeid. Rakendus peab valideerima, et edastatavad andmed oleks nii süntaksilt kui ka semantiliselt korrektsed. Süntaktiliselt korrektne tähendab, et andmed on oodatud vormis. Näiteks kui rakendus lubab kasutajal sisestada neljakohalise arvu „konto ID“, et täita mingit tüüpi operatsiooni. Peab rakendus valmis olema selleks, et kasutaja üritas teostada süstimisrünnet ning valideerima, et sisestatud tekst oli neli märki pikk ja koosnes ainult numbritest (lisaks õige päringu parameetri kasutamisele). Semantiliselt korrektne tähendab, et sisestatud andmed on lubatud vahemikus. Näiteks alguskuupäev peab olema enne lõppkuupäeva. Valideerimine peab toimuma kindlasti ka serveri poolel* [9].

C6. *Digitaalse identiteedi rakendamine – digitaalne identiteet on kasutaja unikaalne esindatus veebis. Autentimine on protsess, mille käigus kontrollitakse, kas kasutaja on see, kes ta väidab end olevat. Seansihaldus on protsess, mille käigus server säilitab kasutaja autentimise oleku nii, et kasutaja saaks süsteemi kasutamist jätkata ilma uuesti autentimata* [9].

C7. *Pääsuõiguste reguleerimine – pääsuõiguste reguleerimine on konkreetse päringu lubamise või keelamise protsess sõltuvalt päringu sooritaja (kasutaja, programm, protsess) õigustest. See hõlmab ka privileegide andmise ja keelamise protsessi. Tuleb märkida, et autoriseerimine (juurdepääsu kontrollimine kindlale funktsionaalsusele või ressurssidele) ja autentimine (identiteedi kontrollimine) ei ole samaväärsed* [9].

C8. *Andmete kaitsmine – tundlikud andmed (paroolid, krediitkaardi andmed, tervishoiuandmed, isikuandmed jne) nõuavad erilist tähelepanu, eriti kui need teemad on käsitletud GDPR-s. Ründajad võivad rakendustest mitmel viisil andmeid varastada. Näiteks, kui tundlikud andmed saadetakse läbi avaliku võrgu, mis pole piisavalt turvaline, võib ründaja varastada teiste kasutajate andmeid. Samuti võib ründaja kasutada süstimisrünnet, et varastada paroole ja muid isikuandmeid rakenduse andmebaasist ja avalikustada saadud informatsiooni* [9].

C9. *Turvalisuse logimise ja seire rakendamine – logimist kasutavad paljud arendajad vigade tuvastamise ja diagnostika eesmärkidel. Turvalisuse logimine on põhimõttelt analoogne – turvainfo logimine rakenduse käitusajal. Seiramine on rakenduse ja turvalisuse logide läbivaatamine, kasutades automatiseerimist* [9].

C10. *Vigade ja erandite käsitlemine – vigade ja erandite käsitlemine on programmeerimiskontseptsioon, mis võimaldab rakendusel vastata erinevatele seisunditele (näiteks probleemid võrgu- või andmebaasiühendusega) erineval viisil. Erandite ja vigade korrektne käsitlemine on koodi usaldusväärsuse ja turvalisuse tagamiseks äärmiselt oluline. See peab toimuma rakenduse kõigis osades, nii kriitilise äriloogika, turvaelementide kui ka raamistike koodis* [9].

3 Olemasolevad lahendused

Lõputöö raames läbiviidud praktikumi eesmärgiks oli lasta tudengitel parandada üldlevinud turvaauke. Autorile teadaolevalt pole varem analoogseid praktikume Tallinna Tehnikaülikoolis korraldatud. Samas on olemas teisigi sihilikult ebaturvalisi rakendusi, mis keskenduvad OWASP-i poolt välja toodud turvariskidele.

OWASP JuiceShop[1] on veebirakendus, mis on tehtud teadlikult ebaturvaline, et illustreerida kriitilisemaid turvaauke. JuiceShopis on väga palju erineva tasemega ülesandeid ja neid võib lahendada nii veebilehitsejas kui enda arvutis, sest see on avatud lähtekoodiga projekt. Samas ei väljenda JuiceShop selle lõputöö eesmärke, sest ülesanneteks on seal vaid turvaaugu tuvastamine, kuid mitte selle parandamine. Lisaks on kogu kood kirjutatud JavaScriptis (Node.js[2] ja Angulari[3] rakendus), aga selle lõputöö raames taheti tutvustada turvaauke Tallinna Tehnikaülikooli informaatika esimese kursuse tudengitele tuttavas keeles – Javas.

RangeForce[4] on firma, mis pakub I-Tee baasil erinevaid virtuaallaboreid küberturvalisuse õppimiseks. „I-Tee on kauglaborite süsteem, mis põhineb *Ruby on Rails*il ja kasutab VirtualBoxiga virtualiseerimist. I- Tee on välja töötanud Eesti Infotehnoloogia Kolledž (praegune TalTech IT Kolledž)“ [10]. *Ruby on Rails* on raamistik veebirakenduste loomiseks. Autor tutvus laboritega, kus pidi vigu tuvastama, mitte parandama. Laborid, kus on võimalik tegeleda ka turvavigade parandamisega, on tasulised, mistõttu jäid need vaatlusest välja. Kuivõrd antud lõputöö eesmärk oli anda põgus ülevaade suurimatest turvariskidest, pole RangeForce siinkohal otstarbekas, sest üldiselt minnakse ühe teemaga süvitsi, näiteks on eraldi praktikum vaid murdskriptimise teemal.

[1] https://juice-shop.herokuapp.com

[2] https://nodejs.org/en

[3] https://angular.io

[4] https://rangeforce.com

Avatao[1] on keskkond turvavigade leidmiseks ja parandamiseks. Keskkond on samuti tasuline ning seal on ülesandeid erinevates programmeerimiskeeltes (ka Javas), kuid üldiselt pole ligipääsu kogu lähtekoodile, vaid mingile osale sellest ja paranduste tegemine nõuab kindlat lähenemist (nt konkreetse teegi, objekti, meetodi vms kasutamist), mitte ei võimalda näiteks koodi ümber struktureerimist või alternatiivsete lahenduste rakendamist.

Lisaks on veel mõningaid mitte nii levinud platvorme, millest paljud baseeruvad aegunud tarkvaral. Enamasti ei võimalda need platvormid õppejõul tudengeid hinnata, abistada või anda neile tagasisidet. Palju on ka erinevaid CTF-platvorme (ingl *Capture the flag*), kuid kaitsmisteemalisi lahendusi on märksa vähem.

[1] https://avatao.com

4 Praktikumi tehniline kirjeldus

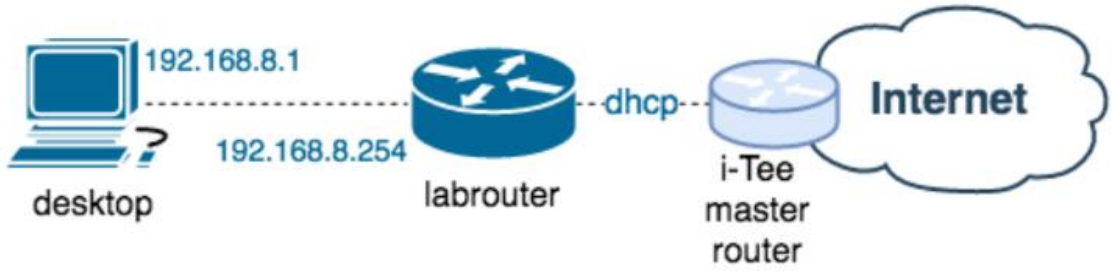
Käesolev peatükk kirjeldab lõputööna valminud praktikumi ülesehitust. Üldjoontes jagunes praktikumi ülesehitus kolme osasse:

1. Praktikum toimus virtuaalmasinal I-Tee keskkonnas[1].
2. Virtuaalmasinas asus turvaaukudega veebirakendus
3. Rakenduse parandatud lähtekood lisati GitLabi hoidlasse, kus rakendus pideva integratsiooni konveier

Praktikumi tööjuhend on väljatoodud lisades (vt Lisa 1).

4.1 Virtuaalmasin

Virtuaalmasina ülesehitus on esitatud joonisel 1, kus *desktop* on töölaud (masin, millel tudengid praktikumi lahendasid), *labrouter* on ruuter töölaua masina ja I-Tee keskkonna I-Tee *master router*-i vahel. I-Tee *master router* on ülemruuterseade, mis ühendub internetti.



Joonis 1. Virtuaalmasina struktuur [11].

Tudengitele kättesaadaval *desktop* masinal oli 3 gigabaiti muutmälu ja 20 gigabaiti kõvaketta mahtu. Masina seadistamiseks kasutati ette antud virtuaalmasina põhja, mille operatsioonisüsteemiks oli Ubuntu MATE 16.04.

[1] https://elab.cs.ttu.ee

Masinale lisati:

- Java 8 – serveripoolne programmeerimiskeel
- Intellij IDEA - integreeritud programmeerimiskeskkond, mõeldud eelkõige Java baasil arendusteks
- Node.js – avatud lähtekoodiga tarkvarasüsteem
- NPM – paketihaldur Node.js teekide jaoks (ingl *Node Package Manager*)

4.2 Turvaaukudega veebirakendus

Praktikumis esitatud turvaaugud valiti välja OWASP-i Top 10 projektist. Järgnevalt on esitatud iga rakenduse vea ja OWASP-i Top 10 projekti turvaaugu vaheline seos. Lisaks on välja toodud, millise OWASP rakenduste turvalisuse kontrollimise standardi punkti vastu on eksitud ning milline OWASP ennetavate kontrollide nimekirja punkt oleks aidanud seda viga ära hoida. Sealjuures on põgusalt kirjeldatud turvaaugu olemust, kuid jäetud välja lahenduse kirjeldus, et antud lõputööst poleks võimalik lugeda välja vastuseid, kui see praktikum peaks toimuma ka järgmistel aastatel.

4.2.1 Turvaaukude kategooria valimine

1. Esimene rakenduse viga (vt Lisa 1 alapunkt 14.1) esindab OWASP-i turvaauku A3. Rakenduse kasutajaliidesesse saadetakse tunduvalt rohkem informatsiooni, kui seal kuvatakse. Selline andmevahetus võimaldab kasutajal kätte saada delikaatseid andmeid teiste kasutajate kohta. Siinkohal on märgata ka eksimist GDPR-i vastu, mis võib muuhulgas ettevõttele tuua rahalise trahvi. Seda turvaviga oleks saanud vältida, kui oleks õigesti rakendanud OWASP *Proactive Controls* C8 nõuet, mis ütleb, et andmed (eriti tundlikud andmed) peaksid olema hoolikalt kaitstud. Eksitud on OWASP ASVS andmekaitse verifitseerimise nõuete (V8) ja API-de ja veebiteenuste verifitseerimise nõuete (V13) vastu.

2. Teine rakenduse viga (vt Lisa 1 alapunkt 14.2) esindab OWASP-i turvaauku A8. Kasutajal on võimalus muuta välju, millele tal ei tohiks ligipääsu olla. Selle vea oleks saanud ära hoida, kui oleks õigesti rakendanud OWASP *Proactive Controls* C5 nõuet, mis ütleb, et kõik sisendparameetrid tuleb valideerida. Eksitud on

OWASP ASVS pääsuõiguste verifitseerimise nõuete (V4) ja API-de ja veebiteenuste verifitseerimise nõuete (V13) vastu.

3. Kolmas rakenduse viga (vt Lisa 1 alapunkt 14.3) esindab OWASP-i turvaauku A7 ehk murdskriptimine. Antud olukorras on võimalus XSS rünnakuga tekitada ettevõttele äriline kahju, kasutades rakenduse nõrkust soovitud informatsiooni kuvamiseks või selles muudatuste tegemiseks endale sobival viisil. Selle vea oleks saanud ära hoida, kui oleks õigesti käsitlenud OWASP *Proactive Controls* punkti C4, mis ütleb, et andmed tuleb kodeerida ja neist tuleb eemaldada rakendusele ohtlikud elemendid ning punkti C5, mis ütleb, et sisendparameetreid tuleb alati korrektselt valideerida. Eksitud on OWASP ASVS valideerimise, saniteerimise ja kodeerimise nõuete (V5) vastu.

4. Neljas rakenduse viga (vt Lisa 1 alapunkt 14.4.) esindab OWASP-i turvaauku A5. Siinkohal õnnestub kasutajal manipuleerida teiste kasutajate andmetega, sest rakenduses pole korrektselt arendatud pääsuõigused, ja sellega tekitada kahju nii ettevõttele kui teistele kasutajatele. Selle vea oleks saanud ära hoida, kui oleks õigesti rakendanud OWASP *Proactive Controls* punkti C7, mis hõlmab pääsuõiguste reguleerimist. Eksitud on OWASP ASVS pääsuõiguste verifitseerimise nõuete (V4) vastu.

5. Viies rakenduse viga (vt Lisa 1 alapunkt 14.5) esindab OWASP-i turvaauku A2. Kuigi sedalaadi turvaaugud tulevad välja ainult juhul, kui süsteemist on lekkinud andmed, näiteks andmebaas on avalikustunud, siis võimaldab see siiski süsteemi andebaasile ligipääsu omavatel inimestel enda teadmisi pahatahtlikel eesmärkidel ära kasutada. Näiteks saades teada mõne inimese parooli, on tal võimalik proovida seda parooli mõnes teises süsteemis rakendada ja pole ka väga ebatõenäoline, et andmebaas lekib. Seda on juhtunud ka suurte süsteemidega. Näiteks lekkisid LinkedIn-i (töö- ja äriinfoportaal) paroolid 2012. aastal [12]. Selle vea oleks saanud ära hoida, kui oleks õigesti rakendanud OWASP *Proactive Controls* punkti C6, mis hõlmab digitaalse identiteedi rakendamist. Eksitud on OWASP ASVS autentimise verifitseerimise nõuete (V2) vastu.

4.2.2 Tehnilised andmed

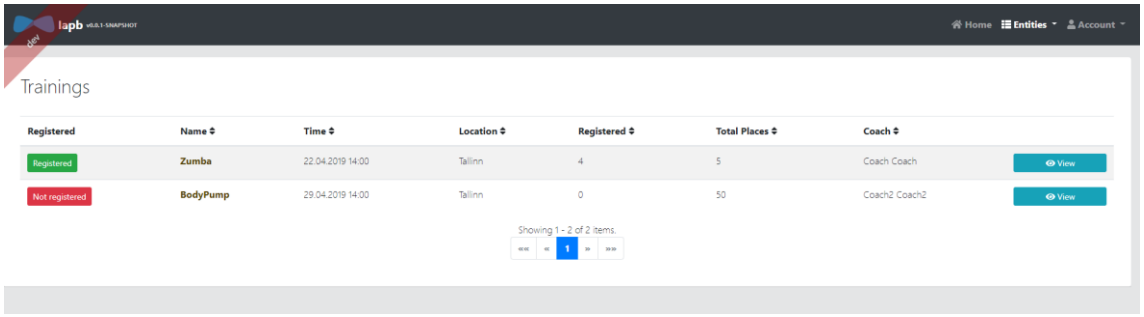
Praktikumi raames valminud veebirakendus (vt Joonis 2) loodi kasutades JHipster platvormi ja Gradle-it ning selle serveripoolseks keeleks on Java, andmebaasiks residentbaas H2, kasutajaliidese arenduseks kasutati HTML-i ja Angulari. Täpsemalt oli serveripoolseks arhitektuuriks kasutusel Spring Boot raamistik, kasutades REST-i ja Hibernate ORM-i.

JHipster - *avatud lähtekoodiga rakenduste generaator Spring Boot + Angular / React projektide loomiseks* [13]

Gradle - *avatud lähtekoodiga ehituse automatiseerimise tööriist* [14]

REST (ingl *Representational State Transfer*) - *arhitektuuri stiil veebi arvutisüsteemide vaheliste standardite pakkumiseks, mis lihtsustab süsteemide omavahelist suhtlemist* [15]

Hibernate ORM (ingl *Object-relation mapping*) - *teek, mis pakub rakendustele, teistele teekidele ja raamistikele objekt-relatsioonvastendust* [16].



Joonis 2. Kuvatõmmis turvaaukudega veebirakendusest.

4.2.3 Funktsionaalsed andmed

Lõputöö tulemusena valmis turvaaukudega rakendus, mis kujutab endast broneerimissüsteemi treeningutele (vt Joonis 2). Rakenduse kasutajad on treenerid ja treeningutest osa võtvad tavakasutajad.

Rakenduses on 3 rolli: treener, tavakasutaja ja administraator. Järgnevalt on esitatud käesoleva lõputööga seotud funktsionaalsed nõuded, mis puudutavad treeningute domeeni.

Rakenduse nõuded tavakasutajale:

- Peab võimaldama näha nimekirjavaadet treeningutest
- Peab võimaldama näha detailvaadet konkreetse treeningu kohta
- Peab võimaldada registreeruda treeningule

Rakenduse nõuded treenerile ja administraatorile:

- Peab võimaldama näha nimekirjavaadet treeningutest, kuhu praegune kasutaja on märgitud treenerina (administraatori puhul nimekirjavaadet kõikidest treeningutest)
- Peab võimaldama näha detailvaadet konkreetse treeningu kohta
- Peab võimaldama treeningute lisamist
- Peab võimaldada treeningute muutmist
- Peab võimaldama treeningute kustutamist

4.3 GitLab CI konveier

Pideva integratsiooni (CI – ingl *continuous integration*) eesmärk on jagada kood ühisesse koodihoidlasse. Käesoleva lõputöö raames kasutati GitLab CI konveierit, et hinnata tudengite tööd. Veebirakenduse lähtekoodiga on kaasas .gitlab-ci.yml fail (vt Joonis 4), mis kirjeldab konveierit, mida hakkab töötlema GitLab Runner. „GitLab Runner on avatud lähtekoodiga projekt, mida kasutatakse tööde käivitamiseks ja tulemuste saatmiseks GitLabi“ [17] (vt Joonis 3).

Selle praktikumi raames loodud konveieris on kolm tööd. Esimese töö käigus ehitatakse rakenduse paigaldusühik (.war fail) ja luuakse Dockeri konteiner. Docker on programm, mis võimaldab konteinerdust[1]. Konteinerdamine on „.hajusrakenduste virtualiseerimise meetod: rakendus kapseldatakse koos oma töökeskkonnaga (failid, teegid jms) konteinerisse, täielikku virtuaalmasinat loomata ja operatsioonisüsteemi

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

ühiskasutusega“ [2]. Teise töö käigus käivitatakse Dockeri konteiner. Kolmanda töö käigus testitakse rakendust üksustestidega.

Joonisel 3 on esitatud praktikumi esitamise töökäik. Esialgu lisab tudeng koodi GitLabi salve (*code pushed to GitLab*). Seejärel otsib GitLab sõltuvalt .gitlab-ci.yml faili sisule vastava GitLab Runneri (*GitLab trigger GitLab Runner*) ning siis hakkab GitLab Runner töötlema konveierit, mis on kirjeldatud .gitlab-ci.yml failis (*install dependencies, run tests job*).

Joonis 3. GitLab CI konveieri tööprotsess [18].

```
stages:
    - build
    - deploy
    - test

build:
    tags:
        - iti0103-2019-shell-hipster
    stage: build
    script:
        - echo "Building"
        - chmod 755 ./gradlew
        - ./gradlew clean bootWar -
Dorg.gradle.java.home=/usr/lib/jvm/java-1.8.0-openjdk-amd64
        - mv build/libs/*.war src/main/docker/
        - cd src/main/docker
        - docker build -t iapb .

deploy:
    tags:
        - iti0103-2019-shell-hipster
    stage: deploy
    script:
        - echo "Deploying"
        - docker stop iapb
        - docker rm iapb
        - docker run -d -p 8080:8080 --name iapb iapb

test:
    tags:
        - iti0103-2019-shell-hipster
    stage: test
    script:
        - echo "Testing"
        - chmod 755 ./gradlew
        - ./gradlew test
```

Joonis 4. GitLab CI konveieri kood.

5 Tööprotsess

Käesolev töö oli valdavalt praktiline. See peatükk kirjeldab eelkõige lõputöö praktilise osa valmimist, mis sisaldab rakenduse arendamist, virtuaalmasina ja GitLab CI seadistamist, praktikumi tudengitele tutvustamist ning tagasiside andmist.

5.1 Rakenduse analüüs

Esimene iteratsioon oli tausta uurimine ja analüüs. Autor töötas välja lahenduse praktikumi struktuurile, valis välja käsitletavad turvaaugud ja rakenduse narratiivi, et praktikumi lahendamine oleks kaasahaaravam. Veebirakenduste turvalisus valiti praktikumi teemaks, kuna lõputöö kirjutamise ajal polnud taolisele spetsiifikale suunatud teemasid praktilises lähenduses informaatika õppekavas käsitletud. Teooriaga tutvudes valiti turvaukude põhjaks OWASP Top 10 projekt.

5.2 Rakenduse arendamine

Teises iteratsioonis tegeleti tarkvaraarendusega. Oluline oli siduda turvaaugud reaalelulisse konteksti, et tudengid mõistaksid paremini turvaaukude sisu ning tunneksid need edaspidi lihtsamini ära. Seetõttu otsustati vältida isoleeritud mõnerealist koodi ja luua selle asemel terviklik turvaaukudega veebirakendus, kus tudengitel oli võimalik ise haavataid kohti näha, läbi mängida ja testida. Kuivõrd veebirakenduse loomine polnud praktikumi peamiseks eesmärgiks, vaid abivahend turvaaukude illustreerimisel, otsustati kasutada JHipster rakenduse generaatorit, et hoida kokku arendusele kuluvat aega ja vältida kontrollimatute turvaaukude teket.

5.3 Virtuaalmasina konfigureerimine

Kolmandas etapis keskendus autor virtuaalmasina konfigureerimisele. Praktikumi ülesehitus nägi ette, et tudengite ainsaks tööülesandeks on turvaaukude parandamine ja kontrollitakse ainult turvaaukude parandamist hõlmavat tööd. Võimaldamaks tudengitel eelseadistatud keskkonna kasutamine, otsustati virtuaalmasina kasuks. Virtuaalmasina

jaoks vajalik põhi oli olemas, kuid vajas olulist kohandamist. Selline lähenemine tingis oodatavast märkimisväärselt suurema töömahu autori jaoks, kes teostas lisaks vajaliku tarkvara paigaldamisele ka kõvaketta mahu suurendamise ja konfigureeris võrguseadeid.

5.4 Testkasutajatega praktikumi läbi töötamine

Praktikumi testimisel osales kaks testkasutajat, kellest üks oli tarkvaraarendaja ja teine rakenduste testija. Mõlemad testkasutajad puutuvad igapäevatöös kokku veebirakenduste arendamise või testimisega.

Arendaja eesmärk oli proovida tööjuhendi põhjal määrata turvavea kategooria ja leida lahendus vea parandamiseks. Selle käigus selgus, et tööjuhendis on vaja teha korrektuure, et tudengitel oleks lihtsam mõista, mida vea parandamisel oodatakse.

Testija eesmärk oli tuvastada rakenduses olevad turvavead ilma tööjuhendit nägemata. Tal oli selleks aega 10 minutit. Niivõrd lühikese aja jooksul õnnestus leida kaks viga viiest, mis ilmestab selgelt, et antud praktikumis kasutatud turvavead on ohtlikud. Ülejäänud kolm eeldasid natuke sügavamat tutvumist rakendusega ning abivahendi kasutamist. Siinkohal on mõeldud mõnda rakenduse päringute jälgimiseks ja muutmiseks kasutatavat tarkvara, näiteks Postman[1].

5.5 Testide kirjutamine

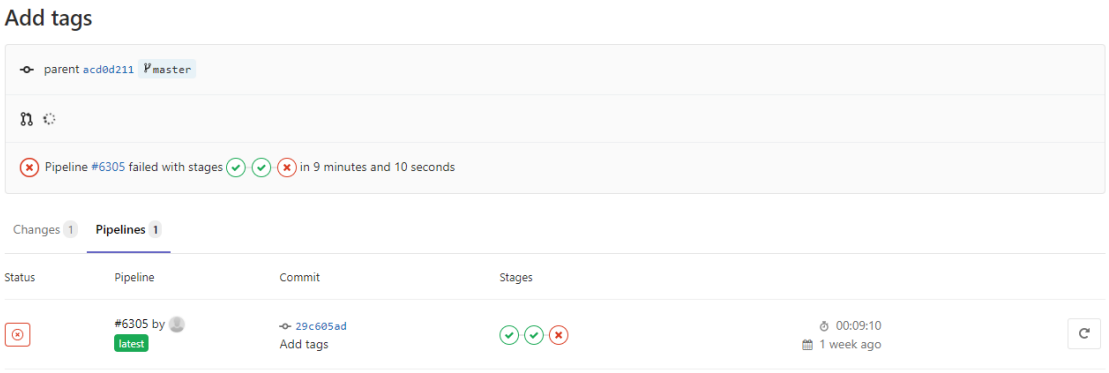
Viiendas iteratsioonis kirjutas autor üksustestid Javas, mis hiljem rakendusid praktikumi kontrollimise faasis. Esialgu oli plaanis kasutada REST-assured[2] raamistikku testimiseks, mis võimaldab testida REST-i rakendusliideseid. Pärast rakenduse tutvustamist testkasutajatele selgus, et rakendus on piisavalt keeruline ja testid võiksid olla tudengitele abiks praktikumi lahendamisel. Otsustati muuta testid tudengitele nähtavaks. Selleks kirjutati valmis Java üksustestid, mis lisati veebirakenduse lähtekoodi.

[1] https://www.getpostman.com

[2] http://rest-assured.io

5.6 GitLabi konveieri konfigureerimine

Kuuendas faasis toimus GitLabi pideva integratsiooni seadistamine, et hilisem praktikumide kontrollimine oleks sujuvam ja kiirem. Näide GitLab CI töödest on esitatud joonisel 5. GitLabis oli juba olemas kood, mis tudengite tööd vastavalt hoidla nimele kokku kogus ja ühte gruppi tõstis.



Joonis 5. Ekraanitõmmis GitLab CI konveieri tööst.

5.7 Loeng tudengitele rakenduse tutvustamiseks

Seitsmendas faasis toimus loeng, kus töö autor osales, et tutvustada praktikumi ja teha lühike demonstratsioon rakendusest ning vastata tudengite küsimustele. Seejärel said tudengid 5 päeva aega kodutöö lahendamiseks.

5.8 Tööde kontrollimine ja tagasiside andmine

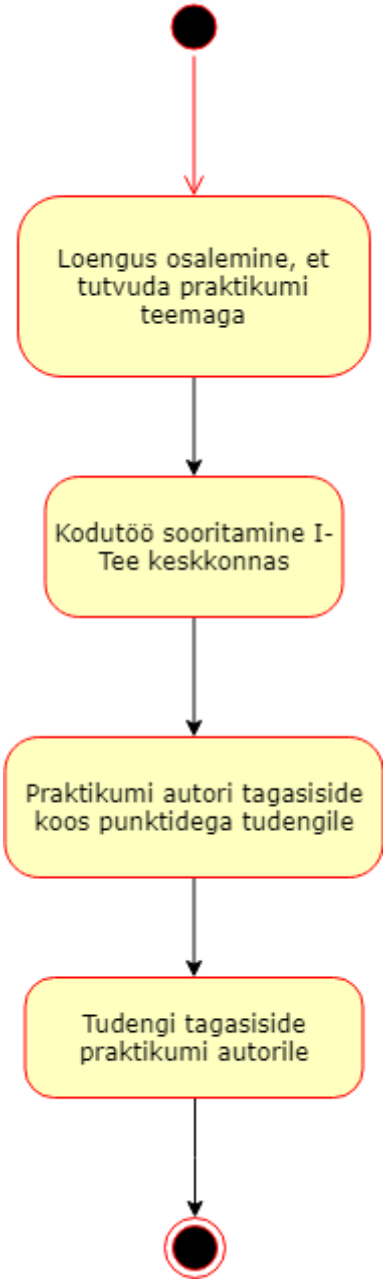
Järgnes tööde kontrollimine lähtudes hindamisel Java üksustestide tulemustest (vt Joonis 6). Kasutati *git diff* funktsionaalsust, mis võimaldab näha erinevust esialgse Giti (versioonihaldustarkvara) peaharu ja muudetud haru vahel. Pärast hindamist andis autor igale tudengile individuaalse tagasiside, lisades sinna soovitusi edaspidiseks ja kasulikke materjale, millega tutvuda.

```
DONE  Compiled successfully in 39064ms8:33:50 PM

   798 modules

> Task :processResources
> Task :classes
> Task :compileTestJava
> Task :processTestResources
> Task :testClasses
> Task :test

ee.ttu.iapb.web.rest.AccountResourceIntTest > testPasswordEncoding FAILED
    org.junit.ComparisonFailure at AccountResourceIntTest.java:817

ee.ttu.iapb.web.rest.TrainingResourceIntTest > getAllTrainings FAILED
    java.lang.AssertionError at TrainingResourceIntTest.java:394

ee.ttu.iapb.web.rest.TrainingResourceIntTest >
deleteTrainingCreatedByOtherCoach FAILED
    java.lang.AssertionError at TrainingResourceIntTest.java:406

ee.ttu.iapb.web.rest.TrainingResourceIntTest >
changeTheDataByRegisteringYourselfToTheTraining FAILED
    org.junit.ComparisonFailure at TrainingResourceIntTest.java:437

2019-05-13 20:34:58.655  WARN 2811 --- [      Thread-15]
o.s.b.f.support.DisposableBeanAdapter    : Invocation of destroy method
failed on bean with name 'inMemoryDatabaseShutdownExecutor':
org.h2.jdbc.JdbcSQLException: Database is already closed (to disable
automatic closing at VM shutdown, add ";DB_CLOSE_ON_EXIT=FALSE" to the db
URL) [90121-197]

153 tests completed, 4 failed
> Task :test FAILED
```

Joonis 6. Üksustestide tulemus GitLabis.

6 Tööprotsess tudengi vaatest

Järgnevalt on esitatud praktikumi sooritamise protsess tudengi seisukohalt (vt Joonis 7).



Joonis 7. Tudengi tööprotsessi tegevusskeem.

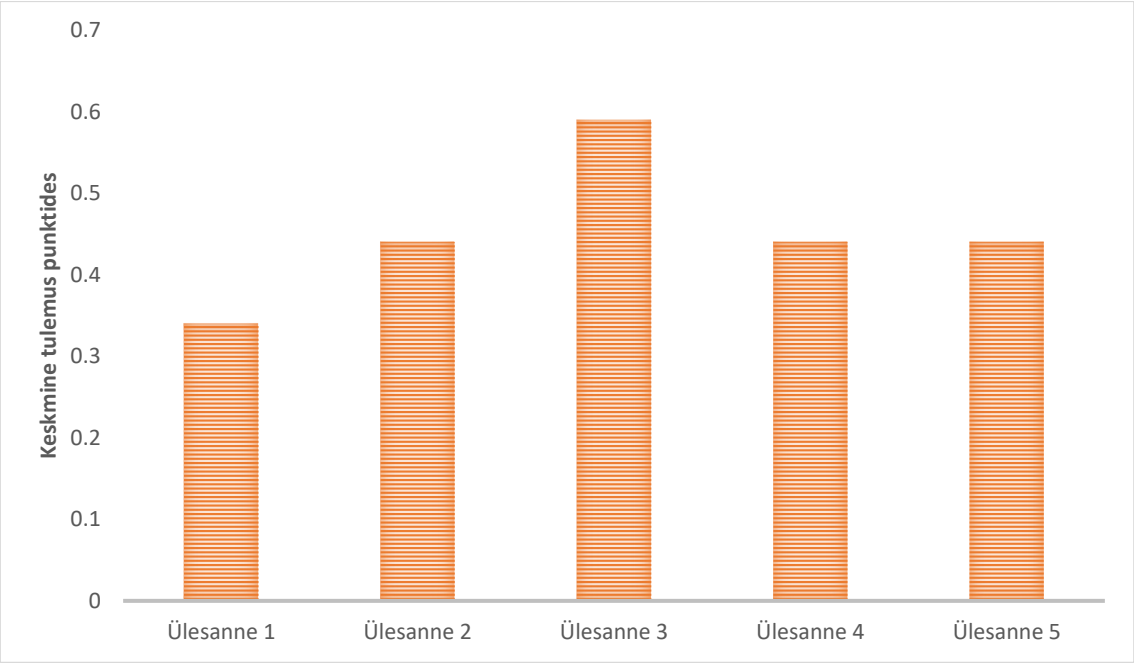
7 Tulemused

Praktikumi tööjuhend täpsema infoga vigade iseloomust on välja toodud lisades (vt Lisa 1). Tööde hindamine toimus alljärgnevalt:

- Muudatus on lisatud GitLabi salve - 1 punkt.
- Kõik vead on parandatud – maksimaalselt 5 punkti. Kokku oli 5 ülesannet. Täieliku lahenduse eest sai maksimaalselt 1 punkti, pooliku lahenduse eest 0,5 punkti.
- Kuni 3 boonuspunkti lisatöö eest. Boonuspunkti saamiseks pidi tudeng välja tooma rakendusest mõne juhendis kirjeldamata turvaaugu ning suutma selle ka parandada.

Selliseid tudengeid, kes esitasid suvaliselt muudetud lähtekoodi, tegemata parandusi, oli kokku seitse ja nad on jäetud edaspidisest analüüsist välja. Tabelis 1 on esitatud tulemused iga tudengi ja parandatud turvavigade (iga ülesanne väljendas ühte turvaviga) kaupa. Aritmeetiline keskmine tulemus oli 3,38 punkti. Maksimum oli 6 punkti ilma boonusteta ja 9 punkti koos boonustega.

Joonisel 8 on esitatud praktikumi ülesannete tulemuste aritmeetilised keskmised iga turvavea kohta. Tulemused on ümardatud kahe komakoha täpsusega. Esile kerkivad ülesanded 1, 3 ja 5 (vt Joonis 8). Esimene, sest osutus tudengitele keskmisest keerulisemaks. Seda ülesannet ei lahendanud keegi maksimumpunktidele, sest enamik tudengite lahendus parandas küll selle konkreetse vea, kuid olemasolev funktsionaalsus kannatas (kadus treeningule registreerumise võimalus). Kolmas ülesanne oli ainus, mis eeldas muudatuste tegemist kasutajaliidese tasemel, mitte serveri poolel, ja nagu näha jooniselt 8 on see kõige paremini lahendatud ülesanne. Viienda ülesande lahendamisel kasutasid paljud tudengid oma algoritmi. Selle eest said nad täispunktid, kuid tagasisidesse lisati neile soovitus edaspidi kasutada juba eksisteerivaid lahendusi.



Joonis 8. Praktikumi tulemused ülesannete kaupa.

Tabel 1. Praktikumide tulemused turvavigade kaupa.

Tudeng	**Viga 1**	**Viga 2**	**Viga 3**	**Viga 4**	**Viga 5**	**Kokku**	
Tudeng 1	0.5	1	1	0.5	1	6	Boonus +1
Tudeng 2	0.5	1	1	1	1	5.5	
Tudeng 3	0.5	1	1	1	1	5.5	
Tudeng 4	0.5	1	0	1	1	5.5	Boonus +1
Tudeng 5	0.5	1	1	0.5	1	5	
Tudeng 6	0.5	1	1	0	0	3.5	
Tudeng 7	0.5	0	0.5	1	0	3	
Tudeng 8	0	0	0	1	1	3	
Tudeng 9	0.5	1	0	0	0	2.5	
Tudeng 10	0.5	0	1	0	0	2.5	
Tudeng 11	0.5	0	0.5	0.5	0	2.5	
Tudeng 12	0	0	1	0.5	0	2.5	
Tudeng 13	0	0	1	0	0	2	
Tudeng 14	0	0	0	0	1	2	
Tudeng 15	0.5	0	0	0	0	1.5	
Tudeng 16	0	0	0.5	0	0	1.5	

8 Tagasiside tudengitelt

Pärast praktikumi sooritamist küsiti tudengitelt tagasisidet. Tagasiside andis 32 tudengit, kellest 23 sooritas praktikumi vähemalt ühele punktile. Neil paluti vastata kolmele kriteeriumile kuuepunktilisel Likert skaalal (0 kuni 5). Tulemused on välja toodud tabelites 2, 3 ja 4. Tudengid on tagasiside tabelis (vt Tabel 2) vastavuses tulemuste tabeliga (vt Tabel 1). Tabel 2 väljendab nende tudengite tagasisidet, kes praktikumi lahendasid. Tabelis 3 on esitatud nende tudengite tagasiside, kes vaid esitasid praktikumi (1 punkt esitamise eest). Tabel 4 väljendab nende tudengite tagasisidet, kes ei esitanud praktikumi. Kriteeriumite jaotus oli järgnev:

- Kas oli huvitav? (vt Tabel 2, Tabel 3 ja Tabel 4 veerg „huvitav“) – 0 tähendas, et praktikum oli ülimalt igav ja sedalaadi õpetamisviis on ebasobiv ning 5 tähendas, et praktikum oli igati eeskujulik, kaasahaarav ja selliseid kodutöid võiks rohkem olla

- Kas oli keeruline? (vt Tabel 2, Tabel 3 ja Tabel 4 veerg „keeruline“) – 0 tähendas, et labor oli ülimalt lihtne; 1 tähendas, et see oli väga lihtne; 2 tähendas, et see oli pigem lihtne; 3 tähendas, et see oli pigem keeruline; 4 tähendas, et see oli päris keeruline ja 5 tähendas, et see oli väga keeruline

- Kas oli kasulik? (vt Tabel 2, Tabel 3 ja Tabel 4 veerg „kasulik“) – 0 tähendas, et labor oli ebavajalik ja tudeng ei usu, et sellist asja kunagi elus kasutama hakkab ning 5 tähendas, et sellest laborist on tudengi hinnangul tema tulevases või juba praeguses elus palju kasu

Lisaks oli iga kriteeriumi vastuseks võimalik märkida „Ei kommenteeri“, millele lisaks oli võimalik eraldi lahtris oma hinnangut täpsustada.

Tabel 2. Praktikumi lahendanud tudengite tagasiside praktikumile.

Tudeng	**Huvitav**	**Keeruline**	**Kasulik**
Tudeng 1	5	1	5
Tudeng 2	5	3	5
Tudeng 3	4	5	3
Tudeng 4	3	2	3
Tudeng 5	Ei andnud tagasisidet		
Tudeng 6	5	4	4
Tudeng 7	2	5	2
Tudeng 8	3	4	4
Tudeng 9	4	4	4
Tudeng 10	5	4	5
Tudeng 11	5	5	3
Tudeng 12	2	5	3
Tudeng 13	1	5	1
Tudeng 14	2	5	2
Tudeng 15	4	5	4
Tudeng 16	4	5	3

Tabel 3. Praktikumi esitanud tudengite tagasiside praktikumile.

Tudeng	**Huvitav**	**Keeruline**	**Kasulik**
Tudeng 17	3	4	2
Tudeng 18	Ei kommenteeri	5	Ei kommenteeri
Tudeng 19	3	5	3
Tudeng 20	1	3	1

Tabel 4. Praktikumis mitteosalenud tudengite tagasiside praktikumile.

Tudeng	**Huvitav**	**Keeruline**	**Kasulik**
Tudeng 21	3	3	3
Tudeng 22	4	4	5
Tudeng 23	5	5	5
Tudeng 24	3	5	2
Tudeng 25	5	5	5
Tudeng 26	2	2	2
Tudeng 27	3	3	3
Tudeng 28	4	4	4
Tudeng 29	3	3	3
Tudeng 30	Ei kommenteeri	Ei kommenteeri	Ei kommenteeri
Tudeng 31	Ei kommenteeri	Ei kommenteeri	Ei kommenteeri
Tudeng 32	Ei kommenteeri	Ei kommenteeri	Ei kommenteeri
Tudeng 33	Ei kommenteeri	Ei kommenteeri	Ei kommenteeri

8.1 Kommentaarid tudengitelt

Kahest tudengist, kes praktikumis ei osalenud, mainis üks märkusena, et ta pole laboriga tutvunud ning teine, et tal oli probleeme I-Tee keskkonna jõudlusega ja ei saanud seetõttu praktikumi sooritada. Kommentaari lisasid ka kolm lahenduse esitanud tudengit, kellest üks (tudeng 6) kinnitas samuti, et tal esines I-Tee keskkonna jõudlusega probleeme. Teine märkis, et tal oli huvitav näha kellegi teise poolt loodud projekti (tudeng 20). Kolmas kiitis ülesande praktilisust, kuid temale oli turvavigade parandamine ületamatult keeruline, sest puudusid veebirakenduste arendamise oskused. Ta oleks soovinud, et oleks rohkem juhendeid kesiste oskustega tudengite jaoks (tudeng 11).

8.2 Tagasiside kokkuvõte

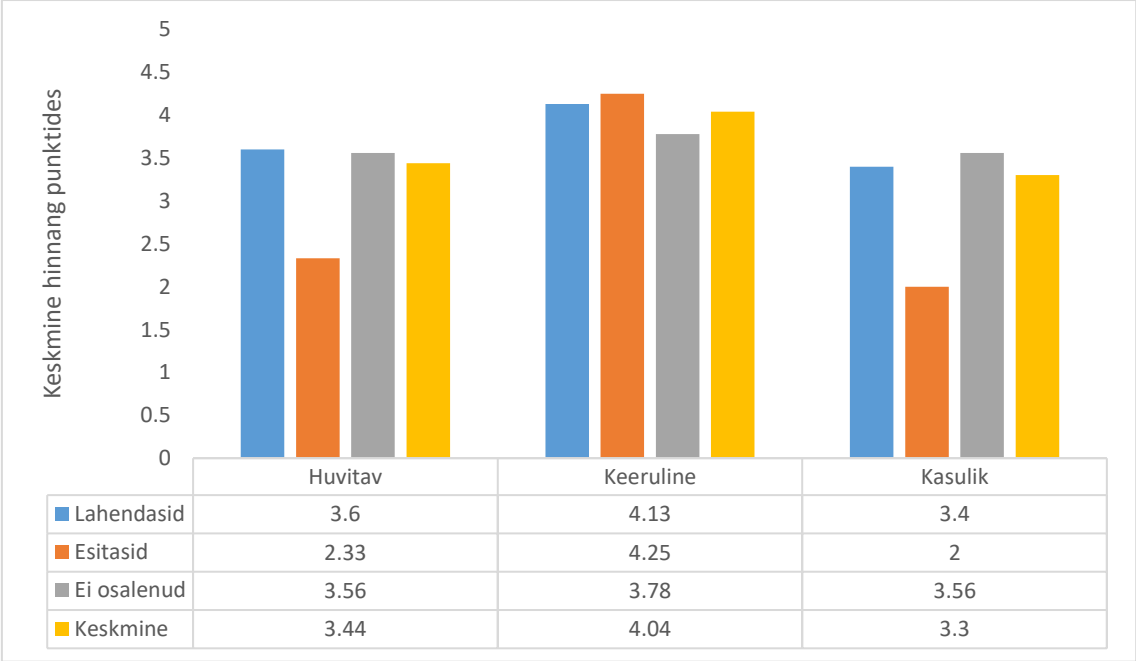
Tagasiside tulemused on esitatud joonisel 9. Tulemustest kajastub, et tudengid pidasid seda praktikumi väga raskeks, hinnates praktikumi keerulisust keskmiselt 4,04 punkti kuuepunktilisel Likerti skaalal (0 kuni 5). Üldiselt need tudengid, kes praktikumi vähemalt osaliselt lahendasid, pidasid seda huvitavaks ja kasulikuks. Need tudengid, kes tööd ei esitanud, hindasid reeglina praktikumi ebahuvitavaks, mittekasulikuks ja samal ajal keeruliseks. Kahjuks lisasid kommentaari vaid mõned üksikud tudengid, mis raskendab hinnangute analüüsimist. Tõenäoliselt mõjutas keerukus ka mõne tudengi arvamust kasulikkuse osas – on raske hinnata midagi kasulikuks, kui see on nii keeruline, et seda lahendada ei osata. Praktikum ei olnud kohustuslik ja töö autorile ei ole teada, kui palju tudengeid üldse praktikumiga tutvus või seda lahendada üritas. Samuti pole teada praktikumi lahendamisest loobumise põhjused ega ka võimalus, et tudengite suhtumine praktikumisse oli mõjutatud sellest, et praktikum ei olnud kohustuslik.

Kommentaarina toodi välja, et I-Tee keskkonna jõudlusega esines probleeme. Seekord sai otsustatud I-Tee keskkonna kasuks, sest see võimaldab praktikumi autoril kõik vajaliku ette seadistata, näiteks Java, Node.js ja NPM õiged versioonid lisada. Samuti töötab I-Tee keskkonnas praktikum kõigil ühtemoodi ehk probleemide korral on tudengeid lihtsam aidata. Lisaks tegi üks tudeng ettepaneku turvavigade parandamised viia kasutajaliidese poole, sest nii on lihtsam pärast muudatuste tegemist rakendus uuesti ehitada. See aga läheb vastuollu praktikumi eesmärgiga tuua välja kõige kriitilisemad

turvavead, mis sageli väljenduvad just serveripoolsete vigadena, sest kasutajaliidese kontrollidest on võimalik mööda pääseda.

Joonisel 9 on välja toodud tulemused praktikumi tulemuste kaupa, kus

- Lahendasid – viitab tudengitele, kes vähemalt osaliselt praktikumi lahendasid
- Esitasid – viitab tudengitele, kes ei teinud ülesannete parandusi, kuid esitasid praktikumi sellest hoolimata
- Ei osalenud – viitab tudengitele, kes praktikumi ei esitanud (ei ole teada, kas nad proovisid praktikumi lahendada)
- Keskmine – kõikide tudengite hinnangute keskmised



	Huvitav	Keeruline	Kasulik
Lahendasid	3.6	4.13	3.4
Esitasid	2.33	4.25	2
Ei osalenud	3.56	3.78	3.56
Keskmine	3.44	4.04	3.3

Joonis 9. Tudengite tagasiside praktikumile kuuepunktilisel Likert skaalal (0 kuni 5).

9 Järeldused ja edasiarendus

Kokkuvõtvalt jääb autor arvamusele, et praktikum oli kasulik ja selle võiks järgmistelgi aastatel plaani võtta, kuid esimesel kursusel on veel liiga vara antud teemat tutvustada. Edaspidi toimub Küberturbe aluste aine kolmandal kursusel. Kui esimese kursuse tudengitele oli praktikum liialt keeruline, siis kolmanda kursuse tudengitele ei pruugi see pakkuda piisavat väljakutset. Kolmanda kursuse tudengite tarbeks tuleks praktikumi veidi keerulisemaks teha, näiteks lisades funktsionaalsust rakendusele, kaotades ära vihjed koodist jne. Ilmselt tuleks sel juhul ka rakendada keerulisemad testid ja neid tudengitele mitte näidata. Töö autor pakub välja näiteks lisada serverisse eraldi rakendus, mis koosneb ainult testidest ja hakkab testima praktikumis loodud veebirakendust (näiteks kasutades REST-assured raamistikku) või kopeerida testid alles pärast hoidlasse lisamist rakenduse juurde. Kolmanda kursuse tudengid saavad tõenäoliselt ka paremini hakkama veebirakenduse käivitamisega isiklikus arvutis, seega saab jätta välja I-Tee keskkonna, mis seekord nii mõnelegi tudengile probleeme valmistas. Samuti peaks uuendama tarkvara, sest see aegub aastaga ning võib-olla võiks praktikumi sooritamiseks olla rohkem aega.

9.1 Soovitused järgmisteks aastateks

- Anda tudengitele rohkem aega lahendamiseks
- Korraldada praktikum teise või kolmanda kursuse tudengitele
- Kui praktikumi lahendavad parema kogemusega tudengid, siis peaks ka selle keerukust tõstma ja muutma rakendust põnevamaks, näiteks lisades funktsionaalsust
- Viia testimine serverisse
- Värskendada praktikumi põhjaks olnud rakenduse tarkvara või arendada hoopis uus rakendus
- Lasta tudengitel teha praktikum isiklikus arvutis

10 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua praktikum, mis võimaldaks tudengitel saada praktilisi kogemusi veebirakenduste turvalisusest.

Eesmärgi saavutamiseks kasutati teoreetilise põhjana OWASP-i projekti Top 10, valides sealt välja 5 turvaauku, mis tuli tudengitel parandada. Iga vea juures esitati OWASP-i rakenduste turvalisuse kontrollimise standardi põhjal selle kategooria ning toodi välja OWASP-i ennetavate kontrollide nimekirja punktid, mille järgimine oleks vea tekke ära hoidnud. Seejärel konfigureeriti virtuaalmasinat, et praktikum oleks tudengitele eelseadistatud I-Tee kauglaborite süsteemis kättesaadav ning lisati turvaaukudega veebirakendus virtuaalmasinasse. Seejärel seadistati GitLabi pideva integratsiooni konveier, et tulemuste testimine oleks veakindlam.

Praktikumis osales 23 tudengit, kes hindasid seda praktikumi suhteliselt keeruliseks. Tudengid, kes suutsid oma töös turvaauke parandada, hindasid praktikumi võrdlemisi huvitavaks ja kasulikuks. Autor on samuti veendunud sedalaadi praktikumi kasulikkuses ning pooldab praktikumi toimumist ka järgmistel aastatel, sest antud teemavaldkond on vaieldamatult päevakajaline. Lähtudes tudengite tagasisidest tuuakse välja mõningad ettepanekud järgnevateks aastateks. Ühe ideena pooldatakse praktikumi viimist stuudiumi hilisemasse faasi. Lisaks sellele soovitatakse kaaluda virtuaalmasina kasutamisest loobumist, sest selle seadistamine on ajakulukas ning esines probleeme nii töökindlusega kui jõudlusega. Võimalus on ka virtuaalmasinale eraldatud ressursse suurendada või optimeerida.

Kasutatud kirjandus

[1] Eesti Keele Instituut, "Eesti keele seletava sõnaraamat," [Online]. Available: http://eki.ee/dict/ekss. [Accessed 15 May 2019].

[2] Cybernetica AS, "Andmekaitse ja infoturbe leksikon," [Online]. Available: https://akit.cyber.ee. [Accessed 15 May 2019].

[3] Tallinna Tehnikaülikool, "Tallinna Tehnikaülikooli õppeinfosüsteem," [Online]. Available: https://ois2.ttu.ee. [Accessed 15 May 2019].

[4] Riigi Infosüsteemi Amet, „Küberturvalisus 2019," Riigi Infosüsteemi Amet, 2019.

[5] OWASP Foundation, "About The Open Web Application Security Project," [Online]. Available: https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project. [Accessed 15 May 2019].

[6] OWASP Foundation, "OWASP Top Ten Project," [Online]. Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf. [Accessed 15 May 2019].

[7] D. Cuthbert, A. v. d. Stock, J. Manico, M. Burnett and J. C. Grossmann, "What is ASVS?," [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project. [Accessed 15 May 2019].

[8] OWASP Foundation, „Application Security Verification Standard 4.0," OWASP Foundation, 2019.

[9] K. Anton, J. Bird and J. Manico, "OWASP Top 10 Proactive Controls," [Online]. Available: https://www.owasp.org/images/b/bc/OWASP_Top_10_Proactive_Controls_V3.pdf. [Accessed 15 May 2019].

[10] T. Tänav, M. Ernits, K. Keijo, C. Fischer, A. Guitar and M. Toom, "I-Tee," [Online]. Available: https://github.com/magavdraakon/i-tee. [Accessed 15 May 2019].

[11] A. Rauschecker, „I-Tee The Lost Manual".

[12] Elsevier, "Password hacks show major sites are vulnerable," *Computer Fraud & Security,* pp. 1, 3, 19 June 2012.

[13] JHipster, "JHipster," [Online]. Available: https://github.com/jhipster. [Accessed 15 May 2019].

[14] Gradle Inc., "What is Gradle?," [Online]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Accessed 15 May 2019].

[15] Codecademy, "What is REST?," [Online]. Available: https://www.codecademy.com/articles/what-is-rest. [Accessed 15 May 2019].

[16] Hibernate, "Hibernate ORM," [Online]. Available: https://github.com/hibernate/hibernate-orm. [Accessed 20 May 2019].

[17] GitLab, "GitLab Runner," [Online]. Available: https://docs.gitlab.com/runner. [Accessed 15 May 2019].

[18] F. B. Zakariae, "Medium," [Online]. Available: https://medium.com/@zakariae/setting-up-symfony-continuous-deployment-using-rancher-274838fcf632. [Accessed 19 May 2019].

Lisa 1 – Tööjuhend inglise keeles

11 Setup

- **You don't need to create the application. All you need to do is fix the existing application.**
- Start the ITI0103-hipster lab at https://elab.cs.ttu.ee/my_labs
- Open the Desktop
- From the main menu select Application -> Programming -> Intellij IDEA Community Edition
- Click on the green arrow at the top-right corner (Figure 3)

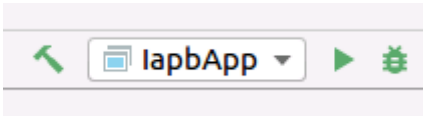


Figure 10. Screenshot of Intellij IDEA.

- Open Terminal at the bottom-left corner (Figure 4)

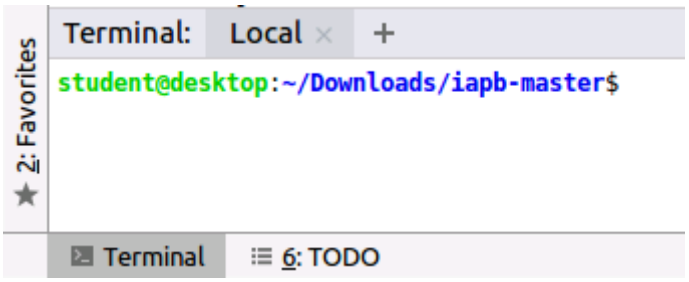


Figure 11. Screenshot of a terminal of Intellij IDEA.

- Type „**npm start**" and click enter
- Your application should start in the browser

12 Story

A company (let's call it YourFitness) struggled with its finances so the management had a meeting to work out the future game plan. They decided that each innovative company should have a website, so they're going to have one. Unfortunately, they didn't have much money, so they decided to spend only a little amount on developing. However, there is a saying „You get what you pay for" and it turned out that the new web application has a couple of pretty serious security problems.

Now they need you as a security specialist to help them out. Don't worry, they gathered all their savings to hire you.

The application itself is a rather simple application with only one domain (training). There are 3 roles (user, coach, admin). However, you need the admin only for the one last bug.

You can log in with following credentials:

1) User
    a. User: user
    b. Password: user
2) Coach
    a. User: coach
    b. Password: coach
3) Admin
    a. User: admin
    b. Password: admin

In fact, there are more users and you can create more as you want.

While logged in you can see three tabs (Figure 5).

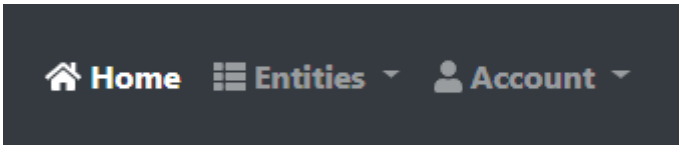


Figure 12. Screenshot of the website menu.

Try to focus on the Entities tab, where you have Training entity. Search for the "TODO" (ignore gradle.properties and .gitattributes files).

As a coach it is possible to add a new training, edit your existing trainings, delete your training, see a detailed view of your training and see your trainings as a list.
As a user it is possible to see the list of trainings, see a detailed view of a training and to register yourself to the trainings (just enable the button in front of the training row).

13 Useful information

13.1 How to open DevTools?

Press Ctrl+Shift+I on your keyboard when the browser is open. You will most likely need only the Network tab. I'm using Firefox screenshots for this lab. These pictures might differ a bit.

13.2 How to make requests in Postman?

Open any of the requests under the Network tab while logged in (Figure 6). You should see Authorization under the Request headers. Copy the value (Bearer ...).

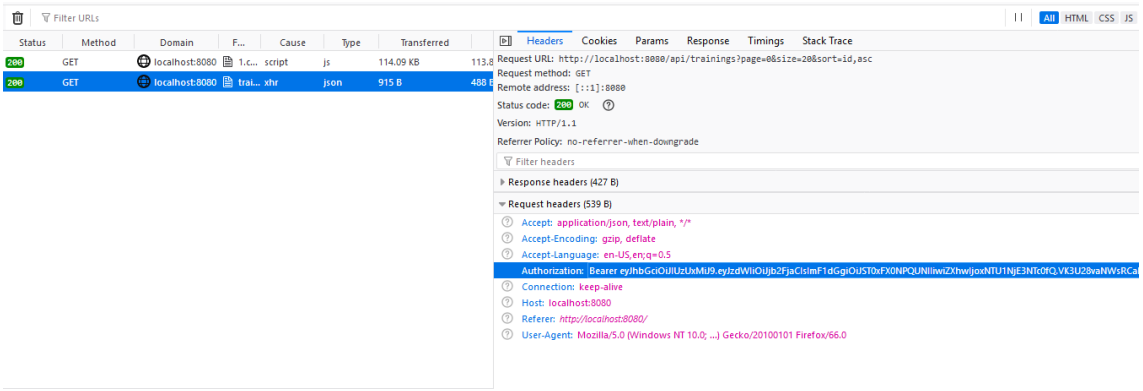


Figure 13. Screenshot of a Chrome Devtools.

Open Postman (you'll find it in the Downloads folder). You don't have to register. Just click „Skip". Replace Authorization value (under the Headers tab) with the one you copied (Figure 7).

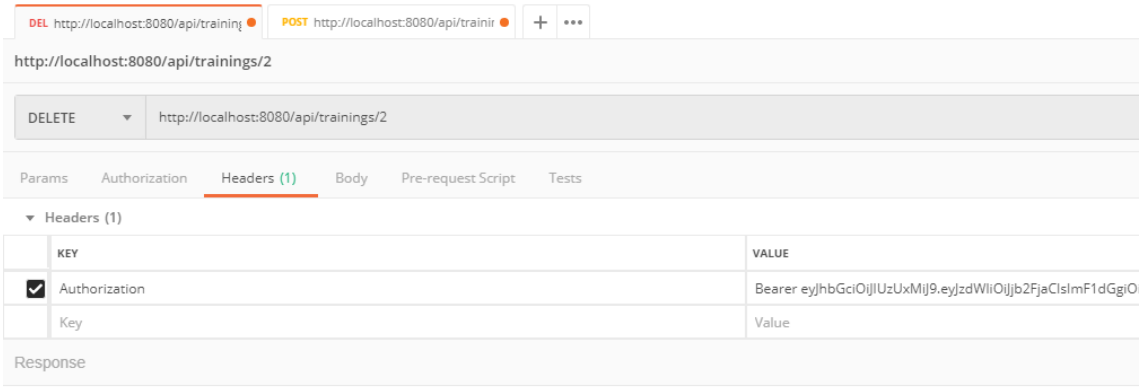


Figure 14. Screenshot of a DELETE request in Postman.

14 Bugs

Now let's get started. YourFitness listed a couple of bugs you'll need to fix:

14.1 Bug 1

It turned out that the users are curious about other people joining the session. So, they found that it is possible to see users registered to training from the DevTools tab.

14.1.1 How to reproduce

- Log in as a user.
- Click on Entities -> Training from the application menu.
- Open DevTools and click on the Network tab.
- Look at the detailed view of any training.
- Click on the request in a Network tab with a following Request URL: http://localhost:8080/api/trainings/1.
- Click on Response. You should see other users and sensitive data such as emails, although they are hidden from the UI (Figure 8).

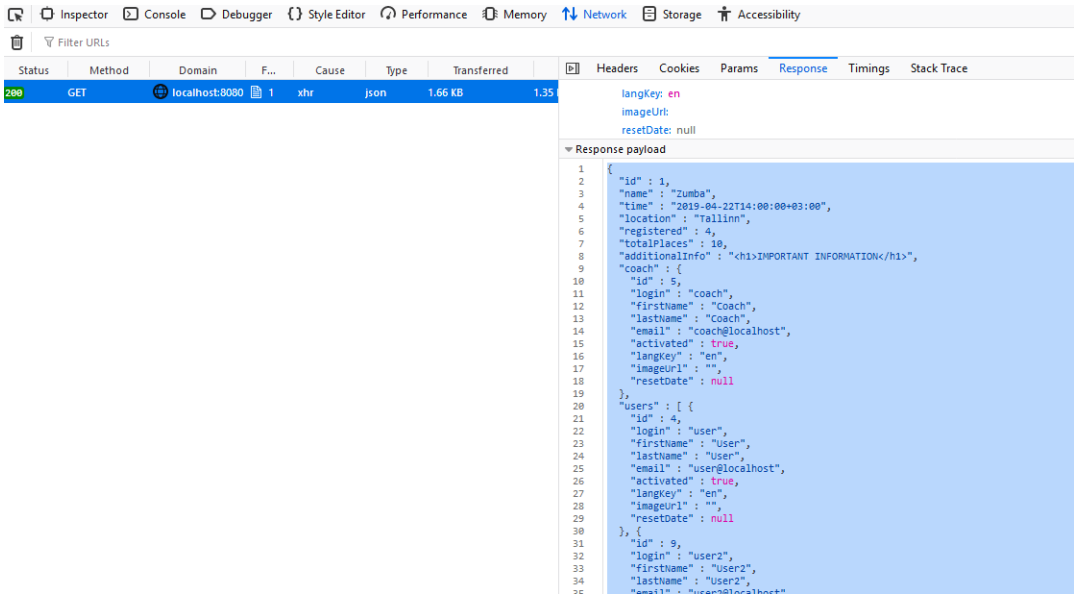


Figure 15. Screenshot of Chrome Devtools.

14.1.2 What do you have to do

- Hide other users data from Network tab. It means that you should not send this data to the front-end at all. Check classes: TrainingServiceImpl.java, TrainingRepository.java.

14.2 Bug 2

It was also possible to change other fields from the Training entity while registering yourself to the training. So, the users found a way to book more places to the training, so they could register themselves to a training with their friends even when there were not enough places left.

14.2.1 How to reproduce

- Log in as a user.
- Click on Entities -> Training from the application menu.
- Open DevTools and click on the Network tab.
- Look at the detailed view of any training.
- Click on the request with a Request URL like this http://localhost:8080/api/trainings/1 .
- Click on Response. Copy everything you see. Look at Figure 5.
- Open Postman application. You should have a POST request opened in the tabs.
- Paste everything you copied to the Body tab. Don’t forget to change the authorization value (Figure 3).
- Try to change anything (for example number of total places) and then click “Send” (Figure 9)

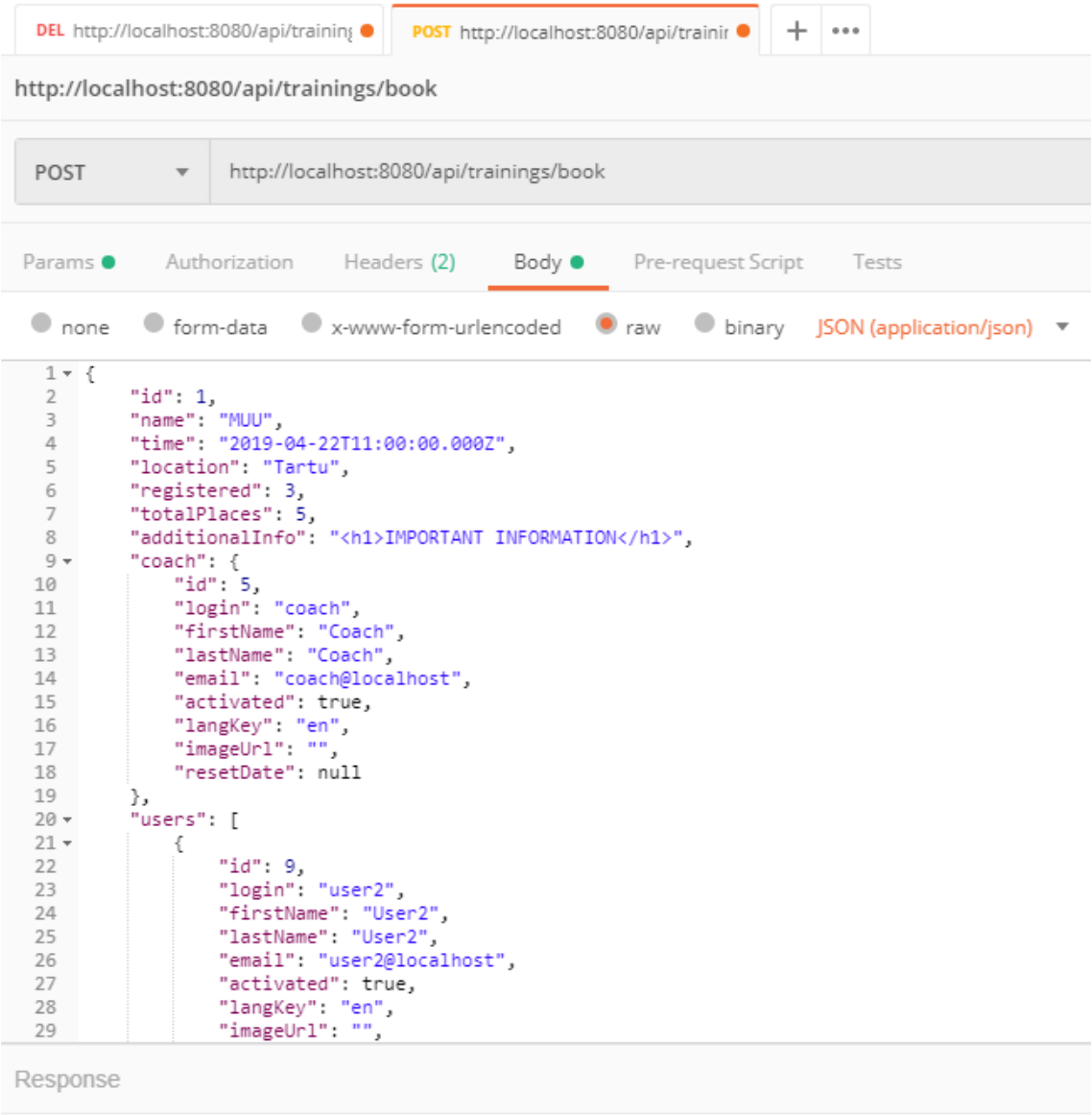


Figure 16. Screenshot of a POST request in Postman.

- Refresh browser and you should see that the information about the training changed, although this request should allow you to only register yourself and not change the data.

14.2.2 What do you have to do

- You must disable the option to change the data. Check class: TrainingServiceImpl.java

14.3 Bug 3

Coaches also found a way how to make their trainings more appealing. It was possible to add scripts or tags into the textboxes.

14.3.1 How to reproduce

- Log in as a coach.
- Click on Entities -> Training from the application menu.
- Click “View” on “Zumba”.
- You should see that additional info is written as a header, although only regular text should be allowed (Figure 10).

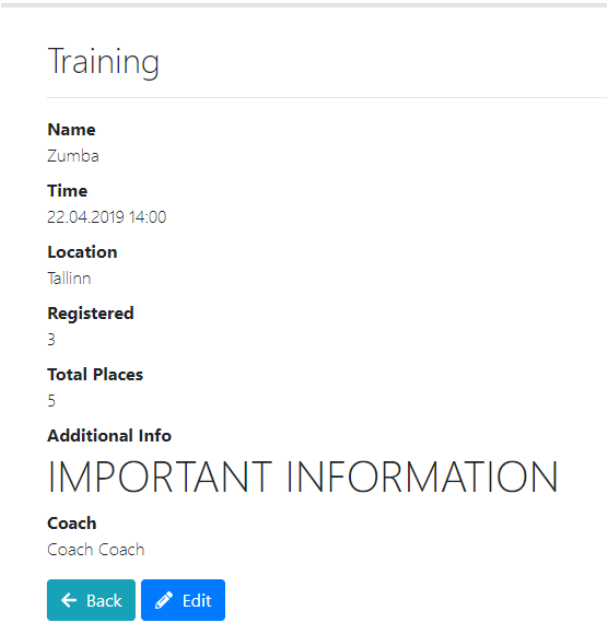


Figure 17. Screenshot of the web application.

14.3.2 What do you have to do

- You must disable the option to add scripts or tags to the text box. Check file: training-detail.component.html

14.4 Bug 4

These coaches can be rather devious. They were able to see a detailed view of other coaches’ trainings, although they didn’t see these trainings in a list. What else? They found out that is possible to delete other coaches’ trainings.

14.4.1 How to reproduce

- Log in as a coach and add a new training.
- Click on detailed view of the training and remember the id (probably 1001).
- Log in as a coach2 (user: coach2, pw: coach2).
- Open DevTools and click on the Network tab.
- Now click on Entities -> Training from the menu.
- Click on detailed view of any training.
- Replace the id in the URL with the one you created previously (Figure 11).

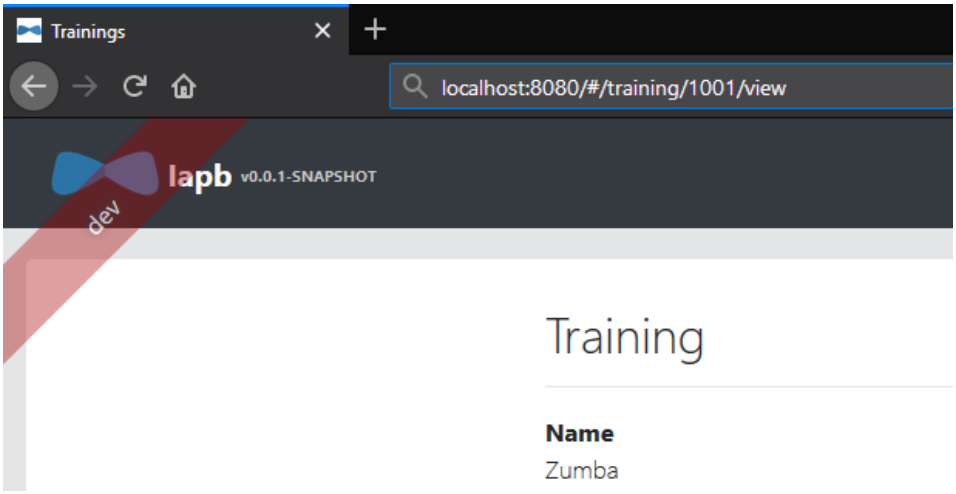


Figure 18. Screenshot of the web application.

- You are now able to see other coach's training.
- Open Postman. Don't forget to change the authorization value (Figure 3).
- You should have DELETE request opened in the tabs.
- Replace the id in the URL with the one you created previously and click "Send" (Figure 12)

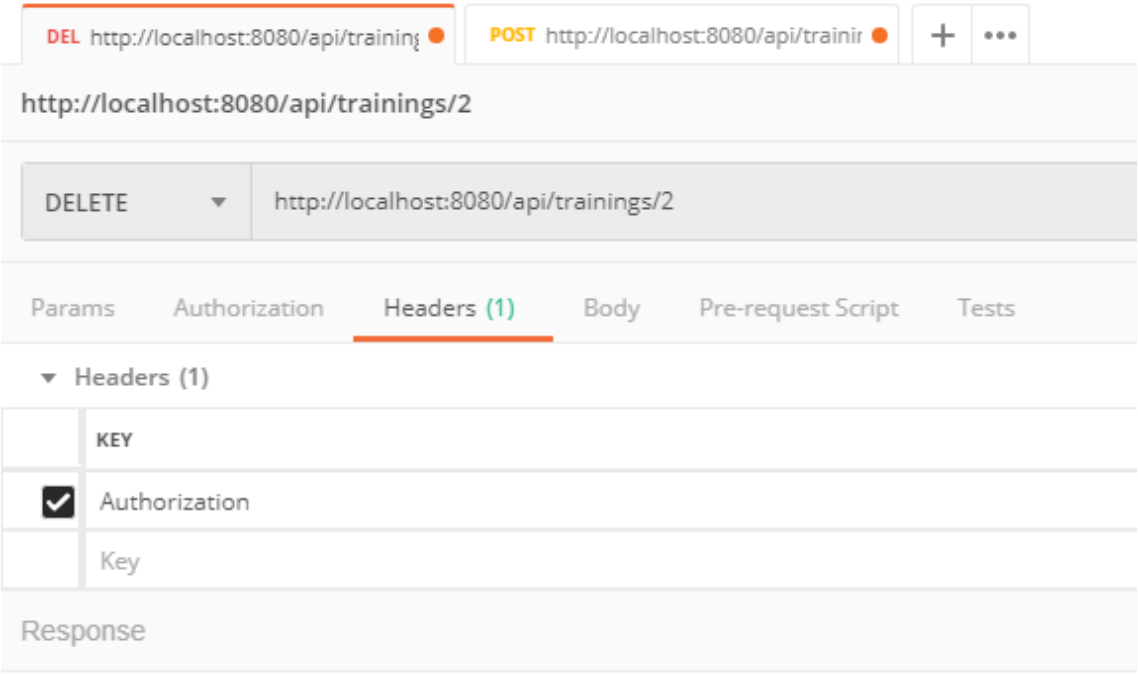


Figure 19. Screenshot of a DELETE request in Postman.

- You should now see that the training is deleted, although current user wasn't the owner (coach) of the training and it should not be possible for the current user.

14.4.2 What do you have to do

- You should fix access rights for the roles. Coach should be able to delete only his/her trainings not all the other coaches' trainings. Check class: TrainingResource.java

14.5 Bug 5

There is one hidden bug as well.

Hint: You have access to the database. Go look at the data in jhi_user table.

Log in as an admin. Click on the Administration -> Database or just go to the URL http://localhost:8080/h2-console and click „Connect" (no need for the password).

14.5.1 What do you have to do

- Check class: SecurityConfiguration.java

15 When finished

15.1 Try to run the tests.

gradlew test

15.2 Go to the Terminal and type following commands. You'll need to replace the text written in red.

git config --global user.email "your_uni_id@ttu.ee"

git config --global user.name "Your Name"

git init

git remote add origin https://gitlab.cs.ttu.ee/your_uni_id/iti0103-2019.git

git add .

git commit -m "Initial commit"

git push -u origin master

15.3 Go to gitlab.cs.ttu.ee

You should see iti0103-2019 project on your dashboard. Open it and go to Settings -> CI/CD. Expand "Runners" and click on the "Enable shared Runners" (Figure 13).

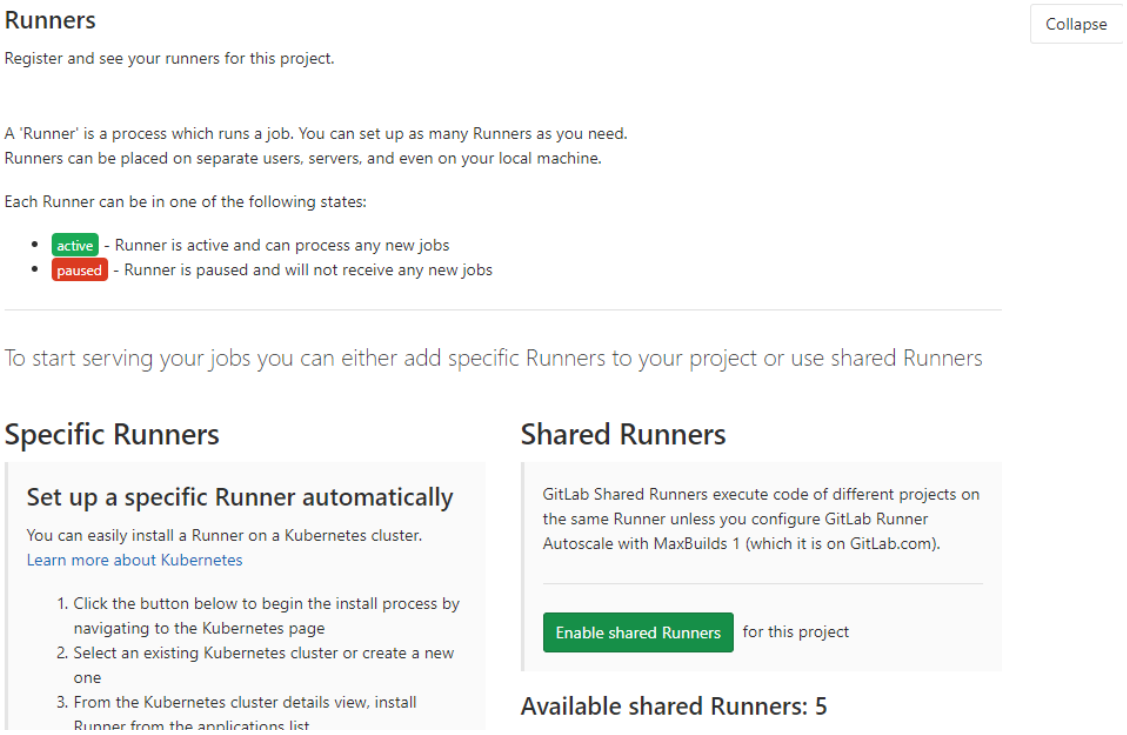


Figure 20. Screenshot of GitLab settings.

Now go back to the project dashboard (Figure 14). You should see an orange circle (it might be blue with the message “Running” as well).

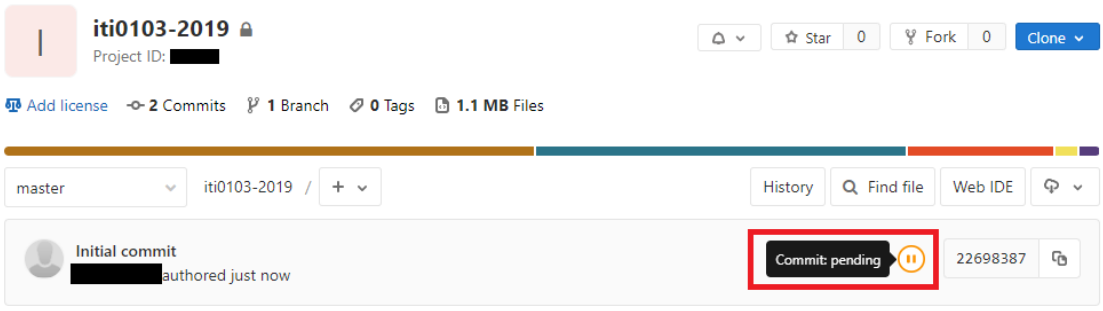


Figure 21. Screenshot of the project dashboard in GitLab.

Click on the circle and then click on the red “X” to stop the pipeline and then click “Retry” (Figure 15). If everything went smoothly you should get the results to your email after a couple of minutes. Don’t worry when the message says that “Pipeline has failed”, these tests are quite general and don’t say much. You may have done everything correctly, but still some of the tests may fail.

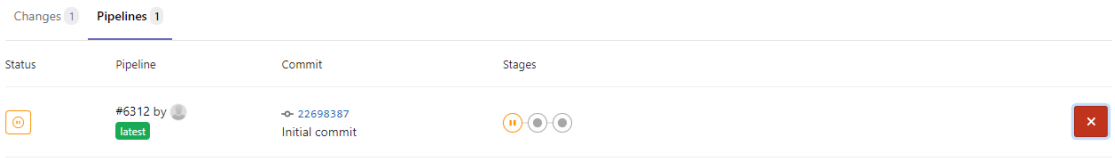


Figure 22. Screenshot of GitLab CI pipeline.