EESTI INFOTEHNOLOOGIA KOLLEDŽ

Kermo Pajula

VEEBIRAKENDUSE TURVATESTIMINE CHIVALO.COM NÄITEL

Diplomitöö

Infotehnoloogia süsteemide administreerimise õppekava

Juhendaja: Elar Lang

Konsultant: Lembit Viilup

Tallinn 2013

Autorideklaratsioon

Deklareerin, et käesolev diplomitöö, mis on minu iseseisva töö tulemus, on esitatud Eesti Infotehnoloogia Kolledžile lõpudiplomi taotlemiseks IT süsteemide administreerimise erialal. Diplomitöö alusel ei ole varem õppekava lõpudiplomit taotletud.

Autor Kermo Pajula ……..

(allkiri ja kuupäev)

Töö vastab kehtivatele nõuetele

Juhendaja Elar Lang ……..

(allkiri ja kuupäev)

Sisukord

Joonised

Tabelid

Lühendite ja mõistete selgitused

- AES (*Advanced Encryption Standard*) – salajase võtmega krüpteerimismeetodi standard, mis kasutab 128, 192 ja 256-bitiseid võtmeid ning Rijndaeli algoritmi;
- ASVS (*Application Security Verification Standard*) – Rakenduste turvalisuse verifitseerimise standard, üks OWASP projektidest;
- BEAST (*Browser Exploit Against SSL/TLS*) rünne – rünne transpordikihi turbeprotokolli pihta turvatud liikluse pealt kuulamiseks.
- Chivalet – chivalo.com veebirakenduse andmebaasi kirje lisamiseks programmeeritud JavaScript lahendus;
- CSRF (*Cross-Site Request Forgery*) – autoriseerimata päringute sooritamine kasutaja nimel, keda rakendus usaldab;
- HTML (*HyperText Markup Language*) – kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks;
- idufirma – ettevõte, mis on loodud eesmärgiga leida uus skaleeruv ärimudel;
- JavaScript – kliendipoolne skriptimiskeel veebilehtede interaktiivsuse tõstmiseks;
- musta kasti (*black box*) meetod – turvatestimine lähtekoodi ja serveri konfiguratsiooni uurimata;
- OWASP (*Open Web Application Security Project*) – Avatud veebirakenduste turbeprojekt;

- RSA (*Rivest-Shamir-Adleman*) – algoritm, avaliku võtmega krüpteerimismeetod;
- SAMM (*Software Assurance Maturity Model*) – Tarkvara küpsuse tagamise mudel, üks OWASP projektidest;
- SHA1 (*Secure Hashing Algorithm*) – USA Riikliku Julgeolekuagentuuri poolt arendatud krüptograafiline räsimisfunktsioon;
- SQL *injection* – koodi süstimine veebirakendusse sooritamaks päringuid rakenduse andmebaasikihis;
- TC (*Threat Classification*) – Ohtude klassifitseerimine, WASC ekspertide poolt koostatud dokument-viitamisjuhend;
- TLS (*Transport Layer Security)* – transpordikihi turbeprotokoll;
- WASC (*Web Application Security Consortium*) – Veebirakenduste turvalisuse konsortsium;
- XSS (*Cross-Site Scripting*) – omavoliline skriptide käivitamine rünnatavas veebirakenduses;

Sissejuhatus

Kaasaegses idufirmade ühiskonnas keskendutakse veebirakenduste loomisel võimalikult kiirele arendusele ja toote või teenuse lansseerimisele. Pahatihti mõeldakse üllitise turvalisusele liiga hilja, kui üldse ning vaeva ja hoolega kasvatatud hinnalise kasutajabaasi usaldus selle vastu on kiire kaduma.

Chivalo.com on autori praktikaettevõtte poolt arendatav veebirakendus, mis võimaldab kasutajatel luua kingisoovide nimekirju ning pakkuda nende jagamiseks ja teiste kasutajatega lävimiseks sotsiaalne platvorm. Rakendus töötab Play! raamistikul programmeerituna Java keeles ning kasutab Ebean andmebaasilahendust.

Kuna ettevõtte näol on tegemist idufirmaga ning olemasolev tööaeg leiab kasutust arenduses, ei ole ettevõtte juhtkond kursis rakenduse turvalisusega ning kasutajate andmed võivad olla kaitsmata. Chivalo.com on antud hetkel ettevõtte jaoks ainuke äri ning seega on tegu kriitilise tähtsusega süsteemiga.

Käesoleva diplomitöö eesmärk on anda ülevaade veebirakenduse chivalo.com turvalisusest sooritades turvatestimine ühe standardi järgi kolmest enamlevinud ja üldtunnustatud standardist. Dokumenteeritakse turvatesti protsess ning leitavad tugevused ja nõrkused, viimaste parandamiseks pakutakse välja lahendused.

Kuna ülesanne turvatestimiseks anti autorile ettevõttepraktika lõpuosas, sätestati sellest tulenevalt diplomitöö skoop arvestades ettevõtte juhatuse soove ja tingimusi. Vaatluse alla võetakse kirje lisamine andmebaasi Chivalet-nimelist lahendust kasutades ning kuna mainitud funktsionaalsuse kasutamiseks peab kasutaja olema chivalo.com lehele sisse logitud, uuritakse ka Chivaleti kaudu veebirakendusse sisse logimise turvalisust.

Diplomitöö on suunatud nii arendajatele kui ka administraatoritele. Töö kirjeldab tehnilist laadi turvatestimise protsessi ning nõuab lugejalt teadmisi veebirakenduste turvalisuse valdkonnas.

Töö jaguneb neljaks peatükiks. Esimeses peatükis kirjeldatakse testimise aluseks olevat veebirakendust ning viiakse lugeja kurssi töö skoobiga. Teises tutvustatakse kolme testimiseks valikus olnud üldtunnustatud standardit ning põhjendatakse käesoleva töö raames järgmiseks võetud standardi valik.

Kolmandas peatükis kirjeldatakse veebirakenduse adekvaatsust standardis esitatud nõuetega ning peatüki viimases alapunktis võetakse kokku rakenduse funktsionaalsuse testimisel leitud nõrkused. Neljandas peatükis pakutakse viimaste parandamiseks võimalikud lahendused.

Autor tänab töö juhendajat asjalike nõuannete ja tagasiside eest.

1. Chivalo.com

Chivalo.com on ettevõtte Bitsnbrains S.L. poolt arendatav veebirakendus, mis võimaldab kasutajatel luua kingisoovide nimekirju. Lisaks on võimalik neid oma sõpradega jagada, kommenteerida ja soovitada. Chivalo.com on diplomitöö valmimise ajal vaid hispaaniakeelne. Rakenduse toimimise eesmärk ja põhimõte jagunevad viieks punktiks [8]:

- lisa oma soov nimekirja;
- jaga seda oma sõpradega;
- saa endale oma unistuste kingitus;
- leia ideid oma sõprade kingituste jaoks;
- kingi alati midagi perfektselt sobivat.

1.1 Olukorra kirjeldus

Bitsnbrains S.L. näol on tegu on idufirmaga, kus enamik tööajast panustatakse uue funktsionaalsuse arendusse. Seetõttu puudub ülevaade veebi turvalisusest. Antud rakendus on ettevõttele kogu nende äri, seega on tegu selle ettevõtte jaoks kriitilise tähtsusega süsteemiga. Kui turvaprobleemide tõttu kasutajate usaldus kaotatakse, võib seda olla väga raske tagasi võita.

Rakendus on arendatud Play! raamistikule Ebean andmebaasilahendust kasutades, seega võivad leiud olla laiendatavad ka teistele sarnastele.

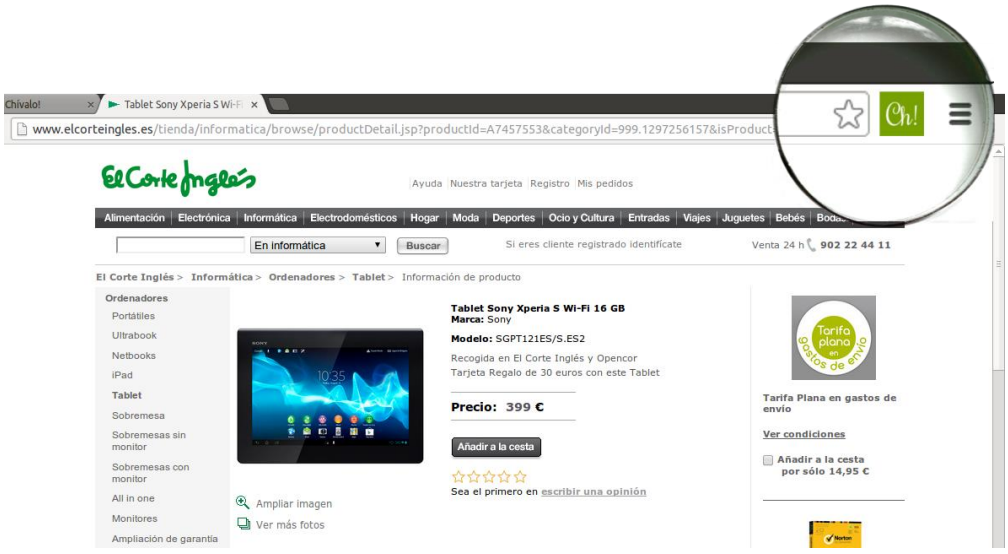
1.2 Testitav funktsionaalsus

Sooritades diplomitöö kirjutamise ajal ettevõttepraktikat rakendust arendavas ettevõttes, anti autorile ülesanne turvatestimiseks praktikaperioodi lõpus. Tulenevalt sellest esitati soovid ja tingimused testitava funktsionaalsuse osas.

Rakenduse turvatestimisel võetakse vaatluse alla andmebaasi kirje lisamise tehniline funktsionaalsus, mis on mõeldud toimima kasutades Chivalet-nimelist lahendust. Chivalet on serveriga suhtlev JavaScript programmikood, mida on võimalik käivitada järjehoidjaribalt või brauseri laiendusena. Kuna kasutaja peab olema antud lahenduse kasutamiseks chivalo.com lehele sisse logitud, vaadeldakse ka sisselogimise turvalisust.

Kirje lisamine andmebaasi käib läbi kolme sammu [8]:

1. Olles soovitavat toodet sisaldaval veebilehel, tuleb käivitada Chivalet (Joonis 1).



Joonis 1 - Chivaleti käivitamine

2. Kui ollakse sisse logitud chivalo.com veebilehele, tuleb valida sobiv pilt toote või teenuse kohta (Joonis 2). Vastasel juhul on võimalik avanevas aknas sisselogimine ning seejärel oma tegevuse jätkamine.

Joonis 2 - Pildi valimine

3. Lisada täpsed andmed ja valida nimekiri, kuhu toode lisada (Joonis 3). Privaatse nimekirja või toote puhul see avalikult nähtav pole. Siin toimub suhtlemine serveri ja andmebaasiga.

Joonis 3 - Täpsete andmete lisamine

2. Standardi valik

Kuna Play! raamistiku osi on toetajate poolt testitud OWASP Top 10 järgi [10, 11], kavatsetakse testida rakendust ühe standardi järgi kolmest: OWASP ASVS, OWASP SAMM või WASC TC. Tegu on üldtunnustatud standarditega, mille põhjal tehtud järeldused saab üldistada sarnastele rakendustele.

2.1. Open Web Application Security Project

OWASP (*Open Web Application Security Project*) on ülemaailme heategevuslik projekt, mille eesmärk on parandada veebirakenduste turvalisust tõstes teadlikkust tarkvara turbe osas nii eraisikute kui ka ettevõtjate seas, et nad oskaksid langetada kaalutletud otsuseid turvariskide osas. OWASP kogukond toodab kõigile vabalt kättesaadavaid artikleid, metodoloogiaid, dokumentatsioone, tööriistu ja tehnoloogiaid veebiturvalisuse huvides. OWASP ei ole seotud ühegi tehnoloogiaettevõttega, kuna seotus erasektoriga muudab veebirakenduste turvalisuse kohta erapooletu, praktilise ja tasuva sisu tootmise raskemaks [1].

OWASP projektid katavad mitmeid rakenduste turvalisusega seotud aspekte. Nende eesmärk on pakkuda dokumente, tööriistu, õppekeskkondi, juhiseid ja muid materjale aitamaks organisatsioone tõhustada oma võimet toota turvalist tarkvara [2].

2.1.1. Application Security Verification Standard

ASVS (*Application Security Verification Standard*) on üks OWASP projektidest, mille peamine eesmärk on leevendada kaost valmis rakenduste turvatestimise ümber. ASVS on kava, mida järgides rakenduste testimisel on kerge kindlaks teha nii vead äriloogikas, kui ka tehnilised nõrkused, mida võivad ära kasutada näiteks XSS (murdskriptimine) ja SQL *injection*. Standardit võib kasutada kehtestamaks kindlustunnet oma rakenduse turvalisuse kohta. Nõuded loodi järgnevaid sihte silmas pidades [3]:

- meetrika - pakub tarkvaraarendajatele ja rakenduste omanikele kriteeriumid, mille alusel hinnata võimalikku usalduse taset oma rakenduste puhul;
- juhendamine - annab juhiseid arendajatele turvakontrollide loomiseks ning viitab, millele tähelepanu pöörata, et rakenduste turvanõuetele vastavalt rakendusi luua;
- hanke korraldamine - võimaldab rakenduste tellimisel tarkvaratootjatelt lepingusse täpselt kirja panna turvanõuded, millele rakendus peab vastama.

ASVS tasemed on [4]:

- tase 1 - automaattestimine;
    - tase 1A - dünaamiline skaneerimine;
    - tase 1B - lähtekoodi skaneerimine;
- tase 2 - manuaalne testimine;
    - tase 2A - turvatestimine;
    - tase 2B - lähtekoodi analüüsimine;
- tase 3 - disaini testimine;
- tase 4 - sisemine testimine.

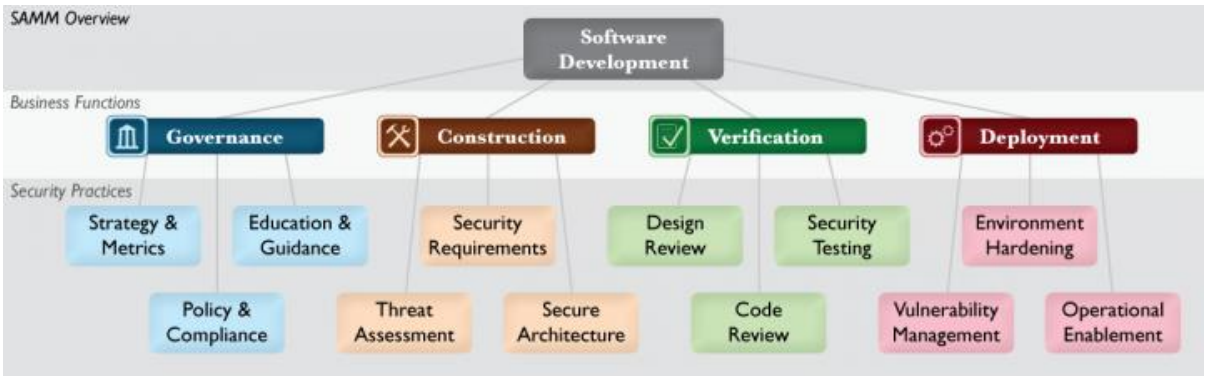
Valitud tasemel testitakse rakenduse vastavust järgmiste nõuetega [7]:

- V1 - turbearhitektuuri dokumentatsiooni nõuded;
- V2 - autentimise kontrolli nõuded;
- V3 - sessioonihalduse kontrolli nõuded;
- V4 - ligipääsukontrolli verifitseerimise nõuded;

- V5 - sisendi valideerimise nõuded;
- V6 - väljundi kodeerimise nõuded;
- V7 - krüptograafia verifitseerimise nõuded;
- V8 - vigade käsitlemise ja logimise nõuded;
- V9 - andmete kaitsmise nõuded;
- V10 - side turvalisuse nõuded;
- V11 - HTTP turvalisuse nõuded;
- V12 - turvakonfiguratsiooni nõuded;
- V13 - kuritahtliku koodi otsimise nõuded;
- V14 - sisemise turvalisuse nõuded.

2.1.2. Software Assurance Maturity Model

SAMM (*Software Assurance Maturity Model*) on avatud raamistik aitamaks organisatsioonidel formuleerida ja implementeerida strateegiaid tarkvara turvalisuse tagamiseks. SAMM on abiks ettevõtte olemasolevate turvapraktikate evalveerimiseks, tarkvara turvalisuse hindamise programmi parendamiseks konkreetsete näidete põhjal ja turvalisusega seotud tegevuste defineerimiseks ja mõõtmiseks organisatsioonis. SAMM loodi silmas pidades paindlikkust, et seda oleks võimalik rakendada nii väikestes kui ka suurtes ettevõtetes hoolimata arendusstiilist. Lisaks on võimalik seda mudelit kasutada terves ettevõttes või ainult ühe projekti raames [5]. Mudel on lahti selgitatud järgneval joonisel (Joonis 4).



Joonis 4 - SAMM

2.2. Web Application Security Consortium

WASC (*Web Application Security Consortium*) on tööstuspraktikutest ja firmajuhtidest koosnev rahvusvaheline ekspertgrupp, mis arendab ja annab välja avatud üldtunnustatavaid turvastandardeid lähtuvalt parimatest praktikatest Internetis. Aktiivse kogukonnana avaldatakse järjepidevalt tehnilist informatsiooni, artikleid ja turvajuhiseid, mida kasutavad ettevõtted, haridusasutused, valitsused, tarkvaraarendajad ja turvaspetsialistid üle maailma.

WASC eesmärgiks on harida turgu ja luua avatud suhtluskeskkond loomaks, diskuteerimaks ja levitamaks teadmisi veebirakenduste turvalisusest. Nende põhimõtete vastu on tootjapõhiste tehnoloogiate, teenuste ja lahenduste propageerimine [8].

2.2.1. Threat Classification

TC (*Threat Classification*) on WASC ekspertide ühine panus toomaks selgust veebi turvalisuses ja klassifitseerimaks veebirakendusi varitsevaid ohte ja nõrkusi. TC projekt loodi selleks, et arendada ja edendada veebiturva tööstusharus standardne terminoloogia juhtumite kirjeldamiseks. Tarkvaraarendajatel ja turvaspetsialistidel on ligipääs nimekirjale järjepidevalt identifitseeritavatest ja defineeritavatest turvaprobleemidest [6].

Threat Classification v2.0 on oma olemuselt dokument, mis käitub peamiselt kui viitamisjuhend raportite ja presentatsioonide jaoks. TC materjal on ilmunud paljudes raamatutes ja seda kasutavad paljud firmad, teiste seas IBM ja HP. Sooritades

turvatestimist, saab kasutada seda kontrollnimekirjana uurides testimisobjekti reageerimist kirjeldatud ohtudele. Mõningates ettevõtetes on TC definitsioonid kasutusel rikete jälgimissüsteemides kogudes informatsiooni nõrkuste ekspluateerimise tiheduse või rünnete sooritamise aegade kohta [7].

2.3. Kokkuvõte

OWASP SAMM on väga paindlik ja sobib nii väiksemate kui ka suuremate ettevõtete jaoks hoolimata kasutusel olevast arendusstiilist, kuid on planeeritud pigem juba eksisteerivate turvapraktikate ja strateegiate arendamiseks.

WASC TC on väga täpne klassifitseeritud nimekiri teadaolevatest veebirakendusi varitsevatest nõrkustest ja ohtudest ning selle järgi testimise jaoks on vaja diplomitöö raames liiga palju aega. Otstarbekam on seda dokumenti kasutada viitamismaterjalina turvatestide raportite koostamisel.

OWASP ASVS pakub teemaks oleva võrdlemisi uue rakenduse testimiseks sobiva meetrika ning võimaldab ettevõttel saada selge ülevaade veebi turvalisusest. ASVS abil on võimalik tuvastada nii vigu äriloogikas kui ka tehnilises funktsionaalsuses, mis mõlemad on antud ettevõtte jaoks äärmiselt olulised.

Sünteesis rakendatakse OWASP ASVS tase 2A musta kasti meetodil (manuaalne turvatestimine lähtekoodi ja serveri seadistust uurimata) diplomitöö mahu piiratuse tõttu ning kuna soovitakse jäljendada ründaja positsiooni. Ründaja võib küll välja uurida kasutatava raamistiku, kuid isegi selle lähtekoodi tundes ei teki tal edumaa äriloogikavigade ekspluateerimiseks ning raamistiku lähtekood on oma laia leviku tõttu OWASP Top10 järgi turvatestitud (vt. peatükk 2).

3. Turvatestimine

Diplomitöö analüüsiosa tulemusena (vt. alapunkt 2.3, lk 17) võetakse testimisel järgmistes alapeatükkides järgimiseks OWASP ASVS taseme 2A musta kasti meetod (manuaalne turvatestimine lähtekoodi ja serveri seadistust uurimata).

Lähtuvalt ettevõtte poolt seatud tingimustest, sooritatakse turvatestimine testimiseks loodud keskkonnas, mille aluseks on reaalse keskkonna programmikood ja andmed ning töötab kohtvõrgus üles seatud testserveris.

Testserveri konfiguratsioon ühtib reaalajas veebirakendust töös hoidva serveri konfiguratsiooniga.

ASVS nõuded valdkondade kaupa jagunevad spetsiifilisteks nõueteks, millele vastavust kirjeldatakse vastavates järgnevates alapeatükkides. Antakse ülevaade tegevustest rakenduse nõuetele vastamise kontrollimiseks ning tuvastatavatest nõrkustest. Mainitakse neid punkte ASVS standardist, mida musta kasti meetodi piirang käsitlemast ei takista. Kõik nõuded alapeatükkide kaupa ja hinnang nendele vastavuse kohta koondtabeli kujul asub lisas (Lisa 1).

Kava nõrkuste parandamiseks pakutakse välja töö viimases peatükis (vt. peatükk 4, lk 32).

3.1. V1 - Turbearhitektuuri dokumentatsiooni nõuded

Kõigi ASVS tasemete jaoks on vajalik rakenduse turvaarhitektuuri dokumentatsioon tagamaks terviklikkus ja täpsus ASVS nõuetele vastavuse verifitseerimisel [4].

Esimene tingimus nõuab, et kõik rakenduse komponendid oleksid identifitseeritud. Musta kasti piirangu tõttu uuriti vaid rakenduse avalehe HTML dokumenti. Selles sisaldus mitu viidet JavaScript ressursile, mille kasutamist lehel ei tuvastatud (jquery.ba-dotimeout.min.js, native.history.js).

Google Chrome brauseri DevTools [12] konsoolis kuvatakse iga lehe laadimisel veateade rakenduse toimimiseks vajaliku komponendi puudumise kohta `"NetworkError: 404 Not Found - https://chivalo.com/piwik/js/"`.

3.2. V2 - Autentimise kontrolli nõuded

Autentimise kontrolli nõuded käsitlevad turvaliselt kasutajakonto andmete genereerimise ja haldamise tingimusi [4]. Kontrolliti, kas mõnda autentimist nõudvat toimingut on võimalik teostada ilma keskkonnale ligipääsu omamata ning kas kuritahtlikult ligipääsu omandamise takistamiseks on seatud vajalikud tõkked.

Ühtegi eset oma soovinimekirja sisse logimata lisada ei õnnestunud. Chivaleti kasutades ning otse brauseri aadressiribalt päring sooritades suunati kasutaja logimisvormini ja kirjet andmebaasi ei lisatud.

Sisse logides ei olnud võimalik näha paroolivälja eelnevalt sisestatud paroole, küll aga oli *autocomplete* lubatud. Vale parooliga sisselogimist prooviti sada korda nelja minuti jooksul (nii palju aega kulus saja katse sooritamiseks) jõuründe simuleerimiseks, kontot selle tulemusena ei lukustatud.

Autentimiskontroll teostatakse kirje lisamisel välise vaatluse järeldusena (lähtekoodile ligipääs puudub musta kasti meetodi piirangu tõttu) kliendi poolel, kuna Chivaleti käivitusskript kontrollib kasutaja brauseris sessiooni kehtivust näitava küpsise olemasolu.

Sisselogimisel saadetakse POST meetodil HTTP päring serverile ning kasutajanime ja parooli kattuvust kontrollitakse serveri poolel.

Autentimise kontrolli ebaõnnestumisel kuvatakse kasutajale veateade ning õiguseid ei anta. Sisselogimisel liiga lühikene parool sisestades on teateks „*Longitud mínima 5*“ („Minimaalne pikkus 5“). Kasutajanime ja parooli mitte klappimisel kuvatakse teade „*Usuario o contraseña incorrecto*“ („Kasuajanimi või parool on vale“). Veendumaks, et õiguste olemasolu kontrollitakse kirjetega toimingute sooritamisel, prooviti teise kasutaja soovinimekirjast ese kustutada. Kuvati veateade „*No tienes permiso para estar aquí*“ („Sul ei ole luba siin viibida“).

Parooli tugevuse kontrolli implementeeritud ei ole. Ei nõuta suurte ja väikeste tähtede ning numbrite sisaldamist. Ainukese tingimusena peab parool olema vähemalt 5 märki pikk ning selleks lubati seada „aaaaa“. Inglise keelse tähestiku puhul on erinevaid tähti 26 ja seega on erinevaid võimalusi 26^5 = 11881376. Radeon 5770 graafikakaardi protsessoril kulub sellise parooli ära arvamiseks jõuründel $26^5/(3{,}3*10^9)$ = 0.036 sekundit [14]. Kuna on võimalik seda parooliks ka ainult numbrid, on sellisel juhul erinevaid võimalusi 10^5 = 100000 ning selle murdmiseks kulub 1/30000 sekundit [14].

Kontohalduse funktsioonide turvalisuse testimiseks prooviti näha teiste kasutajate privaatseid tooteid ning lisada toode teise kasutaja nimekirja. Selleks prooviti vastavalt muuta brauseri aadressiribal toote ID tähistavat numbrit ning muuta toote lisamisel nimekirja POST meetodil HTTP päringus nimekirja ID numbrit. Kummalgi juhul päring edukas ei olnud ning kasutajale kuvati veateade või suunati tagasi lehele, kust ta tuli.

Kasutaja andmete muutmise turvalisuse testimiseks prooviti muuta teise kasutaja andmeid ning uurida, kas tundlikku informatsiooni lekib kasutaja andmete muutmise päringute käigus. Andmete muutmisel ei saadeta päringutes serverile ei kasutajanime ega ID numbrit, seega ei suudetud ligi pääseda teise kasutaja andmete haldusele ning ei tuvastatud tundliku info lekkimist.

Uuesti autentimist nõutakse kasutajalt parooli muutmisel kehtiva parooli sisestamise näol, muude toimingute käigus ei ole vajalik ennast teist korda autentida.

Kasutaja konto andmed ei aegu kunagi ning sessioon ei aegu enne brauseri sulgemist. Ennast uuesti autentida on vaja juhul, kui brauseris ei eksisteeri „jäta mind meelde“ küpsis. Vastasel juhul genereeritakse sellest lähtuvalt uus sessiooniküpsis.

3.3. V3 - Sessioonihalduse kontrolli nõuded

Sessioonihalduse kontrolli nõuded sisaldavad endas HTTP päringute, vastuste, sessioonide, küpsised, päiste ja logimise kasutamise nõudeid sessioonide turvaliseks haldamiseks [4].

Musta kasti meetodi piirangu tõttu saab vaid väita, et raamistiku vaikimisi sessioonihalduse kontroll on rakenduses tõenäoliselt implementeeritud, kuna kasutusel on küpsis nimega „PLAY_SESSION“. Selle olemasolu näitab sessiooni kehtivust ning sisaldab raamistiku poolt genereeritud räsi. Kui kasutaja rakendusest välja logib, märgitakse küpsis brauseris kehtetuks, kuid selles sisalduvad andmed serveris oma kehtivust ei kaota. Kui ründaja arvutis olevasse brauserisse sama infot sisaldav küpsis luua, ollakse automaatselt lehele antud kasutajana sisse logitud ning sama kehtib ka „jäta mind meelde“ küpsise kohta.

Sessioon aja möödudes oma kehtivust ei kaota, kuna sessiooni infot sisaldav küpsis kehtib kuni brauser suletakse. Kui kasutaja on valinud võimaluse „jäta mind meelde“, luuakse vastavasisuline küpsis, mille olemasolul brauseris rakenduse uuesti avamisel sellest lähtuvalt uus sessiooniküpsis genereeritakse.

Väljalogimise nupp asub rakenduses autentimist nõudvatel lehtedel üleval paremas nurgas, küll aga puudub see Chivaleti aknas.

Sisse logides sessiooni ID muutumise kontrollimiseks jäeti meelde eelmise sessiooni ID ning tuvastati, et sisse logides genereeritakse alati küpsisele uus väärtus, milles sisaldub uus ID (eelmine jääb aga kehtima). Uuesti autentimisel ei taastata sama sessiooni, vaid alustatakse uus. Välja logimisel küpsis märgitakse brauseris kehtetuks, sessiooni andmed serveris aga kehtivust ei kaota.

Play! raamistik märgistab sessiooni HMAC-SHA1 algoritmiga räsitud 64 prinditavast ASCII tähemärgist koosnevat võtmega (vähemalt 384 bitti) [15]. Sessiooniküpsise sisu muutmisel mistahes enda välja mõeldud tähekombinatsiooni vastu, logiti kasutaja välja ning nõuti uuesti autentimist.

3.4. V4 - Ligipääsukontrolli verifitseerimise nõuded

Ligipääsukontrolli verifitseerimise nõuetes on sätestatud, kuidas peaksid veebirakendused ligipääsu inspekteerima erinevates kohtades läbi erinevate rakenduskihtide. Defineeritakse nõuded URL-ide, ärifunktsioonide, andmete, teenuste ja failide testimiseks [4].

Õiguste piiramise ja ainult autenditud kasutajale mõeldud URL päringute sooritamisõiguste kontrollimiseks prooviti lisada toode teise kasutaja soovinimekirja ning mõni toode sealt kustutada. Toote lisamiseks modifitseeriti Chivaleti poolt teostatavat POST meetodil HTTP päringut ning muudeti nimekirja number. Kasutajale veateadet ei kuvatud, kuid toimus tagasisuunamine lisamisvormini ja kirjet andmebaasi ei lisatud. Toote kustutamiseks muudeti vastavas GET meetodil tehtavas HTTP päringus kirje ID numbrit, mille tulemusena kuvati kasutajale veateade „*No tienes permiso para estar aquí*“ („Sul ei ole luba siin viibida“).

Chivaleti aknas toote lisamise vormil kuvatakse valik võimalikest valuutadest. Failide muutmisõiguste kontrollimiseks prooviti lisada nimekirja lisaks EUR ja USD vääringutele JPY modifitseerides Chivaleti HTML faili kliendi poolel. Vääring ilmus valikusse, kuid vormi postitamisel serveripoolne kontroll seda läbi ei lasknud ning kasutaja suunati tagasi lisamisvormini. POST meetodil HTTP päringu vastuseks oli „400 Bad Request“.

Objektide kaitse testimiseks üritati näha teise kasutaja privaatseid tooteid muutes GET meetodil HTTP päringus toote ID numbrit, mille tulemusena kuvati kasutajale veateade „*No tienes permiso para estar aquí*“ („Sul ei ole luba siin viibida“). Kataloogide lehitsemine ei olnud samuti edukas, üritades ligi pääseda kataloogidele `chivalo.com/assets/stylesheets`, `chivalo.com/assets/img` või

`chivalo.com/assets/javascript`, kuvati kasutajale valge leht ning nende sisu näha ei õnnestunud.

Sisendi limiteerimine äriloogika turvalisuse tagamiseks on edukas lisatavate piltide kontrollimisel. Liiga suur või vale sisuga (faililaiendit ei kontrollita, pildid töödeldakse ImageMagick programmi poolt ja tuvastatakse failiformaat ning rikutud pildid, mis sisaldavad näiteks programmikoodi) pilt tuvastatakse filtrite poolt ja kasutaja näeb veateadet „*errors.image_fetch_failed*“ („Pildi toomine ebaõnnestus“).

Ligipääsukontrollid toimuvad serveri poolel, kuna veateated või märgid päringu ebaõnnestumisest ilmnesid pärast päringute sooritamist. Nende ebaõnnestumine on turvaline ja tundliku informatsiooni leket ei tuvastatud. Toote lisamisel teise kasutaja soovinimekirja ja valuuta nimekirja omavolilisel täiendamisel kasutajale veateadet ei kuvatud. Üritades ligi pääseda teise kasutaja privaatsele kirjele, kuvati veateade „*No tienes permiso para estar aquí*“ („Sul ei ole luba siin viibida“). Kataloogide lehitsemisel on ainus tulemus valge leht.

3.5. V5 - Sisendi valideerimise nõuded

Antud alapunkti nõuded kätkevad endas puhvri täitumise, valideerimismustri sisendi ohtlikest sümbolitest puhastamise, märgistiku ja valideerimise nõuete kontrollimist [4]. Selleks prooviti Chivaleti väljadesse ning sisselogimisvormi kirjutada lisaks tähtedele ja numbritele erinevaid sümboleid ning pildi asemel üles laadida muid faile.

Keskkond ei ole aldis puhvri ületäitumisele, kuna 342MB suuruse pildi lisamisel kuvati veateade „*errors.image_fetch_failed*“ („Pildi toomine ebaõnnestus“) ning kirje andmebaasi ei jõudnud. Sama teade ilmnes ka sellisel juhul, kui pildi asemel üritati serverisse üles laadida JavaScript või PHP fail.

Sisendi kontrolli ebaõnnestumisel kuvatakse kliendi poolel viga punase piirjoone näol sisestuskasti ümber siis, kui tootele üritatakse lisada liiga pikk pealkiri. Kasutajale antakse märku valest paroolist või kasutajanimest sisselogimisel ning väljadesse sisestatud ebastandardsetest kirjamärkidest ja sõnedest SQL *injection*-i jaoks (Lisa 2) – „*Usuario o*

contraseña incorrecto“ („Kasutajanimi või parool on vale“), mille kontroll toimub serveri poolel POST meetodil HTTP päringut töödeldes.

Tootele hinna lisamisel on kasutajakogemus halvatud, kuna ignoreeritakse kõiki nupuvajutusi peale numbrite. POST meetodil HTTP päringut manipuleerides ja sinna tähti lisades veateadet ei kuvata ning kasutaja suunatakse tagasi vormile, kirjet andmebaasi ei lisata. Lubatakse sisestada 12 numbrit enne koma ja 2 numbrit pärast koma. Samasugune ilma veateateta tagasisuunamine toimub ka juhul, kui kirjelduse väljale kirjutatakse liiga palju teksti (limiiti kasutajale ei kuvata ning seda ei suudetud tuvastada, testimiseks sisestatud teksti pikkus oli 5000 tähemärki).

Google Chrome brauseri DevTools [12] võimaluste abil tehti kindlaks, et igale sisendiväljale on määratud UTF-8 märgistik. Sisendandmete vastavust nõuetele kontrollitakse serveri poolel, kuna nii toote lisamisel nimekirja kui ka sisselogimisel saadetakse info serverile POST meetodil HTTP päringuga ning veateated kuvatakse pärast selle töötlemist ning lehe uuesti laadimist.

3.6. V6 - Väljundi kodeerimise nõuded

Rakenduse väljundi kodeerimise nõuetele vastavuse kontrollimiseks sisestati toote lisamisel oma soovinimekirja andmeväljadesse HTML sümboleid ja JavaScript skripte, mida on kindlasti vaja kasutajale HTML dokumendi koostamisel ja väljastamisel varjundada või konverteerida HTML spetsiaalsümboliteks [4].

Skriptimärgendid sisestati nii toote pealkirja, kirjelduse kui ka nimekirja nime väljadesse. Andmete kuvamisel HTML dokumendis *body* osas olid kõikidel lehtedel potentsiaalselt ohtlikud sümbolid konverteeritud.

XSS turvaauk tuvastati toote pealkirja toomisel metaandmetesse, kus oli võimalik *meta* märgend saboteerida kujul `"><a href="javascript:alert(0);">link</a>`. Selliselt tekkis hüperlink *body* märgendi algusesse, mis käivitas etteantud JavaScript skripti. Soovinimekirja nime andmevälja kirjutades `"><script>alert(0);</script>` kasutati samuti ära metaandmete ebakorrektset kuvamist ning antud nimekirja vaate

laadimisel käivitus automaatselt etteantud skript. Tegu on tõsise turvaauguga, mis võimaldab pahatahtlikul kasutajal serveris käivitada skripte manipuleerimaks ohvrile koostatava ja kuvatava lehekülje sisu ja käitumist, nagu ka näiteks teiste kasutajate sessioonide andmete varastamiseks, mis võib võimaldada hiljem neid andmeid kasutades ohvri sessioon kaaperdada.

3.7. V7 - Krüptograafia verifitseerimise nõuded

Krüptograafia verifitseerimise nõudeid kasutatakse kontrollimaks rakenduse krüpteerimisfunktsioone, võtmehaldust, suvanumbri genereerimist ja räsimist. Soovitusena tuuakse välja, et rakendused peaks alati kasutama FIPS 140-2 või samaväärse standardi järgi valideeritud krüptograafiamooduleid [4].

Musta kasti piirangu tõttu ei olnud võimalik käesoleva töö teemaks oleva rakenduse nõuetele vastavust testida.

3.8. V8 - Vigade käsitlemise ja logimise nõuded

Vigade käsitlemise ja logimise nõuetele vastavuse verifitseerimiseks tuleb veenduda, et rakendus ei väljasta veateadete tekkimisel tundlikku informatsiooni, näiteks sessiooni identifikaatori numbrit või pinu, tavakasutajale. Samuti sätestatakse tingimused vigade käsitlemiseks ning logimiseks [4].

Kõik veateated, mis ei ole seotud pildi laadimisel tekkinud tõrgetega, on genereeritud Play! raamistiku poolt serveri poolel pärast päringute sooritamist ja need ei paljasta kasutajale informatsiooni, mida nad ei peaks nägema.

Veateadete esile kutsumiseks sooritati päring kirje lisamiseks andmebaasi sisestades Chivaleti käivitav URL otse brauseri aadressiribale ja manipuleeriti seal olevaid parameetreid. `http://chivalo.com/submit` aadressil teatatakse, et muutujaid on puudu. Kui lisada veateadete märkmeid järgides vajalikud `?url=x&image`, kuvatakse kasutajale Chivalet tühjade väljadega. Kui kasutaja pole sisse logitud, suunatakse vajalike

parameetrite olemasolul ta sisselogimisvormini. Tundlikku informatsiooni leket ei suudetud tuvastada.

Mitte üheski olukorras tõrgete esinemisel väära sisendi, ebaõnnestunud päringu või URL-i manipulatsiooni tõttu kasutaja ligipääsu ei keelatud.

3.9. V9 - Andmete kaitsmise nõuded

Alapunktis käsitletavatele nõuded käsitlevad tundliku informatsiooni turvalisust [4]. Tundlike andmete kaitsemehhanismid on nii Play! raamistiku kui ka arendajate poolt antud rakenduses implementeeritud. Uuriti nii sisselogimisvormi kui ka toote soovinimekirja lisamise vormi.

Sisselogimisvormi paroolivälјal ei ole võimalik kasutada sõna lõpetamise soovitusi, samas ei ole keelatud brauseri või kolmanda poole tarkvara poolt meelde jäetud paroolide automaatne sisestamine. Sisse logides on päringu vastusepaketis kirjas vajalikud parameetrid `Cache-Control=no-cache, no-store, must-revalidate` info vahemällu talletamise keelamiseks.

Kõik andmed saadetakse serverile parameetritena POST meetodil HTTP päringuga ning kirje andmebaasi lisamiseks ühtegi parameetrit HTTP päringu GET (ega ühelgi teisel) meetodil ei ole võimalik edastada. Ainsana toimub HTTP päringu GET meetodil parameetrite edastamine Chivalet-i käivitava skripti ja Chivalet-i enda vahel, kus saadetakse HTTP päringu GET meetodil pildi aadress, toote aadress ja toote nimi. Päringu vastusepaketis on määratud parameeter `Cache-Control=no-cache`.

3.10. V10 - Side turvalisuse nõuded

Side turvalisuse nõuded sätestavad tingimused, mille abil saab veenduda, et kasutaja suhtlemine rakendusega on turvaline [4].

Rakendus vastab kriteeriumitele teatud mööndustega, kuna nii sisselogimine kui ka kirje lisamine andmebaasi toimuvad üle TLS turvakanali. Sisselogimisel ei ole ebaturvalisele

ühendusele taandumine võimalik, küll aga võib kirje lisamisel manuaalselt HTTPS asemel tavalist HTTP protokolli kasutada.

Kasutusel on StartCom *Class 1 Primary Intermediate Server CA* TLS 1.1 sertifikaat, mis turvab ühenduse 256-bitise krüpteeringuga. Selleks rakendatakse AES_256_CBC krüpteerimisstandard koostöös SHA1 ja ECDHE_RSA võtmevahetusmehhanismiga. SSL kompressiooni ei kasutata.

Qualys SSL Labs SSL Server Test tööriista [22] abil tehti kindlaks, et chivalo.com on haavatav BEAST ründega, mis võib võimaldada ründajal pealt kuulata ka TLS turvakanalil saadetavat liiklust.

3.11. V11 - HTTP turvalisuse nõuded

HTTP turvalisuse verifitseerimine käsitleb HTTP päringute, vastuste, sessioonide, küpsiste, päiste ning logimise turvalisusega seotud nõuetele vastavuse kontrollimist [4].

Chivaletis võib toimuda kasutaja ümbersuunamine kolmes kohas:

- toote lisamisvormilt sisselogimisvormile;
- pärast sisselogimist lisamisvormile tagasi;
- lisamisvormilt päringu vastuse kuvani.

Valideerimata andmete edastamist ühelgi eelpool mainitud sammul ei tuvastatud.

Mozilla Firefox brauseri laienduse HTTP Tester abil tõestati, et rakendus ei aktsepteeri ühtegi HTTP päringu meetodit peale GET ja POST-i. Sooritades päringuid teiste HTTP meetoditega http://chivalo.com aadressil (näiteks PUT, TRACE), oli vastus „400 Bad Request“. GET ja POST puhul aga „404 Not Found“, mis näitab, et päringud aktsepteeriti.

Nõutakse, et HTTP päringutele antavates vastustes oleks spetsifitseeritud turvaline märgistik, nagu näiteks UTF-8. Märgistik oli määratud vastustes sisselogimise päringule HTTPS protokolli kasutades ja POST meetodil HTTP päringule toote lisamisel soovinimekirja.

Määratlus puudus vastuses, kui kasutaja polnud chivalo.com lehele sisse logitud ja toimus suunamine sisselogimisvormile.

Rakenduse poolt kasutatavatel küpsistel oli määratud vajalik *HTTPOnly* lipp, kuna neil ei ole nõutud ligipääsetavus JavaScript skriptidest. Soovituslik *secure* lipp puudus kõigil, kaasa arvatud sessiooniküpsisel.

HTTP päised nii päringutes kui vastustes sisaldavad ainult prinditavaid ASCII tähemärke. Kontrollimiseks võeti aluseks W3Schools poolt avaldatud artikkel [13] ning uuriti päringuid kasutaja sisse logimisel ning toote soovinimekirja lisamisel – nii õnnestunuid kui ebaõnnestunuid.

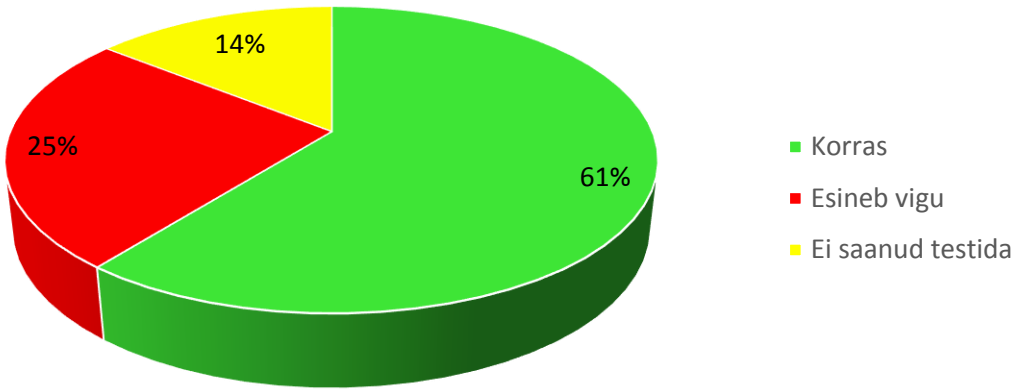
Saadetud POST meetodil HTTP päringut hakatakse töötlema, kui selle saatnud brauseris asub ka sessiooni infot sisaldav küpsis. Puudub kontroll, kas päringu sooritajaks oli Chivalet või oli see pärit tundmatust asukohast. Seega puudub kaitse CSRF rünnete vastu, kuna nendel juhtudel sooritab ründaja päringu kasutaja nimel ning kui kasutaja brauseris eksisteerib chivalo.com sessiooniküpsis, on päring autoriseeritud.

Küll aga töödeldakse andmed samade reeglite järgi, mis Chivaletist pärinevate andmete puhul (vt. alapunkt 3.5, lk 23) ning midagi pahatahtlikku korda saata ei õnnestunud.

3.12. Kokkuvõte

Chivalo.com veebirakenduse OWASP ASVS nõuetega vastavuse kontrollimisel arvestati igat nõuet võrdselt, kriitilised probleemid ei oma suuremat kaalu. ASVS taseme 2A järgi testiti rakenduse vastavust kokku 56 nõudega, millest positiivse hinnangu pälvis 34, vigu esines 14 nõude puhul ning testimiseks puudus piisav informatsioon või oli takistuseks musta kasti meetod 8 nõude jaoks.

Tulemused on visuaalselt toodud järgneval joonisel (Joonis 5) ning detailselt töö esimeses lisas (Lisa 1).



Joonis 5 – Nõuetele vastavuse statistika

Lehel tuvastati skripte, mis ei ole alati kasutusel ning sisaldavad kordusi. Skriptid tuleks laadida vaid nendes kohtades, kus neid rakendatakse. Mida rohkem lehel funktsionaalsust on, seda suurema tõenäosusega võib seal leiduda mõni ekspluateeritav nõrkus. Lisaks raisatakse internetiühenduse ressurssi ja koormatakse serverit. Tuvastatavatest komponentidest oli puudu statistikamootori jaoks vajalik skript.

Sisselogimisvormi paroolivälja ei olnud keelatud selle automaatne täitmine. See võimaldab juhul, kui kasutaja paroolid on brauseris salvestatud, rünnatava arvutisse ligipääsu omamisel ka teemaks oleva veebirakendusse tema kasutajakontoga sisse logida.

Parooli tugevusnõuded on miinimumilähedased. Ainuke tingimus on, et parooli pikkus peab olema 5 märki. Inglise keelse tähestiku puhul on näiteks erinevaid tähti 26 ja seega on erinevaid võimalusi 26^5 = 11881376, mille murdmiseks jõurünnel kulub keskmisest odavamal videokaardi protsessoril aega alla sekundi [14].

Rakenduse seadetes ei ole määratud ajapiiri, millal parool enam ei kehti. See annab ründajale aega parooli lahti murda ilma, et kasutajalt nõutaks selle muutmist.

Sessiooniküpsises sisalduv info ei kaota kehtivust pärast väljalogimist. Küpsis küll kustutatakse ja uuesti sisse logides genereeritakse uus, aga vana jääb toimima. Kohtvõrgus küpsiste väärtuste edastamise liiklust pealt kuulamine ja enda brauserisse teise kasutaja

infot sisaldava küpsise loomine on väga lihtne ja võimaldab hõlpsalt teise kasutaja sessioon üle võtta ja pahatahtlikke tegusid korda saata.

Kasutaja sessioonil ei ole aegumistähtaega, sessioon lõpeb väljalogimisel või brauseri sulgemisel. Sessioon peaks aktiivsuse puudumisel mingi ajavahemiku möödudes aeguma, takistades arvuti järelvalveta jätmisel ründaja ligipääsu rakendusele.

Chivaleti hinnaväljal on keelatud kõik nupuvajutused peale numbrite. Tegu ei ole küll turvaveaga, kuid väärib mainimist kasutajakogemuse halvamise pärast. Sisestades kogemata mõne numbri valesti, ei ole võimalik seda enam muuta.

XSS turvaauk avastati *meta* märgendisse andmete kuvamisel, kus HTML märke ei konverteeritud. Sisestades soovinimekirja või toote pealkirja välja märgendi lõpumärgise ja selle järel skriptimärgendi (`"><script>alert(0);</script>`), käivitub skript nimekirja või toodet kuvaval lehel. Nii võib olla võimalik näiteks ohvri sessiooni kohta andmeid koguda (mitte küll konkreetsel juhul, kuna sessiooni infot hoitakse küpsistes, millel on määratud *HTTPOnly* lipp) ning lehti kuvavaid kõrgemate õigustega kasutajaid rünnata, muuta rünnatava lehe sisu, suunata rünnataval lehel postitatud vormide andmed teisele lehele, levitada viiruseid, teostada CSRF ründeid [19].

Pärast mistahes arvu keelatud operatsioonide või reeglitele mittevastavate päringute sooritamist ei keelatud mingis olukorras kasutaja ligipääs rakendusele. See annab võimaluse ründajal proovida kuitahes palju erinevaid tehnikaid rakenduse murdmiseks ükskõik, millise ajaperioodi jooksul.

Ei sessiooniküpsisel ega ka „jäta mind meelde“ küpsisel ei olnud määratud *secure* lippu, kuna HTTPS turvaühendus ei ole rakenduse kasutamisel peale sunnitud mujal, kui sisselogimisel. Seega on võimalik hõlpsalt ohvri internetiliiklust pealt kuulata, näiteks küpsiste andmed sealt välja filtreerida ja endale salvestada.

Rakendus on haavatav BEAST ründega, mis kasutab ära šifribloki jada nõrkust võimaldades sooritada MITM (*man-in-the-middle*) rünne kuulamaks pealt TLS turvakanalil liikuvaid andmeid, nagu näiteks autentimistunnused.

Puuduvad kaitsemeetmed CSRF rünnete vastu, kuna andmebaasi kirje lisamise päringu sooritaja pärinemist ei kontrollita. Seega on võimalik ründajal peita oma pahatahtlikule veebilehele nähtamatu vorm, mis postitatakse kasutaja teadmata ning kuna tema brauseris on olemas chivalo.com sessiooniküpsis, toimub päring tema nimel.

Enamik välja toodud turvanõrkustest on tekkinud arenduse käigus ja ei ole osa kasutatava raamistiku või andmebaasimootori vigadest.

4. Nõrkuste likvideerimise kava

Järgnevalt pakub autor välja võimalused eelnevas peatükis (vt. alapunkt 3.12, lk 28) välja toodud nõrkuste likvideerimiseks.

4.1. Mitte kasutust leidvate skriptide eemaldamine

Rakenduse JavaScript kood on vähe struktureeritud ja esineb kordusi. Skriptifailid tuleks üle vaadata ja puhastada ning implementeerida asünkroonne skriptide laadimismehhanism, mis laeb need vastaval lehel vaid juhul, kui vajadus tekib.

4.2. Sisselogimisvormi väljadele automaatse täitmise keelamine

Sisselogimisvormi väljadele lisada atribuut `autocomplete="off"`.

4.3. Parooli tugevuse kontrolli tõhustamine

Antud hetkel peab ainsa tingimusena parooli pikkus olema 5 tähemärki. Inglise keelse tähestiku puhul on erinevaid tähti 26 ja seega on erinevaid võimalusi 26^5 = 11881376. Radeon 5770 graafikakaardi protsessoril kulub sellise parooli ära arvamiseks jõuründel $26^5/(3{,}3*10^9)$ = 0.036 sekundit [14]. Autori soovitus on tõsta minimaalne tähemärkide arv 8-ni ja nõuda ka suurtähtede ning numbrite sisaldamist. Nii on erinevaid võimalusi 62^8 = 218340105584896. Sama graafikakaardi protsessoril kulub nüüd jõuründel parooli

murdmiseks $62^8/(3{,}3*10^9)$ = 18.38 tundi [14], mis on 1837879 korda kauem, kui viie väikese tähemärgi puhul.

Lisaks võib rõhutada kasutajatele Facebooki ja Twitteri kontodega sisselogimist, kuna nendes rakendustes on juba parooli tugevuse kontrollid implementeeritud ja tugevad.

4.4. Parooli aegumise piiri seadmine

Kasutajate paroolidele tuleks seada kehtivus ajaliselt või teatud arvu sisselogimiste järgi, pärast mida nõutakse neilt parooli vahetust. Play! raamistikku selline funktsionaalsus sisse ehitatud ei ole. Kasutajate tabelisse tuleks lisada näiteks parooli aegumise kuupäev ning sisselogimisel kontrollida, kas hetkekuupäev on sellest suurem või väiksem ning sellele vastavalt nõuda paroolivahetust või kasutaja autentida.

4.5. Sessiooni invalideerimine ja kehtivusaja sätestamine

Sessiooni invalideerimise tuge Play! raamistik kahjuks ei paku, kuid ka see on võimalik endal hõlpsasti juurde ehitada. Autor pakub lahendusena genereerida igal sisselogimisel andmebaasi kirje kasutaja sessiooni ning IP kohta. Väljalogimisel ning sessiooni lõppemisel tuleb see märkida kehtetuks ning igal järgneval sessiooni loomisel selle olemasolu ja kehtivust verifitseerides tuvastada, kas on tegu sama kasutajaga. Samuti tuleks sellele kirjele lisama aegumisaeg.

Play! raamistik pakub lisaks sessiooni aegumisaja sätestamiseks `application.conf` failis atribuudi `application.session.maxAge` [17].

4.6. XSS turvaaugu parandamine

Kuna antud juhul on XSS turvaauk tekkinud HTML erimärkide konverteerimise unustamise tõttu ning andmed kuvatakse andmebaasist otse *meta* märgendisse, tuleb selle parandamiseks kuvada vastav muutuja programmikoodis kujule `${@tabel.atribuut}`, mille puhul konverteerib Play! raamistik HTML erimärgid automaatselt [20].

Raamistiku poolt on loodud ka konverteerimise ja varjundamise konfiguratsioonivõimalus `application.conf` faili atribuudi `future.escapeInTemplates=true` näol.

4.7. Ligipääsu keelamine pärast mitmeid vigaseid päringuid

Jõurünnete ära hoidmiseks peaks keelama ligipääsu IP aadressilt, kust on tulnud mingi teatud arv pöördumisi, mis on põhjustanud rakenduse töös vigu. See võib olla märk, et ründaja proovib erinevaid tehnikaid või otsib rakenduses turvaauke.

Antud juhul tuleks salvestada sisselogimiskatsed ning näiteks kolme ebaõnnestunud katse järel blokeerida päringuid sooritanud seadme IP aadress või kasutaja näiteks üheks tunniks.

Packt Publishing poolt avaldatud raamatus „*Play Framework: Data Validation Using Controllers*“ pakutakse jõurünnete ära hoidmiseks sisselogimisel kasutada klassi `UnauthorizedDigest`, mis genereerib räsi igal sisselogimiskatsel kasutajanimest, paroolist ning mingist suvalisest sõnest ja `WWW-Authenticate` päisest [16]. Nii on iga päring unikaalne ja on võimalik jõuründeid ära hoida tuvastades päringu saatja.

4.8. Sundida HTTPS ühendus ning määrata *secure* lipp küpsistele

Internetiliikluse pealtkuulamise (kasutaja sisselogimisandmete ja küpsiste varastamise) takistamiseks peaks kohustuslikus korras sundima kasutajaid rakendusega suhtlema üle HTTPS turvaühenduse. Lisaks tuleks „jäta mind meelde“ ning sessiooniküpsisele määrata *secure* lipp, mis muudab nende varastamise keerulisemaks.

4.9. BEAST ründe ära hoidmine

PhoneFactor andmeil on ainus usaldusväärne viis kaitsta rakendusi BEAST rünnete eest on RC4 šifrikomplektide prioritiseerimine. Pakutakse välja järgnevad atribuudid Apache serveri konfiguratsiooni jaoks [23]:

```
SSLHonorCipherOrder On
SSLCipherSuite RC4-SHA:HIGH:!ADH
```

4.10. Kaitse CSRF rünnete vastu

Takistamaks ründajat postitamaks infot vormidest mujalt kui Chivaletist, tuleks võtta kasutusele moodul `play2-authenticitytoken` [21].

Chivaletis olevale toote lisamisvormile on vaja lisada peidetud väli, mis sisaldab serveri poolel genereeritud unikaalset tunnust. Kui kasutaja postitab vormi, kontrollitakse kasutaja poolt saabunud andmed ning kui need ei sisalda mainitud tunnust, ei võeta andmeid vastu.

Kokkuvõte

Käesoleva diplomitöö eesmärk oli anda ülevaade veebirakenduse chivalo.com turvalisusest, dokumenteerida turvatesti protsess, leitavad tugevused ja nõrkused ning viimaste parandamiseks pakkuda välja lahendused.

Testimisel võeti aluseks standard kolme üldtunnustatud valiku hulgast:

- *Open Web Application Security Project Application Security Verification Standard;*
- *Open Web Application Security Project Software Assurance Maturity Model;*
- *Web Application Security Consortium Threat Classification.*

Valituks osutus OWASP ASVS (*Open Web Application Security Project Application Security Verification Standard*), kuna pakub teemaks oleva rakenduse testimiseks sobiva meetrika ning võimaldab tuvastada nii vigu äriloogikas kui ka tehnilises funktsionaalsuses.

Testimisel rakendati OWASP ASVS tase 2A musta kasti meetodil (manuaalne turvatestimine rakenduse lähtekoodi ja serveri seadistust uurimata) diplomitöö mahupiirangu huvides ning ründaja positsiooni võimalikult täpselt jäljendamiseks.

Turvatesti tulemusena leiti rakenduse funktsionaalsust vastavalt standardis seatud nõuetele kontrollimisel järgmised turvavead:

- lehel tuvastati üleliigseid skripte;

- sisselogimisvormi väljadel on lubatud automaatne täitmine;
- kasutaja parooli tugevusnõuded on miinimumilähedased;
- kasutaja paroolil ei ole kehtivusaega;
- sessiooniküpsises sisalduv info ei kaota serveris kehtivust pärast sessiooni lõppemist;
- sessioonil ei ole aegumisaega;
- HTML lehe `meta` elemendi koostamisel tekib XSS turvaauk;
- kasutaja ligipääsu rakendusele ei keelata pärast mistahes arvu ebaõnnestunud päringute sooritamist;
- rakenduse poolt loodavatel küpsistel ei ole määratud *secure* lipp;
- rakendus on haavatav BEAST rünnetega;
- rakendus ei ole kaitstud CSRF rünnete eest.

Töö tulemi teise osana pakkus autor viimases peatükis mainitud turvanõrkuste parandamiseks lahendused. Diplomitöö teemaks oleva veebirakenduse testimise skoobis olnud funktsionaalsus on autori hinnangul turvaline, kui välja toodud vead parandada.

Enamik mainitud turvanõrkustest on tekkinud arenduse käigus ja ei ole osa kasutatava raamistiku või andmebaasimootori tehnilistest vigadest. Autor julgeb soovitada veebirakenduste arendamiseks Play! raamistikku ja Ebean andmebaasilahendust, kuna palju kaitsemeetmeid on neis vaikimisi sisse ehitatud.

Autorile on avaldatud tänu käesoleva töö koostamise ja praktilise väärtuse eest Bitsnbrains S.L. juhatuse poolt ning tööd on peetud Tallinna Linna IT stipendiumi väärilisek.

Summary

Web Application Security Testing based on the Example of chivalo.com. Diploma thesis. Kermo Pajula

The aim of this thesis was to provide an overview of the security of the web application chivalo.com, to document the process of security testing and strengths and weaknesses found and for latter to suggest fixes.

For the testing a standard was picked from three generally recognised options:

- Open Web Application Security Project Application Security Verification Standard;
- Open Web Application Security Project Software Assurance Maturity Model;
- Web Application Security Consortium Threat Classification.

OWASP ASVS was chosen because it provides appropriate metrics for testing the application as well as it allows to establish flaws in business logic and technical functionality.

OWASP ASVS level 2A black box method (manual security testing without examining the source code of the application or the configuration of the server) was applied in testing concerning the volume limit of the thesis and intention to imitate the position of the attacker as precisely as possible.

As the result of the security testing when checking the application's compliance to verification requirements the following security flaws were found:

- redundant scripts were found in the webpage;
- in the fields of the login form autocomplete is allowed;
- user's password strength requirements are close to the minimum;
- user's password does not have expiry time;
- information in the session cookie is not invalidated in the server after the end of the session;
- session does not have expiry time;
- compiling the `meta` tag in the HTML page produces an XSS vulnerability;
- user's access to the application is not denied after any number of failed requests;
- cookies generated by the application do not have the secure flag set;
- the application is vulnerable to BEAST attacks;
- the application is not protected from CSRF attacks.

As the result of the second part of the thesis the author suggested fixes in the last chapter for the security flaws mentioned. The functionality of the application which was within the scope of the thesis is secure by the author's opinion if the flaws mentioned are fixed.

Most of the mentioned security flaws have been emerged during development and are not part of techical flaws of the framework or database engine used. The author dares to suggest using Play! framework and Ebean database solution for developing web application because they have lots of security measures built in by default.

The author has been rendered thanks by the board of Bitsnbrains S.L. for conducting and the practical value of the thesis which has also been considered worthy of the IT scholarship of Tallinn City.

Kasutatud materjal

1. OWASP (14. märts 2013) OWASP [https://www.owasp.org/index.php/Main_Page]
2. OWASP (14. märts 2013) Projects. [https://www.owasp.org/index.php/About_OWASP#Projects]
3. OWASP (14. märts 2013) OWASP Application Security Verification Standard Project. [https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project]
4. OWASP (15. märts 2013) Verification. [https://code.google.com/p/owasp-asvs/wiki/Verification]
5. OWASP (15. märts 2013) Software Assurance Maturity Model. [https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model]
6. WASC (18. märts 2013) Threat Classification. [http://projects.webappsec.org/w/page/13246978/Threat%20Classification]
7. WASC (18. märts 2013) Using the Threat Classification. [http://projects.webappsec.org/w/page/13246982/Using%20the%20Threat%20Classification]
8. WASC (18. märts 2013) About us. [http://www.webappsec.org/aboutus.shtml]
9. Chivalo.com (02. aprill 2013) Instrucciones. [http://chivalo.com/help]

10. Google Groups (03. aprill 2013) „What about OWASP Top 10“. [https://groups.google.com/forum/?fromgroups=#!topic/play-framework/1lUCi9JWifg]
11. Stack Overflow (03. aprill 2013) „playframework owasp top 10“. [http://stackoverflow.com/questions/6382900/playframework-owasp-top-10]
12. Google (13. aprill 2013) „Chrome DevTools“. [https://developers.google.com/chrome-developer-tools/]
13. W3Schools (16. aprill 2013) „HTML ASCII Reference“ [http://www.w3schools.com/tags/ref_ascii.asp]
14. PC Pro (23. aprill 2013) „How a cheap graphics card could crack your password in under a second“ [http://www.pcpro.co.uk/blogs/2011/06/01/how-a-cheap-graphics-card-could-crack-your-password-in-under-a-second/]
15. Google Groups (23. aprill 2013) „Is the sessioon cookie hackable?“ [https://groups.google.com/forum/?fromgroups=#!topic/play-framework/VZKIauK43J0]
16. Packt Publishing (03. mai 2013) „Play Framework: Data Validation Using Controllers“ [http://www.packtpub.com/article/play-framework-data-validation-controllers]
17. Play! Documentation (07. mai 2013) „Configuration parameters“ [http://www.playframework.com/documentation/1.2.3/configuration]
18. Government Security (13. mai 2013) „SQL Injection Strings!“ [http://www.governmentsecurity.org/forum/topic/7794-sql-injection-strings/]
19. Acunetix (13. mai 2013) „What is Cross Site Scripting and How Can You Fix it?“ [http://www.acunetix.com/websitesecurity/cross-site-scripting/]
20. Stack Overflow (14. mai 2013) „Guide to Proper Escaping in Play Framework“ [http://stackoverflow.com/questions/5764679/guide-to-proper-escaping-in-play-framework]
21. Stack Overflow (14. mai 2013) „How to Prevent CSRF in Play [2.0] Using Scala“ [http://stackoverflow.com/questions/9936563/how-to-prevent-csrf-in-play-2-0-using-scala]

22. Qualys SSL Labs (15. mai 2013) „SSL Server Test / chivalo.com“ [https://www.ssllabs.com/ssltest/analyze.html?d=chivalo.com&hideResults=on]
23. Qualys Community (15. mai 2013) „Mitigating the BEAST attack on TLS“ [https://community.qualys.com/blogs/securitylabs/2011/10/17/mitigating-the-beast-attack-on-tls]

Lisad

Lisa 1 – ASVS nõuetele vastavus

Tabel 1 annab ülevaate rakenduse vastavusest OWASP ASVS nõuetega [4].

✔ Korras
✘ Esineb vigu
▬ Ei saanud testida

Tabel 1
ASVS nõuetele vastavus

V1 - Security Architecture Documentation Requirements	
✘	Verify that all application components (either individual or groups of source files, libraries, and/or executables) that are present in the application are identified.
✘	Verify that all components that are not part of the application but that the application relies on to operate are identified.
▬	Verify that a high-level architecture for the application has been defined.
V2 - Authentication Verification Requirements	
✔	Verify that all pages and resources require authentication except those specifically intended to be public.
✘	Verify that all password fields do not echo the user's password when it is entered, and that password fields (or the forms that contain them) have autocomplete disabled.

Tabel 1 jätkub

Tabeli 1 järg

✗	Verify that if a maximum number of authentication attempts is exceeded, the account is locked for a period of time long enough to deter brute force attacks.
✗	Verify that all authentication controls are enforced on the server side.
✓	Verify that all authentication controls fail securely.
✗	Verify that the strength of any authentication credentials are sufficient to withstand attacks that are typical of the threats in the deployed environment.
✓	Verify that all account management functions are at least as resistant to attack as the primary authentication mechanism.
✓	Verify that users can safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.
✓	Verify that re-authentication is required before any application-specific sensitive operations are permitted.
✗	Verify that after an administratively-configurable period of time, authentication credentials expire.
V3 - Session Management Verification Requirements	
–	Verify that the framework's default session management control implementation is used by the application
✗	Verify that sessions are invalidated when the user logs out.
✗	Verify that sessions timeout after a specified period of inactivity.
✓	Verify that all pages that require authentication to access them have logout links.
✓	Verify that the session id is changed on login.
✓	Verify that the session id is changed on reauthentication.
✓	Verify that the session id is changed or cleared on logout.
✓	Verify that only session ids generated by the application framework are recognized as valid by the application.
V4 - Access Control Verification Requirements	
✓	Verify that users can only access protected functions for which they possess specific authorization.
✓	Verify that users can only access URLs for which they possess specific authorization.
✓	Verify that users can only access data files for which they possess specific authorization.
✓	Verify that direct object references are protected, such that only authorized objects are accessible to each user.
✓	Verify that directory browsing is disabled unless deliberately desired.

Tabel 1 jätkub

Tabeli 1 järg

✔	Verify that users can only access services for which they possess specific authorization.
✔	Verify that users can only access data for which they possess specific authorization.
✔	Verify that access controls fail securely.
▬	Verify that the same access control rules implied by the presentation layer are enforced on the server side.
✔	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.
✔	Verify that all access controls are enforced on the server side.
✘	Verify that limitations on input and access imposed by the business on the application (such as daily transaction limits or sequencing of tasks) cannot be bypassed.
V5 - Input Validation Verification Requirements	
✔	Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.
▬	Verify that a positive validation pattern is defined and applied to all input.
✘	Verify that all input validation failures result in input rejection or input sanitization.
✔	Verify that a character set, such as UTF-8, is specified for all sources of input.
✔	Verify that all input validation is performed on the server side.
V6 - Output Encoding/Escaping Verification Requirements	
✘	Verify that all untrusted data that are output to HTML (including HTML elements, HTML attributes, javascript data values, CSS blocks, and URI attributes) are properly escaped for the applicable context.
▬	Verify that all output encoding/escaping controls are implemented on the server side.
V7 - Cryptography Verification Requirements	
▬	Verify that all cryptographic functions used to protect secrets from the application user are implemented server side.
▬	Verify that all cryptographic modules fail securely.
V8 - Error Handling and Logging Verification Requirements	
✔	Verify that that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id and personal information.
✔	Verify that all server side errors are handled on the server.
▬	Verify that all logging controls are implemented on the server.
✔	Verify that error handling logic in security controls denies access by default.

Tabel 1 jätkub

V9 - Data Protection Verification Requirements	
✔	Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features.
✔	Verify that all sensitive data is sent to the server in the HTTP message body (i.e., URL parameters are never used to send sensitive data).
V10 - Communication Security Verification Requirements	
✔	Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid.
✔	Verify that failed TLS connections do not fall back to an insecure connection.
V11 - HTTP Security Verification Requirements	
✔	Verify that redirects do not include unvalidated data.
✔	Verify that the application accepts only a defined set of HTTP request methods, such as GET and POST.
✘	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).
✔	Verify that the HTTPOnly flag is used on all cookies that do not specifically require access from JavaScript.
✘	Verify that the secure flag is used on all cookies that contain sensitive data, including the session cookie.
✔	Verify that HTTP headers in both requests and responses contain only printable ASCII characters.

Lisa 2 – Sõnede nimekiri *SQL injection*-i jaoks

Government Security foorumi kasutajate poolt koostatud sõnede nimekiri [18].

```
admin'--
' or 1=1--
'" or 1=1--
administrator'--
superuser'--
test'--
' or 0=0 --
' or 0=0 --'
' or 0=0 #
" or 0=0 --
" or 0=0 --'
'" or 0=0 --
or 0=0 --
' or 0=0 #
" or 0=0 #
or 0=0 #
' or 'x'='x
" or "x"="x
') or ('x'='x
" or 1=1--
or 1=1--
' or a=a--'
' or a=a #
' or a=a--
' or "a"="a
' or 'a'='a
" or "a"="a
') or ('a'='a
") or ("a"="a
hi" or "a"="a
hi" or 1=1 --
hi' or 1=1 --
hi' or 'a'='a
hi') or ('a'='a
hi") or ("a"="a
' or 1=1--
" or 1=1--
or 1=1--
" or "a"="a
') or ('a'='a
```