TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Johannes Kunts IAPB142850

Charon pistikprogrammi turvalisuse analüüs ja täiustamine

Bakalaureusetöö

Juhendaja: Ago Luberg

Infotehnoloogia teaduskond

Tallinn 2021

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: Johannes Kunts

25.05.2021

Annotatsioon

Käesoleva bakalaureusetöö raames analüüsitakse ning parendatakse TalTech ülikoolis kasutusel oleva Moodle'i pistikprogrammi Charoni turvalisust ning kirjeldatakse saadud tulemusi ja pakutakse välja võimalusi nende parandamiseks ja pistikprogrammi turvalisuse tõstmiseks.

Töö on jagatud kaheks osaks: analüüs ning turvalisuse tõstmine. Analüüsi osas tegeletakse turvalisuse uurimise ning leitud turvavigade kirjeldamisega. Turvalisuse tõstmise osas kirjutatakse töö raames tehtud muudatustest ning kirjeldatakse pistikprogrammi turvalisuse tõstmise võimalusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 18 leheküljel, 4 peatükki, 8 joonist, 0 tabelit.

Abstract

Analysis and Improvements of Charon Plugin Security

This bachelor's thesis analyses and improves the security of a plugin called Charon that is used in Moodle of TalTech university. It also describes gotten results and proposes oportunities to improve the results to raise the security of the plugin.

The thesis is divided into two parts: analysis and improving the security. In the part of analysis the security is examined and found security issues are being described. The chapter of improving the security it's written about changes made to the plugin in the course of this thesis and more possible changes are being described to improve the security even further.

The thesis is in Estonian and contains 18 pages of text, 4 chapters, 8 figures, 0 tables.

Lühendite ja mõistete sõnastik

Argon2	Krüptograafiline räsifunktsioon
API	*Application Programming Interface*, rakendusprogrammiliides
cookie	Küpsisefail, sessiooniidentifikaator
CSS	*Cascading Style Sheets*, küljendamisel kasutatav märgistuskeel
DB	Laravelis kasutusel olev PHP klass
Eloquent ORM	*Eloquent Object-Relational Mapper*,
Gitlab	Veebipõhine arendus+käitus platvorm
Gitlab CI	*Gitlab Continuous Integration*, Gitlab'i pidev integratsioon
HMAC	*Hash-based Message Authentication Code*, räsipõhine sõnumiautentimiskood
HTML	*HyperText Markup Language*, hüperteksti märgistuskeel
JavaScript	Programmeerimiskeel peamiselt veebilehtede skriptimiseks
Laravel	PHP veebirakenduse raamistik
middleware	Vahevara
Moodle	*Modular Object-Oriented Dynamic Learning Environment*, modulaarne objektorienteeritud õppekeskkond
OWASP	*The Open Web Application Security Project*, avatud veebirakenduse turvalisuse projekt
PDO	*PHP Data Objects*, PHP liides andmebaaside ligipääsuks
PHP	*PHP: Hypertext Preprocessor*, skriptimiskeel
phpcs	*PHP CodeSniffer*, skript, mis aitab tuvastada eksimisi levinud koodistandardite vastu
phpmd	*PHP Mess Detector*, lähtekoodi analüüsija
Query Builder	Laraveli andmebaasi päringu ehitaja
SHA-256	*Secure Hash Algorithm*, 256-bitine turvaline räsialgoritm
SQL	*Structured Query Language*, struktuurpäringukeel
Vue	JavaScripti raamistik
XSS	*Cross-site scripting*, murdskriptimine

Sisukord

Jooniste loetelu

1 Sissejuhatus

1.1 Taust

TalTech ülikoolis on kasutusel Moodle'i platvorm, millele on 2017. aastal bakalaureusetöö raames kirjutatud pistikprogramm Charon ehk programmeerimisülesannete automaattestimissüsteemi liidestus. Charonile on ka 2019. aastal bakalaureusetöö tarbeks loodud lisafunktsionaalsust ning lisaks sellele on pistikprogramm järjepidevalt arenduses mitmete tudengite poolt.

See, et antud programmi on arendatud ja arendatakse mitmes eri faasis erinevate tudengite poolt, kes pole ilmtingimata üksteisega seotud, tähendab, et kasutusel on mitmeid erinevaid koodipraktikaid või pole neid kogemuse puudumise tõttu üldse. Ühtlasi puudub arendusprotsessis seeniorarendaja, kes oleks kindlaid häid koodipraktikaid või abistavaid arendustööriistu kasutusele võtnud ning oleks eeskujuks nende järgimisel ja kasutamisel.

Eelnevalt mainitu põhjal võib oletada, et programmis võib esineda turvalisuse kitsaskohti.

1.2 Eesmärk

Antud töö eesmärk on uurida, millises seisus on Charon pistikprogrammi turvalisus praegu, millised on puudujäägid ning sõltuvalt uurimise tulemustest üritada neid vajalike metoodikatega parendada.

1.3 Ülevaade tööst

Uurimisel lähtutakse esmalt programmis kasutusel olevatest pakettidest ning veebis kirjas olevatest vastavate pakettide turvaohtudest uurides nende esinemist Charonis.

Lisaks sellele uuritakse OWASP 10 enim levinud veebirakenduste turvariskide esinemist antud programmis valides kümne seast välja antud rakenduse kontekstis kõige tõenäolisemad.

2 Analüüs

2.1 OWASP

Avatud veebirakenduse turvalisuse projekt (OWASP - The Open Web Application Security Project) on asutus, mis tegeleb tarkvara turvalisuse parendamisega. Nad korraldavad turvalisusega seotud konverentse, väljastavad antud teemal artikleid ja avatud lähtekoodiga projekte.[1]

2.1.1 OWASP'i Top 10

OWASP'i Top 10 on standardeid teadvustav dokument arendajatele. See sisaldab kümmet kõige kriitilisemat turvariski veebirakenduste juures. Antud nimistu on koostatud aastal 2017. ning sisaldab endas kümmet turvariski[2], millest käesoleva bakalaureusetöö raames sai lähemaks uurimiseks välja valitud kõige tõenäolisemalt esinevad riskid, milleks olid järgnevad kolm:

A1 Süstimisrünne – Süstimisnõrkused (näiteks SQL) esinevad, kui ebausaldusväärsed andmed saadetakse käsitsi või päringuga rakendusse. Ründaja võib niimoodi jooksutada soovimatuid käske või saada ligipääsu andmetele, millele tal tegelikult õigusi pole.[2] [3]

A7 Murdskriptimine (Cross-Site Scripting – XSS) – XSS vead esinevad, kui rakendus kuvab veebilehel ebaturvalisi andmeid ilma korrektse valideerimiseta. XSS vigade kaudu saavad pahatahtlikud kasutajad käivitada ohvri veebilehitsejas skripte, misläbi saavad omastada nende sessiooni, muuta kasutatavat veebilehte või suunata ohvreid teistele pahaloomulistele veebilehtedele.[2][3]

A9 Teadaolevate haavatavustega komponentide kasutamine – Komponendid nagu näiteks teegid, raamistikud ja muud tarkvaramoodulid töötavad samade privileegidega nagu rakendus ise. Haavatavate komponentide kasutamine võib muuta rakenduse

turvalisuse kasutuks ja võimaldada erinevaid rünnakuid, mis võivad viia andmete kaoni või serveri ülevõtmiseni.[2][3]

2.2 SQL-süstimisrünne

Charoni turvalisust uurides keskenduti peamiselt SQL süstimisründele. Kuna oli olemas ligipääs rakenduse lähtekoodile, siis ei olnud vajalik hakata kõiki päringuid ükshaaval süstimisründe vastu testima, vaid oli võimalik teha lähtekoodist järeldusi, kus on potentsiaalne oht olemas.

Laravelis kasutusel olev Query Builder (PHP klass, millega ehitatakse SQL- või muid andmebaasipäringuid) ja PDO (*PHP Data Object* ehk PHP andmeobjekt), mis on liides andmebaaside ligipääsuks PHP-s[4], on üldjuhul süstimisründe vastu turvatud, sest kasutatakse PDO parameetrite sidumist.[5] See tähendab, et seotud parameetreid ei ole vaja eraldi vastuvõetavaks teha.

Küll aga tuleb SQL-süstimisrünne mängu siis, kui hakatakse kasutama Laravelis kasutusel oleva DB fassaadi võimalus sisestada „toorest” SQL-i. Selleks on mainitud fassaadis erinevaid meetodeid:

- selectRaw
- whereRaw
- orWhereRaw
- havingRaw
- orHavingRaw
- orderByRaw
- raw

Kõik eelnevad meetodid peale viimase võtavad vastu teise argumendina vastu massiivi sidumist ootavatest parameetritest. See eeldab, et esimeses argumendis olevas sõnes on küsimärgiga tähistatud kohad, kuhu järjest neid parameetreid asendama asutakse.[6]

Et vältida liigset ajakulu ning testida läbi kõik rakenduses kasutusel olevad päringud, et leida võimalikud SQL-süstimisründe ohud, sai üles otsitud vaid kõik eelmainitud meetodite kasutuskohad Charoni lähtekoodis ning kontrollitud, kas antud kohtades on heade koodimistavade vastu eksitud või mitte. Tulemuseks leiti kaks meetodit, mis kujutasid endast ohtu:

- getDefenseRegistrationsByCourseFiltered (DefenseRegistrationRepository.php)
- findAllSubmissionsForReport (SubmissionsRepository.php)

Uurides neist esimest leiti, et antud meetod on kasutuses 'courses/{course}/defenseRegistrations/{after}/{before}/{teacher_id}/{progress}' päringus, läbi SubmissionsController.php kontrolleri, kus oli näha, et päringu sisendandmeid ei ole mitte kuidagi valideeritud, mis teeb päringu suureks süstimisründe ohuks.

Edasi sai proovitud süstimisrünnet UNION päringu kaudu. UNION’it kasutatakse, et lisada soovitud SQL-süstimine legitiimsele päringule ning kombineerida meie soovitud informatsioon päris informatsiooniga. Sellise päringu jaoks on tarvis teada, mitut veergu andmebaasist küsitakse[7] ning selleks tuleb lisada päringule järjest NULL väärtuseid kuniks veergude arv on korrektne.

Et taaskord vältida liigset ajakulu sai koheselt valitud päringu argument $teacher_id, millesse meie süstimisrünne teha. Esmalt sai küsitud andmebaasi tabelite nimesid sisestades argumendi väärtuseks „’ UNION ALL SELECT concat(TABLE_NAME), null, null, null, null, null, null, null, null, null, null, null, null FROM information_schema.TABLES -- „ ning süstimisrünne oli edukas ja väljastati kõikide tabelite nimed.

Järgmise sammuna sai juba valida soovitud tabel, milleks osutus näiteks mdl_user, mis sisaldab endas kõikide kasutajate andmeid ning seades argumendi väärtuseks ’ UNION ALL SELECT concat(column_name), null, null, null, null, null, null, null, null, null, null, null, null FROM information_schema.COLUMNS WHERE TABLE_NAME='mdl_user' -- „ olid tulemusteks kõik soovitud tabeli veergude nimed, et saaks järgmise sammuna küsida valitud veergude väärtusi.

Järgmise päringu parameetri väärtuseks oli seega „’ UNION ALL SELECT username, password, firstname, lastname, email, null, null, null, null, null, null, null, null FROM mdl_user -- „ ning saadud tulemus on näha järgmisel joonisel.

Saadud vastuses on iga elemendi atribuutide nimed nimetatud päringu algselt soovitud tulemuste põhjal, mitte meie süstitud päringu veergude nimede järgi.

```
[
    {
        "id": "guest",
        "choosen_time":
"$2y$10$J.xfWtHnla2h21OA2926TOXQBujvkq559.qVrm
tU.EwaBW4pGlP16",
        "student_id": "Guest user",
        "student_name": " ",
        "charon_id": "user@example.com",
        "defense_duration": null,
        "my_teacher": null,
        "submission_id": null,
        "progress": null,
        "charon_defense_lab_id": null,
        "lab_name": null,
        "teacher": {
            "id": null,
            "firstname": null,
            "lastname": null,
            "fullname": " "
        }
    },
    {
        "id": "dev",
        "choosen_time":
"$2y$10$E/FSIWQAJ2urlJj1BPohsOcWjuSjRIKQTGSBUr
jYu/EiHaViwxuJi",
        "student_id": "Admin",
        "student_name": "User",
        "charon_id": "user@example.com",
        "defense_duration": null,
        "my_teacher": null,
        "submission_id": null,
        "progress": null,
        "charon_defense_lab_id": null,
        "lab_name": null,
        "teacher": {
            "id": null,
            "firstname": null,
            "lastname": null,
            "fullname": " "
        }
    }
]
```


Joonis 1. getDefenseRegistrationsByCourseFiltered päringu vastus

Kokkuvõtteks saab öelda, et antud päring on ebaturvaline, sest võimaldab SQL-süstimisrünnet. Uurides selle päringu kirjeldust on näha, et tegelikult on seal kasutusel ka vahevara (*middleware*), mis nõuab selle kasutamiseks ligipääsuõigusi, seega päris iga kasutaja SQL-süstimisrünnet toime panna ei saa. Antud päringu puhul on nõutud, et autoriseeritud kasutaja omaks päritava kursuse haldamise õigusi. Küll aga on ülikoolis levinud ka tudengite kasutamine abiõppejõududena, kes võivad antud turvaviga kasutades omistada erinevat infot, mida saab potentsiaalselt kurjalt ära kasutada ning miski ei välista ka päris õppejõu halbu motiive info omistamisel.

2.3 Murdskriptimine

Charon kasutab lisaks Laravelile ka JavaScripti teeki Vue. Vue's on rakendatud mitmeid kaitsemehhanisme XSS rünnakute kaitseks. Suurim oht seisneb *v-html* atribuudi kasutamises.[8] Kasutades antud atribuuti koos ebakindla sisendväärtusega on veebileht avatud murdskriptimisele.

Charonis sai otsitud üles kõik v-html kasutuskohad ning üle vaadatud, kust pärineb nende sisend. Ohtlikuks kujunesid kaks kasutuskohta:

- SubmissionModal.vue
- OutputSection.vue

Mõlemas failis on määratud HTML elemendi sisuks läbi v-html atribuudi „submission.mail". Antud sisendit uurides jõuti tagasi andmebaasi tabelini charon_submission, kust antud andmed pärinevad. Andmed sinna andmebaasi tabelisse jõuavad omakorda päringust „tester_callback" ning sisendeid ei ole ka selle päringu puhul valideeritud. Välja puhul on tegemist e-maili sisuga HTML elemendina ning on genereeritud mõne teise rakenduse poolt. Juhul kui mujal genereeritud e-maili sisse lisatakse mingid väärtused, mida kasutaja saab ise sisestada ning mida ka seal rakenduses ei valideerita, siis võib ohtlik skript jõuda läbi Charoni andmebaasi rakenduse kasutajaliidesesse ning sel juhul on tegemist XSS turvaveaga. Kui teises rakenduses aga väärtuseid valideeritakse ning genereeritav HTML on turvaline, ei tohiks antud juhul probleemi esineda. V-html'i kasutamist alati vältida ei saa, kuid sel

juhul tuleb veenduda, et sinna jõudvad andmed on turvalised ning pärinevad usaldusväärsest allikast.

2.4 Teadaolevate haavatavustega komponentide kasutamine

Käesoleva bakalaureusetöö raames sai Charon pistikprogrammis kasutusel olevaid raamistike ning teeke kontrollitud turvavigade nimekirja vastu, mis on PHP Turvalisuse Nõuannete Andmebaasis (PHP Security Advisories Database). Antud andmebaas sisaldab viiteid teadaolevatele haavatavustele erinevates PHP projektides ja teekides.[9]

Oma programmi selle andmebaasi vastu kontrollimiseks lokaalselt oma arvutis on vajalik alla laadida käsurea tööriist[1]. Antud tööriist laeb värskeima versiooni eelmainitud andmebaasist alla ning kontrollib seda PHP projekti composer.lock failiga võrreldes.[10]

1 https://github.com/fabpot/local-php-security-checker

2.4.1 PHP Turvalisuse Nõuannete Andmebaasi tulemused

```
Symfony Security Check Report
=============================

 packages have known vulnerabilities.

laravel/framework (v5.5.44)
---------------------------

 * [CVE-NONE-0001][]: RCE vulnerability in
"cookie" session driver
 * [CVE-NONE-0002][]: Guard bypass in Eloquent
models

[CVE-NONE-0001]:
https://blog.laravel.com/laravel-cookie-
security-releases
[CVE-NONE-0002]:
https://blog.laravel.com/security-release-
laravel-61834-7232

Note that this checker can only detect
vulnerabilities that are referenced in the
security advisories database.
Execute this command regularly to check the
newly discovered vulnerabilities.
```

Joonis 2. PHP Turvalisuse Nõuannete Andmebaasi kontrolli tulemused.

Tulemuste kohaselt on Charon pistikprogrammis kasutusel vaid üks teadaolevate turvariskidega teek. Selleks on kasutusel oleva raamistiku Laraveli versioon 5.5.44.

Antud raamistiku versioonis on kaks turvaviga – Eemalt koodi süstimise haavatavus Laraveli „cookie” sessiooni draiveris (RCE vulnerability in "cookie" session driver) ning Validatsioonist möödapääsemine Eloquent’i mudelites (Guard bypass in Eloquent models).

2.4.2 Eemalt koodi süstimise haavatavus Laraveli „cookie" sessiooni draiveris

Laravelis saab hoida kasutaja sessiooni mitmel erineval viisil. Valida on näiteks faili, andmebaasi, küpsise (*cookie*) ja mitme muu variandi vahel.[11] Selle turvavea ära kasutamine eeldab, et rakendus kasutaks just *cookie* sessiooni draiverit.[12]

Turvaviga annab võimaluse pahatahtlikul kasutajal endale kokku panna soovitud sessiooni, sest igale kasutaja sisendile saadeti alati vastu sama sisend krüpteerituna. Selle vältimiseks hakkas Laravel uuemates versioonides lisama sisendile küpsise nime HMAC (*hash-based message authentication code* ehk räsipõhine sõnumi autentimiskood) paiskeväärtust enne sisendi krüpteerimist. [12]

Charon seda draiverit hetkel ei kasuta ning seeläbi selle turvavea ärakasutamise ohus ei ole, kuid kindlasti ei tohiks seda draiverit enam enne Laraveli versiooni uuendamist kasutusele võtta.

2.4.3 Validatsioonist möödapääsemine Eloquent'i mudelites

Laravelis on kasutusel Eloquent ORM (Object Relation Mapping), mille üheks osaks on mudelid. Mudel kirjeldab andmebaasi ühe konkreetset tabelit. Mudeli kaudu toimub selle andmebaasi tabeliga suhtlus, sinna sisestamine, sealt kustutamine ning andmete küsimine ning uuendamine. Andmete massiliseks uuendamiseks kasutatakse mudeli meetodit *fill*.[13]

Antud turvaviga on tingitud ühest *fill* meetodis kasutusel olevast *removeTableFromKey* abimeetodist, mis oli algselt lisatud mugavuse eesmärkidel. [14]

```
/**
* Remove the table name from a given key.
*
* @param string $key
* @return string
*/
protected function removeTableFromKey($key)
{
      return Str::contains($key, '.') ?
      last(explode('.', $key)) : $key;
}
```

Joonis 3. Turvaohtu põhjustav Laraveli meetod.

Selle abimeetodi eesmärk on aktsepteerida andmete uuendamisel uuendatavate andmete atribuutides ka tabeli nime tabeli veeru nime ees ning see siis vahetult enne uuendamist sealt eemaldada, et uuendamine oleks korrektne.[14] Näiteks joonisel 4 olevas näites uuendatakse mudeli „name” atribuuti ning antud abimeetod eemaldab uuendamise käigus atribuudi eest mudeli tabeli nime („users”). Antud meetod annab aga võimaluse andmetel mööda pääseda rakenduse validatsioonist, kus kontrollitakse näiteks vaid päringus olevat „name” atribuudi väärtust ning päringus sisalduvat „users.name” väärtust ei valideerita.

```
$model->fill(['users.name' => 'Taylor']);
```

Joonis 4. Turvaohtu põhjustava meetodi kasutamise näide[14].

Sama meetod on omakorda kasutusel ka mitmetes teistes Eloquent’i Model klassi meetodites. Ohustatud meetodid, mis kõik läbi teiste meetodite kasutavad *fill* meetodit on järgnevad:

- fill
- update
- create
- make

- firstOrNew
- findOrNew
- firstOrCreate
- forceCreate
- hydrate
- __construct ehk uue klassi instantsi loomine

Charonis sai üles otsitud kõik antud meetodite kasutusjuhud ning läbi vaadatud, kust kohast ning milline sisend sinna tuleb – kas tegemist on mõne meetodiga, kuhu jõuab kasutaja sisend või mitte. Turvaviga osutuks ärakasutatavaks sel juhul, kui mõnda eelmainitud meetodisse paisata kõik päringu andmed neid filtreerimata *$request->toArray()* või *$request->all()* või filtreerides välja vaid mittetahetud väärtuseid *$request->except(['võti', 'võti_2'])*

Tulemusena selgus, et hetkel Charonis ühtki sellist kasutuskohta ei ole, kus valideerimata andmed jõuaksid sel moel andmebaasi.

3 Turvalisuse tõstmine

Pärast Charoni turvalisuse analüüsimist said tehtud ka mõned sammud turvalisuse tõstmiseks, et edaspidi rakendust arendades ohtudele kiiremini jälile jõuda või neid mingil määral üldse vältida.

Selleks on plaanis kasutusele võtta phpcs, phpmd ning lisada kontroll PHP Turvalisuse Nõuannete Andmebaasis eksisteerivate turvariskide vastu projekti Gitlab'i pidevasse integratsiooni (Gitlab Continuous Integration – Gitlab CI). Kõik eelnev on ära tehtud antud bakalaureusetöö raames ning sellele lisaks on soovitatud kiiremas korras uuendada Laraveli versioon 5.5 pealt 8.x peale.

3.1 phpcs

Phpcs ehk PHP Codesniffer on skript, mis tokeniseerib PHP, JavaScripti ja CSS'i failid, et tuvastada eksimisi levinud koodistandardite vastu. Selle tööriista abiga on lihtsam säilitada programmi lähtekoodis loetavus ja ühtsus.[15]

Töö eesmärgi saavutamiseks lisati phpcs Charoni vajalike pakettide nimistusse ning projekti lisati phpcs reeglistiku baas, mida Charoni arendajad peaksid kohandama enda tingimustele vastavaks, et hoida arendajate seas samade standardite järgimist.

3.2 phpmd

Phpmd (PHP Mess Detector) on lähtekoodi analüüsija, mis on PHP programmeerimiskeele ekvivalent sarnasele Java tööriistale PMD[16], mis otsib levinud programmeerimise vigu (näiteks kasutamata muutujad, ebavajalike objektide loomine jms).[17]

Antud töö raames sai phpmd lisatud Charoni ülesseadmiseks vajalike pakettide nimistusse, et igal arendajal tööriist olemas oleks. Lisaks sai loodud phpmd töötamiseks

tarvilik reeglistik phpmd_ruleset.xml, mis defineerib reeglid, mille rikkumisi phpmd antud projektist otsima peaks.

Hetkel on reeglistikku lisatud vaid üks reegel, mis sai kirjutatud käesoleva bakalaureusetöö raames ning mida on kirjeldatud järgmises alapeatükis.

3.3 phpmd reegel SQL-süstimisründe vältimiseks

Seoses tulemustega, mis ilmnesid analüüsi peatükis SQL-süstimisrünnet uurides sai vastu võetud otsus kirjutada phpmd reegel, mis aitaks edaspidi selliseid ohte vältida.

Esmalt tuli tutvuda phpmd süsteemiga – kuidas see töötab ning kuidas seda kasutada. Reegel võib olla teadlik klassidest, funktsioonidest, meetoditest või liidestest ning kirjutatava reegli jaoks oli vaja, et reegli klass implementeeriks MethodAware nimelist liidest ja saaks sisendiks alati argumendina meetodi sõlme (*node*).[18]

Sealt edasi otsib kirjutatud reegel rekursiivselt läbi meetodi laps-elemendid, millest ta koosneb (näiteks muutujad, argumendid, skoobid jne.). Leides kuskilt sõlme, mis on mõni DB fassaadi meetoditest, mis loob toorest SQL päringut, tehakse kontroll, kas meetodit on kasutatud korrektselt koos parameetrite sidumisega või mitte. Halva kasutuskoha sõlm lisatakse ohtlike sõlmede nimekirja ning iga turvaohuga sõlme kohta lisatakse sõlme asukoha kirjeldus phpmd tööriista väljundisse.

Reeglit kirjutades ilmnes ka aga üks puudujääk phpmd tööriista juures. Nimelt ei suuda ta sõne sisse süstitud muutujat üles leida. Tööriist saab küll aru, et tegemist ei ole literaaliga, kuid satub hätta ta komponentideks tegemisega ning näeb justkui oleks tegemist tühja sõnega. Näiteks kuvab ta „*select * from $muutuja*” hoopis „”-na. Sellest tingituna ei suuda reegel sellisel juhul tagasi vaadata, kust seal kasutatav muutuja pärineb ning kas ta kujutab endast ohtu.

```
/bitnami/moodle/mod/charon/plugin/app/
Repositories/
DefenseRegistrationRepository.php:186
A method
getDefenseRegistrationsByCourseFiltered uses a
variant of DB raw() without bindings:
$filteringWhere on line 217 traces back to
$after on line 191 that is a method argument
$teacher_filter on line 218 traces back to
$teacher_id on line 203 that is a method
argument
$progress_filter on line 219 traces back to
$progress on line 207 that is a method
argument


/bitnami/moodle/mod/charon/plugin/app/
Repositories/LabTeacherRepository.php:115
A method getTeacherReportByCourseId uses a
variant of DB raw() without bindings:
$prefix on line 132


/bitnami/moodle/mod/charon/plugin/app/
Repositories/SubmissionsRepository.php:558
A method findAllSubmissionsForReport uses a
variant of DB raw() without bindings:
$prefix on line 608
$prefix on line 609
$prefix on line 610
$prefix on line 611
$prefix on line 612
$prefix on line 613
A string with injected variable on line 610
A string with injected variable on line 613
```

Joonis 5. phpmd reegli väljund Charoni näitel.

3.4 Laraveli uuendamine

Kindlasti kuulub turvalisuse parendamise juurde pakettide ning eelkõige raamistike piisavalt sage uuendamine uuematele versioonidele. Praegu Charonis kasutusel olevale Laravel 5.5 versioonile väljastati turvaparandusi 2020. aasta 30. augustini[19] ning kõik hiljem ilmnevad või ilmnenud vead jäävad parandamata.

Uusim praegu väljas olev Laraveli versioon on 8, millele väljastatakse turvaparandusi 2022. aasta septembrini[20] ning tuleks kiiremas korras kasutusele võtta. Seeläbi saab parandatud OWASP'is kirjeldatud teadaolevate haavatavustega komponentide kasutamise punkt ning seeläbi PHP Turvalisuse Nõuannete Andmebaasi tulemustes kajastuvad Laraveli turvavead. Muuseas lisandub seeläbi raamistikule Laraveli 5.6 versioonis lisatud Argon2 algoritmiga parooli räsimise võimalus[19] ning 5.8 versioonis lisandunud võimekus API *token*'eid säilitada SHA-256 räsidena. Varasemalt olid API *token*'id säilitatud tavatekstina.[21]

Septembris tasub juba kindlasti arendajatel uuendada Charonis kasutusel olev Laravel väljastatavale versioonile 9, millel on ka pikaajaline tugi ning turvaparandusi teostatakse 2024. aasta septembrini.[20]

3.5 Turvaohtude kontroll Gitlab'i pidevas integratsioonis

Turvalisuse analüüsi käigus sai kontrollitud kasutuses olevaid teeke ja raamistikke PHP Turvalisuse Nõuannete Andmebaasi vastu. Et edaspidi ei peaks arendajad ise sellele kontrollile pidevalt mõtlema, sai see kontroll üles seatud Gitlab'i pidevasse integratsiooni, mis turvaohtude korral ebaõnnestub ning peaks siis seeläbi andma arendajale märku, et mõni kasutuses olev teek võib olla ohtlik. Selleks sai lisatud faili gitlab-ci.yml lõik, mis on kajastatud joonisel 6.

```
security-advisories-vulnerabilities:
  stage: test
  image: umutphp/php-docker-images-for-ci:7.4
  script:
    - local-php-security-checker
```

Joonis 6. Uus lõik gitlab-ci.yml failis.

4 Kokkuvõte

Antud bakalaureusetöö eesmärk oli analüüsida TalTech ülikoolis Moodle'i platvormil kasutatava Charon pistikprogrammi turvalisust ja leida üles turvavead ning seejärel kirja panna juhend nende parandamiseks ja turvalisuse parendamiseks. Ühtlasi oli eesmärgiks leida meetmeid Charoni edaspidise arenduse juures uute turvavigade tekkimise vähendamiseks ning võimalusel nende meetmete rakendamine.

Eesmärgi esmase sammu saavutamiseks ehk turvalisuse analüüsimiseks tutvuti kõigepealt antud rakenduses kasutuses olevate raamistike, pakettide ning neis olevate turvavigadega ja ühtlasi üldlevinud turvavigade ning turvalisust tõstvate heade arendustavadega. Seda tehti esmalt pistikprogrammi lähtekoodi uurides ning seejärel ka praktiliselt rakendust kasutades ning selle turvalisust katsetades.

Turvalisuse parendamiseks sai lisatud rakendusse phpcs, phpmd ja PHP Turvalisuse Nõuannete Andmebaasi rakendamine arenduse protsessi. Phpmd jaoks sai kirjutatud täiesti uus reegel SQL-süstimisründe vältimise aitamiseks ning PHP Turvalisuse Nõuannete Andmebaasi vastu rakenduse kontrollimine sai lisatud Gitlab'i pidevasse integratsiooni.

Kasutatud kirjandus

[1] OWASP [Online] https://owasp.org/ 17.05.2021

[2] OWASP Top Ten [Online] https://owasp.org/www-project-top-ten/ 17.05.2021

[3] OWASP – ega ometi o-herilane? [Online] https://blog.ria.ee/owasp-ega-ometi-o-herilane/ 17.05.2021

[4] PHP Data Objects: Introduction[Online] https://www.php.net/manual/en/intro.pdo.php 17.05.2021

[5] Database: Query Builder [Online] https://laravel.com/docs/5.5/queries 17.05.2021

[6] Laravel Queries: Raw Expressions [Online] https://laravel.com/docs/5.5/queries#raw-expressions 17.05.2021

[7] MySQL SQL Injection Practical Cheat Sheet [Online] https://perspectiverisk.com/mysql-sql-injection-practical-cheat-sheet/ 17.05.2021

[8] Potential dangers [Online] https://vuejs.org/v2/guide/security.html 17.05.2021

[9] PHP Security Advisories Database [Online] https://github.com/FriendsOfPHP/security-advisories 17.05.2021

[10] Local PHP Security Checker [Online] https://github.com/fabpot/local-php-security-checker 17.05.2021

[11] Laravel HTTP Session [Online] https://laravel.com/docs/5.5/session 17.05.2021

[12] Laravel Cookie Security Releases [Online] https://blog.laravel.com/laravel-cookie-security-releases 17.05.2021

[13] Laravel Eloquent: Getting Started [Onlne] https://laravel.com/docs/5.5/eloquent, 17.05.2021

[14] Security Release: Laravel 6.18.34, 7.23.2 [Online] https://blog.laravel.com/security-release-laravel-61834-7232 17.05.2021

[15] PHP CodeSniffer [Online] https://github.com/squizlabs/PHP_CodeSniffer 17.05.2021

[16] PHPMD - PHP Mess Detector [Online] https://phpmd.org/ 17.05.2021

[17] PMD Source Code Analyzer [Online] https://pmd.github.io/ 17.05.2021

[18] How to write a Rule for PHPMD [Online] https://phpmd.org/documentation/writing-a-phpmd-rule.html 17.05.2021

[19] Laravel 5.6 Release Notes [Online] https://laravel.com/docs/5.6/releases 17.05.2021

[20] Laravel 8.x Release Notes [Online] https://laravel.com/docs/8.x/releases 17.05.2021

[21] Laravel 5.8 Release Notes [Online] https://laravel.com/docs/5.8/releases 17.05.2021

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[2]

Mina, Johannes Kunts

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Charon pistikprogrammi turvalisuse analüüs ja täiustamine”, mille juhendaja on Ago Luberg

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

25.05.2021

2 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – phpmd reeglistik

```
<?xml version="1.0"?>
<ruleset name="Charon phpmd ruleset"

xmlns="http://pmd.sf.net/ruleset/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://pmd.sf.net/ruleset/
1.0.0

http://pmd.sf.net/ruleset_xml_schema.xsd"
         xsi:noNamespaceSchemaLocation="

http://pmd.sf.net/ruleset_xml_schema.xsd">


    <exclude-pattern>./bin/</exclude-pattern>
    <exclude-pattern>./bootstrap/</exclude-
pattern>
    <exclude-pattern>./config/</exclude-
pattern>
    <exclude-pattern>./public/</exclude-
pattern>
    <exclude-pattern>./resources/</exclude-
pattern>
    <exclude-pattern>./storage/</exclude-
pattern>
    <exclude-pattern>./vendor/</exclude-
pattern>


    <rule name="RawQueryBindings"
class="RawQueryBindingsRule">
        <priority>3</priority>
    </rule>
</ruleset>
```

Joonis 7. phpmd reeglistiku sisu failis phpmd_ruleset.xml

Lisa 3 – phpcs reeglistik

```
<?xml version="1.0"?>
<ruleset name="charon">
    <!-- Exclude directories -->

<exclude-pattern>/public/bootstrap/</exclude-
pattern>
    <exclude-pattern>/public/config/</exclude-
pattern>
    <exclude-pattern>/public/public/</exclude-
pattern>

<exclude-pattern>/plugin/resources/</exclude-
pattern>

<exclude-pattern>/plugin/storage/</exclude-
pattern>
    <exclude-pattern>./vendor/</exclude-
pattern>


    <rule ref="PSR12"/>


</ruleset>
```

Joonis 8. phpcs reeglistiku sisu failis phpcs_ruleset.xml