TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Uku Sõrmus 222455IAIB

Veebiküpsistega seotud ründeskoopide visualiseerimine

Bakalaureusetöö

Juhendaja: Sten Mäses

PhD

Kaasjuhendaja: Elar Lang

MSc

Tallinn 2025

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: Uku Sõrmus

04.06.2025

Annotatsioon

Töös uuriti veebiküpsistega seotud ründetehnikate skoopide muutumist vastavalt kahele tegurile: 1) küpsise seaded (atribuudid, nime eesliited) ja 2) sihtmärkrakenduse ning ründaja kontrolli all oleva rakenduse URL'id. Teema süsteemne mõistmine on eriti oluline turvatestijatele, et kergemini märgata ründajale ärakasutatavaid vigu ja anda tõenduspõhiseid soovitusi arendajatele veebirakenduste turvalisuse tõstmiseks.

Varasemate tööde analüüsi põhjal tuvastati, et need pakuvad samade nimedega ründetehnikatele vastukäivaid definitsioone ja on staatilise iseloomuga, mis raskendab lugejatel tehnikate mõistmist ning kohandamist enda olukorrale. Leiti, et parem viis võiks olla selle informatsiooni edasi andmine dünaamiliselt, vastavalt kasutaja sisendile.

Peale probleemi tuvastamist püstitati täpsemad lahenduse eesmärgid ja valiti rakenduse arhitektuur ning tehniline pinu. Rakendust arendati iteratiivselt, katsetades erinevaid viise info edasiandmiseks ja kogudes jooksvalt tagasisidet. Küpsistega tehtavad toimingud loetleti, jaotati sobivatesse kategooriatesse ning töötati välja brauserite käitumist jäljendav mudel, mis toimingute skoope vastavalt sisendile kuvaks. Tulemuste valideerimiseks viidi läbi ekspertintervjuud elukutseliste turvatestijatega.

Intervjuud näitasid, et loodud tööriist on kasulik ja lihtsustab küpsistega seotud ründesituatsioonide mõistmist. Lisaks tuvastati tööriista arendamise jooksul ka küpsistega seotud turvaprobleeme välistest osapooltest nagu Mozilla Firefox brauser.

Töös loetleti ka piirangud nagu piirdumine kahe brauseri käsitsi testimisega ja edasised arenguvõimalused nagu arvestamine olukorraga, kus veebileht kaasab enda dokumendi sisse kolmandaid osapooli.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 5 peatükki, 18 joonist, 2 tabelit.

Abstract
Visualizing the Scopes of Cookie-Based Attacks

This thesis investigates the changes in scope of cookie-based attacks, based on two factors: 1) settings of a cookie (attributes, name prefixes) and 2) the URLs of the target and attacker controlled applications. It is especially important for web application penetration testers to have a systemic overview of this topic, to have an easier time spotting exploitable vulnerabilities and give evidence-based recommendations to developers to increase the security of their web apps.

The analysis of previous work suggests that attack techniques with same names are defined inconsistently and the works are mostly static in nature, which makes it harder for the readers to apply the techniques to their unique situations. It was found that a better approach might be to present this information dynamically, based on user input.

After recognizing the problem, more specific goals were set for a solution, and the architecture and technical stack of the application were chosen. The tool was developed iteratively, experimenting with different ways to present the information and gather feedback on the run. Actions that could be taken with cookies were enumerated and categorized, and a model replicating the behavior of web browsers was created to visualize the scopes of the actions. To validate the results, interviews were conducted with professional web application penetration testers.

The interviews showed that the created tool is useful and simplifies understanding cookie-related attack scenarios. While creating the tool, cookie-related security issues were found in external projects like the Mozilla Firefox web browser. The work also listed limitations, such as creating the model based on manual testing of two browsers, and future development possibilities, such as accounting for situations where third-party resources are embedded.

The thesis is in Estonian and contains 33 pages of text, 5 chapters, 18 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	rakendusliides (*Application Programming Interface*)
CSRF	päringuvõltsing (*Cross-Site Request Forgery*)
eTLD	kehtiv tippdomeen (*effective top-level domain*, ka *public suffix*)
eTLD+1	registreeritav domeen (*effective top-level domain + 1*, ka *registrable domain*)
HTML	hüperteksti märgenduskeel (*HyperText Markup Language*)
HTTP	hüperteksti edastuse protokoll (*HyperText Transfer Protocol*)
HTTPS	turvaline hüperteksti edastuse protokoll (*HyperText Transfer Protocol Secure*)
JS	JavaScript
UI	kasutajaliides (*User Interface*)
URL	veebiaadress (*Uniform Resource Locator*)
origin	URL komponentide kolmik (skeem, host, port), mis on aluseks brauserite põhilisele turvapoliitikale (vt SOP); lähim eestikeelne vaste on "allikas", töös on selguse huvides kasutusel `origin`
PSL	avalik nimekiri kehtivatest tippdomeenidest (*Public Suffix List*)
site	sait, ajalooliselt eTLD+1 (selle puudumisel host), brauserites küpsiste jaoks kasutatava eraldusmehhanismi alus; töös räägitakse saidist selle termini raames, üldisema mõistena kasutatakse terminit "veebileht"
SOP	brauserite põhiline turvapoliitika veebilehekülgede eraldamiseks (*Same-Origin Policy*); eestikeelne lähim vaste "ühisallika poliitika"
TLD	tippdomeen, üladomeen (*top-level domain*)
TLS	transpordikihi turve (*Transport Layer Security*)
WHATWG	töörühm HTML-i ja seotud tehnoloogiate arendamiseks (*Web Hypertext Application Technology Working Group*)
XSS	murdskriptimine, skriptisüst (*Cross-Site Scripting*)

Sisukord

1 Sissejuhatus 11
1.1 Taust 11
1.2 Probleem 13
1.3 Eesmärk 14
1.4 Ülevaade tööst 14
2 Metoodika 16
2.1 Probleem ja olulisus 16
2.1.1 Varasemate tööde ülevaade 16
2.1.2 Probleemid varasemate töödega 18
2.2 Lahenduse eesmärkide seadmine 21
2.3 Disain ja arendus 22
2.3.1 Arhitektuur ja tehniline pinu 22
2.3.2 Prototüüpimine ja iteratiivne arendus 24
2.3.3 Mudeli loogika 25
3 Tulemused 28
3.1 Tehise demonstratsioon probleemi lahendamiseks 28
3.2 Võrdlus varasemate töödega 35
3.3 Tööriista valideerimine turvatestijate seas 37
3.3.1 Intervjueerimise protsess 37
3.3.2 Intervjuude tulemused 38
4 Diskussioon 40
4.1 Järeldused tulemustest 40
4.2 Töö piirangud 40
4.3 Võimalused edasiseks uurimiseks ja arenduseks 41
5 Kokkuvõte 43
Kasutatud kirjandus 44

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 47
Lisa 2 – Näide TypeScript'i kasutamisest küpsise seadete esitamiseks 48
Lisa 3 – Töö käigus tuvastatud vead välistes osapooltes 49
Firefox ja Chromium 49
Haridus- ja Teadusministeerium 50
Vue playground 51
Muud ettepanekud 53

Jooniste loetelu

Joonis 1. Lihtsustatud näide küpsiste mehhanismi toimimisest jadaskeemina. 11

Joonis 2. Menüütasemel ülevaade algselt plaanitud keskkonnast. 19

Joonis 3. Tööriista arhitektuur (sisendid ja väljundid). 23

Joonis 4. AI abil genereeritud prototüüp algsest tööriistast. 24

Joonis 5. Iteratsioon tööriistast: URL-sisenditest genereeritud tabelid. 25

Joonis 6. Demoversioon küpsiste skoope interaktiivselt näitavast tööriistast. 27

Joonis 7. Sisendid: sihtmärkrakenduse URL ja ründaja kontroll all olev URL. 29

Joonis 8. Sisendid: näide ebasobiva sisendi korral kuvatavast veateatest. 29

Joonis 9. Väljundid: sisend-URL'id oluliste komponentide visualiseerimine. 30

Joonis 10. Väljundid: dünaamiline tabel `site` (skeemita ja skeemiga) ning `origin` skoopide kuvamiseks (vastavalt *schemelessly same-site*, *schemefully same-site* ja *same-origin*). Kui skoop sihtmärgi ja ründaja vahel kattub, kuvatakse mõlema tulba peale üks ühine väärtus. 30

Joonis 11. Väljundid: dünaamiline tabel `origin` ja `site` skoopide kuvamiseks kahe äärejuhtumi korral. 31

Joonis 12. Sisendid: küpsise sätted tööriista vaikeolekus. 31

Joonis 13. Sisendid: küpsise sätted, sihtmärkrakenduse URL'i äärejuhtumite korral. .. 32

Joonis 14. Väljundid: millal on võimalik sihtmärkrakenduse küpsiseid mõjutada. 33

Joonis 15. Väljundid: millal pannakse küpsis HTTP päringule kaasa, sõltuvalt päringu tüübist ja päringut initseeriva lehekülje suhtest sihtmärkrakendusega. .. 33

Joonis 16. Väljundid: millal on võimalik sihtmärkrakenduse küpsiseid mõjutada, küpsise nime `__Host-` eesliite korral. 34

Joonis 17. Firefox toob aadressiribal registreeritava domeeni eraldi esile: (a) `pg.edu.ee` puhul on selleks `pg.edu.ee`, (b) `www.voru.edu.ee` puhul `voru.edu.ee`, (c) `vjk.vil.ee` puhul `vil.ee`, (d) `abjag.vil.ee` puhul samuti `vil.ee`. 50

Joonis 18. Brauseris avanev vaade pärast seda, kui ohver on ründaja lingi avanud:
(a) `blog.vuejs.org`, (b) `vuejs.org` külastamisel..... 53

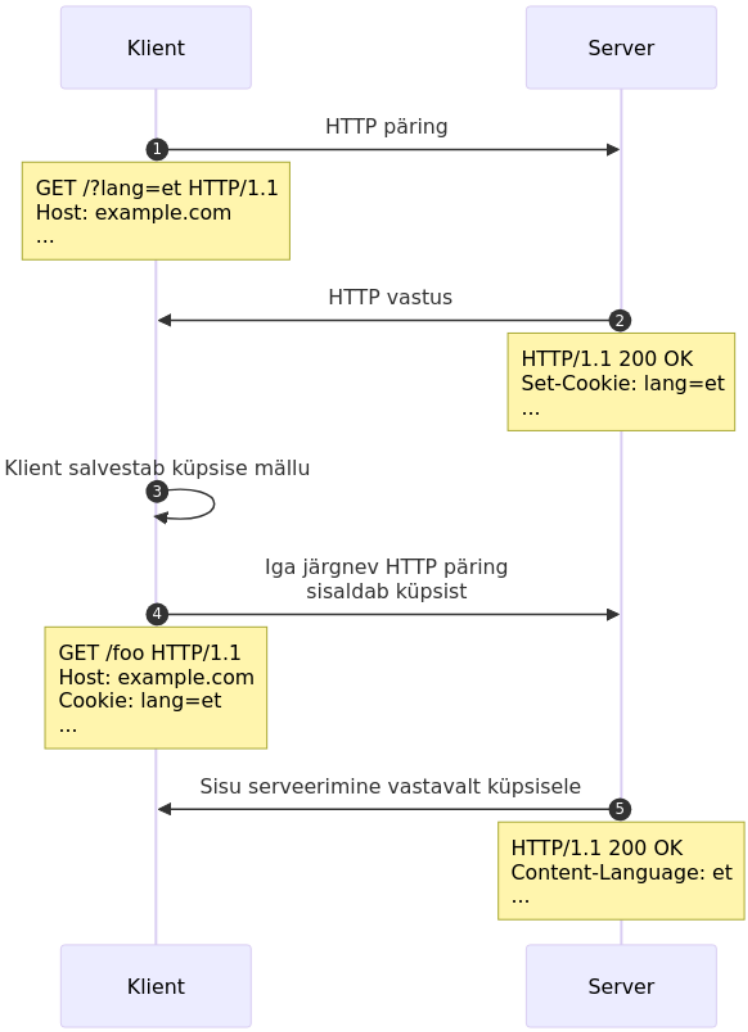
Tabelite loetelu

Tabel 1. Võrdlus varasemate tööriistadega küpsiste käitumise katsetamiseks. 35
Tabel 2. Võrdlus varasema tööriistaga URL-i komponentide visualiseerimiseks..... 36

1 Sissejuhatus

1.1 Taust

HTTP on oma olemuselt olekuvaba protokoll [1] – kui mõnele HTTP serverile saabub kaks HTTP päringut, ei saa see server HTTP tasemel tuvastada, kas päringud tulid samalt HTTP kliendilt või mitte. Selleks, et tekitada veebirakendustele kasutajapõhine "mälu", leiutas 1994. aastal Netscape Navigator veebibrauseri kallal töötanud Lou Montulli veebiküpsised [2], [3]. Nii on võimalik serveritel päringute põhjal kasutajaid ja nende eelistusi eristada, näiteks ostukorvi või keelevaliku salvestamiseks lehekülgedel navigeerimise vahel.



Joonis 1. Lihtsustatud näide küpsiste mehhanismi toimimisest jadaskeemina.

Küpsised (ingl *HTTP cookies*, *web cookies*) on HTTP laiend, mis lubab salvestada serveritel klientide[1] poolele lihtsaid võti-väärtus paare [4]. Joonisel 1 on välja toodud lihtsustatud jadaskeem küpsiste toimimisest:

1. Klient teeb serverile HTTP päringu.
2. Server vastab päringule, `Set-Cookie` vastuspäises on kaasas küpsis, mille nimi on `lang` ja väärtus `et`.
3. Klient salvestab küpsise seadme mällu.
4. Klient lisab kõikidele järgnevatele sama lehekülje päringutele küpsise `Cookie` päringupäises uuesti kaasa.
5. Server saab serveerida sisu vastavalt eelistustele või kasutaja identiteedile.

Tegelikkus sisaldab aga hulganisti nüansse [4], mis saavad oluliseks ka turvaküsimustes (küpsiste konfidentsiaalsus, terviklus, veebirakenduse käideldavus kasutaja jaoks):

- Küpsiseid ei saa kirjutada, lugeda ja kustutada ainult server, vaid ka brauseripoolsed skriptid.
- Küpsise kirjutamisel saab kaasa anda mitmeid atribuute, mis määravad muuhulgas:
  - küpsise eluea (`Max-Age`, `Expires`);
  - skoobi, milliste URL-ide pärimisel peaks brauser küpsise kaasa panema (`Path`, `Domain`);
  - millistelt teistelt URL-idelt eelneva skoobi pihta päringuid tehes peaks brauser küpsise kaasa panema (`SameSite`);
  - kas brauseripoolsed skriptid peaksid saama vastavale küpsisele ligi (`HttpOnly`);
  - kas küpsise seadmisel ja selle lugemisel/kustutamisel/ülekirjutamisel peab olema tegemist turvalise (HTTPS) ühendusega (`Secure`).
- Seadmisel kasutatud atribuute ei saadeta aga tagasi serverile: kaasa pannakse vaid küpsiste nimi-väärtus paarid.
- Küpsise nimele teatud eesliiteid (`__Host-` või `__Secure-`) lisades nõuab brauser vastavate atribuutide olemasolu või puudumist.

[1] Kliendi all mõeldakse siin ja edaspidi graafilisi veebibrausereid (nt Google Chrome, Mozilla Firefox), kui pole märgitud teisiti. Küpsiste mehhanismi võivad kasutada ka muud kliendid peale veebibrauserite, nagu näiteks käsurea tööriist curl [5]. Siin töös on fookus küpsistega seotud rünnetel läbi ohvri brauseri ehk rünne eeldab ohvrilt ründaja kontrolli all oleva lehe külastamist.

- Võimalik on luua mitu (mh sama nimega) küpsist.

Küpsised, nagu ka muud moodsamad kliendipoolsed olekusalvestamise mehhanismid (nt WebStorage API alla kuuluvad LocalStorage ja SessionStorage [6]), on turvaperspektiivist huvitav uurimisobjekt, sest hoiavad tihti seansivõtit ([4], ptk 8.4). See võib lekkimise korral anda ligipääsu ohvri kontole. Võrreldes moodsamate alternatiividega on küpsistel aga kaks olulist ajaloolist erinevust (turvaperspektiivist nõrkust):

1. Küpsised on brauserite poolt kasutatav peamine **kaasõiguse** (ingl *ambient authority*) vorm[2]. Brauser paneb HTTP päringutele küpsised kaasa ilma, et arendaja seda koodis sõnaselgelt välja kirjutaks [3].
2. Küpsised ei järgi põhilist brauserite turvapolititikat erinevate veebilehekülgede eristamiseks – *Same-Origin Policy* (SOP) reeglistikku, mis eristab veebilehekülgi `origin` järgi (URL skeem, host, port) –, vaid lõdvemat **saidi** (ingl `site`) mõistet, milleks on *üldjuhul* registreeritav domeen nagu näiteks `taltech.ee` või `example.co.uk` [7].

Nendest kahest põhjusest juhinduvalt tulenevad ka küpsistele omased ründed. Kaasõiguste tõttu on näiteks võimalik päringuvõltsing (CSRF, *Cross-Site Request Forgery*) ([4], ptk 8.1). Saidi turvapoliitika järgimise tõttu on võimalik omavaheline mõju näiteks URL-ide `http://1.dev.example.com:8080/` ja `https://bank.example.com/` vahel, mis omavad ühist saiti `example.com` – kuigi tegemist võib olla täiesti erinevate veebirakendustega, millel on ka erinev turvatase.

1.2 Probleem

Küpsistega seotud ründetehnikate nimekirjades ja kirjelduste juures puudub tihti selgus, **mis kontekstis rünne täpselt võimalik on** ehk kus URL-il peab ründaja sihtmärkrakenduse suhtes asuma; kui seda on ka mitme lausega selgitatud, ei pruugi see olla kiirelt ja intuitiivselt kohandatav lugeja olukorrale.

Tänu sellele, et küpsiste võimalikke sätete kombinatsioone on võrdlemisi palju (erinevad

[2]Teised brauserite seas kasutatavad kaasõiguste vormid on TLS kliendisertifikaadid ja `HTTP Authentication` (näiteks `"Basic authentication"`) [8], [9], [10]. Väljaspool TCP/IP-mudeli rakenduskihti saab taustautoriteedi alla lugeda ka kliendi IP-aadressi, mille järgi võib server saada kliente eristada.

atribuudid, nime eesliited), võib lisaks olla keeruline mõista, **kuidas erinevad küpsiste seaded rünnete väljavaateid mõjutavad**.

Olukorda teeb veel vähem hoomatavamaks tõsiasi, et kuigi küpsiste mehhanism ise on üle 30 aasta vana, on siiani aktiivses muutumises nii küpsiste standard, võimalikud atribuudid, vaikeseaded, kui ka saidi mõiste (Chromium'i-põhistes brauserites ja ainult teatud kontekstides URL skeemi lisandumine) [4], [11], [12]. Ka veel 2025. aastal on avastatud uusi ründetehnikad, mis spetsifikatsioonide ja implementatsioonide rägastikust leitud nõrkusi ära kasutavad [13].

Probleem on oluline, sest kõige selle keerukuse tõttu võivad nii turvatestijatel kui arendajatel jääda kahe silma vahele olukorrad, mida pahatahtlikud ründajad ära kasutavad, ja jääda ebamääraseks, kuidas kaitsemeetmete (näiteks küpsiste eesliidete) rakendamisel erinevad ründeskoobid muutuvad, mistõttu ei pruugita mõista kaitsemeetmete olulisust.

1.3 Eesmärk

Töö eesmärgiks on luua avatud lähtekoodiga **interaktiivne tööriist**, mis aitab aru saada, kuidas saab ründaja kontrolli all olev veebilehekülg mõjutada ette antud sihtmärkrakenduse veebilehekülge ja selle küpsiseid, kui ohver ründaja kontrolli all olevat lehekülge külastab ning kuidas erinevad küpsiste seaded ründeskoobi suurust mõjutavad.

Töö tulem peaks olema eelkõige mõeldud abistama turvatestijaid veebirakenduste läbistustestidel ründevõimaluste mõistmiseks ja vastama küsimustele nagu *"kui olen leidnud vea leheküljel* `x.site.example`*, kuidas saan mõjutada lehekülge* `y.site.example`*, kui see kasutab küpsist sätetega* `z`*?"*. Vajadusel peaks olema võimalik tööriista väljundile viidata, et lisada see abistava materjalina turvatesti raportisse või veebiturvakoolituse slaididele.

1.4 Ülevaade tööst

Järgnevas peatükis 2 on lahti selgitatud töö metoodika: varasemate tööde kaardistamine ja näidete abil probleemi esiletõstmine, täpsemad oodatava lahenduse eesmärgid ja nõuded, tehnoloogilise pinu valik ning protsess, kuidas on tööriist valminud.

Peatükis 3 on kirjeldatud töö tulemusi ja viidud töö tulemuste valideerimiseks läbi

kvalitatiivne uuring Clarified Security OÜ veebiturvatestijate seas.

Peatükis 4 on toodud välja järeldused tööst ja võimalused edasiseks arenduseks ning uurimiseks. Töö võtab kokku peatükk 5.

Lisaks on loetletud mitmeid töö käigus tuvastatud (turva)probleeme väliste osapoolte seas, mis on kirjeldatud eraldi lisas 3.

2 Metoodika

Tegemist on oma olemuselt tööga, mis koondab olemasolevat informatsiooni ja esitab seda uuel viisil, seega on metoodika aluseks valitud 2006. aastal Peffers'i ja teiste poolt kirjeldatud sammud disainiteaduse uurimisprotsessi jaoks [14]:

1. probleemi identifitseerimine ja olulisuse näitamine;
2. lahenduse eesmärkide seadmine;
3. disain ja arendus (tehise loomine);
4. tehise demonstratsioon probleemi lahendamiseks;
5. tulemuse tõhususe hindamine, itereerimine;
6. tulemuste kommunikeerimine.

Alapeatükis 2.1 antakse ülevaade olemasolevatest allikatest ja nende liikidest ning tuuakse välja neid läbiv ühtne probleem. Probleemile vastavalt seatakse lahenduse eesmärgid (alapeatükk 2.2), mille peale disainitakse ja arendatakse tööriist (uurimuse tehis).

Tulemusi, koos tehise demonstratsiooni ja tõhususe hindamisega, on kirjeldatud lähemalt järgmises peatükis 3. Tulemuste avalik kommunikeerimine algab eelkõige käesoleva uurimistöö avaldamisega.

2.1 Probleem ja olulisus

2.1.1 Varasemate tööde ülevaade

Senised küpsistega seotud ründetehnikaid kirjeldavad tööd võib laias laastus jagada kolmeks:

1. teadustööd akadeemiast – näiteks Squarcina ja teiste poolt 2023. aastal väljastatud *"Cookie Crumbles: Breaking and Fixing Web Session Integrity"* [15] ning Zheng'i ja teiste poolt 2015. aastal väljastatud *"Cookies Lack Integrity: Real-World Implications"* [16];

2. uurimistööd erasektorist ja eraisikutelt – näiteks 2025. aastal küberturbeettevõtte PortSwigger'i blogis Fedotkin'i poolt väljastatud *"Stealing HttpOnly cookies with the cookie sandwich technique"* [13], 2023. aastal Sundara isiklikus blogis avaldatud artikkel *"Cookie Bugs"* [17] ja 2015. aastal filedescriptor pseudonüümi all (olles küberturbeettevõtte Cure53 töötaja) HITCON konverentsil esitatud *"Cookie monster in your browsers"* [18];
3. isiklikud veebipäevikud ja kommuunile muutmiseks avatud küberturbealased teadmuste kogud, mis agregeerivad olemasolevaid tehnikaid eelmisest kahest kategooriast – isikliku blogi näiteks võib tuua Huli *"Beyond XSS"* seeria [19] või Woltjer'i *"Practical CTF"* märkmetekogu [20], millest viimane väidab olema omakorda inspireeritud ühest kuulsaimast üldisest ründetehnikaid koondavast teadmuste kogumist, HackTricks Wiki'st, millel on ka *"Cookies hacking"* peatükk [21].

Lisaks leidub üldiseid, eelkõige arendajatele suunatud ressursse, mis küpsiste käitumist ja `site` ning `origin` skoopide erinevusi (ilma spetsifikatsiooni-tasemel detailidesse laskumata) lahti selgitavad, näiteks:

1. Mozilla Developer Networks (MDN) dokumentatsioon, muuhulgas artiklid küpsiste kohta [22];
2. Google'i töötajate loodud sisu `web.dev` domeenil, muuhulgas interaktiivne tööriist URL-i komponentide ja `site` ning `origin` mõistete visualiseerimiseks aadressil `https://url-parts.glitch.me/` [23], [24].

Küpsiste spetsifikatsiooni võrdlemisel brauserite käitumisega on käesoleva töö jooksul keskendutud kõige värskemale, tulevasele RFC6265bis spetsifikatsioonile, mis on töö hetkel mustandistaatuses ja versiooninumbriga 20 (RFC6265bis-20) [4]. Kui eesmärk oleks leida võimalikke erinevusi küpsiste parsimisel klientide ja serverite vahel ehk ka uusi võimalikke ründetehnikaid, tasuks üle lugeda ka vanemaid spetsifikatsioone.

Muud olulisemad kasutatud spetsifikatsioonid:

- Public Suffix List (PSL) ja selle algoritm [25], mille väljund on vastavalt sisenddomeenile (nt `sub.example.com`) kehtiv tippdomeen (ingl *effective top-level domain* (eTLD), alias *public suffix*), nt `com`, `co.uk`, `travelersinsurance`, `ee`, `edu.ee` või `github.io`, mille alla internetikasutajad saavad või on ajalooliselt saanud domeene

registreerida.

- Elav HTML standard, mis sisaldab saidi definitsiooni (ka skeemita saidi definitsiooni) ja viitab PSL-ile [7].
- Elav URL standard, mis sisaldab `origin` definitsiooni [26], kehtiva tippdomeeni definitsiooni [27] ja registreeritava domeeni (*registrable domain*, alias eTLD+1) definitsiooni [28], viidates ka PSL-ile.

Nii HTML-i kui URL-i elavaid standardeid annab välja WHATWG (töörühm HTML-i ja seotud tehnoloogiate arendamiseks, ingl *Web Hypertext Application Technology Working Group*). PSL on algatatud Mozilla poolt, kuid praegu eestveetud laiema kogukonna poolt. Kui üldiselt on küpsistega seotud info esitletud staatiliselt, leidub ka kaks omavahel sarnast interaktiivset lehekülge, mis aitavad (nende saitide raames) katsetada, millal on küpsised loetavad brauseripoolse JavaScriptiga ja kuidas mõned atribuudid toimivad: setcookie.net [29] ja set-cookie.info [30].

2.1.2 Probleemid varasemate töödega

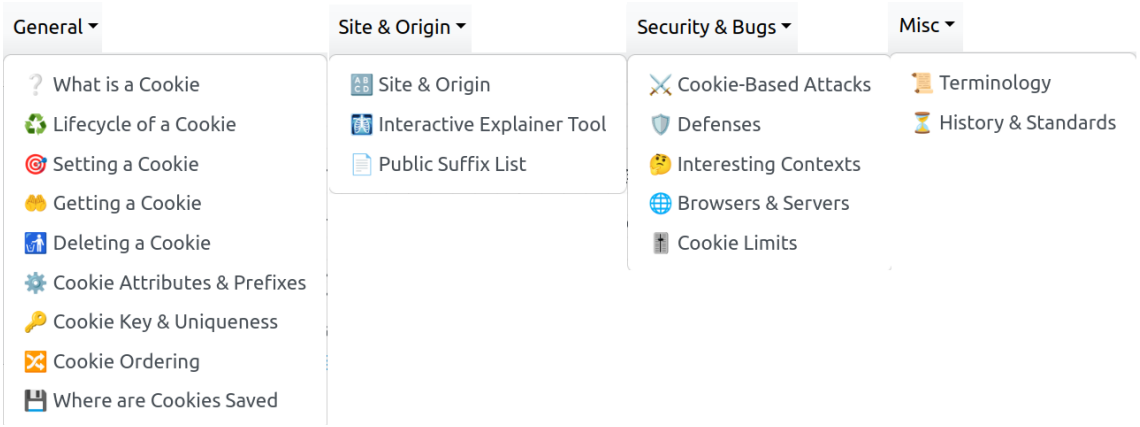
Kõigil küpsistega seotud ründetehnikaid kirjeldavatel tööde kategooriatel on ökosüsteemis oma roll, kuid ka omad unikaalsed plussid ja miinused (vaadatuna turvatestija perspektiivist):

- Teadustööd on oma uuritud valdkonnas täpsed ja pakuvad tihti uudset perspektiivi. Näiteks Squarcina jt poolt nimeta (ingl *nameless*) küpsiste kasutamine `__Host-` ja `__Secure-` küpsise nime eesliidete piirangutest möödapääsemiseks päädis ka muudatustega standardis ([15], ptk 4.2.1). Samas on nende tööde väljund suunatud eelkõige akadeemiale ja spetsifikatsioonide ning implementatsioonide kirjutajatele, mitte otse turvatestijatele või implementatsioone kasutatavatele arendajatele.
- Erasektori ja eraisikute uute tehnikate ja trikkide kirjeldused on tihti lühemad ja seega keskmise tarbija jaoks hoomatavamad ning tihti koos üksikute reaaleluliste näidetega, kus tehnikat on ära kasutatud. Samas ei tule üksikute tehnikate kirjeldused kaasa suurema kontekstiga: näiteks Fedotkin'i kirjeldatud *"cookie sandwich"* tehnika juures ei räägita, mis kontekstis ründaja peab asuma, et vajalikke küpsiseid kirjutada, et soovitud `HttpOnly` atribuudiga küpsist varastada [13].
- Agregeeritud tööd peaksid andma kiire ülevaate, et turvatestijana tunnetada, kas mõni tehnika võib olla antud situatsioonis kasulik. Kaduma läheb aga täpsus ja nüanss ning

tehnikate vahel puudub omavaheline sidusus. Näiteks HackTricks Wiki's on tehnikad ilma näilise suurema süstematiseerimiseta lihtsalt järjestatud ja esineb erinevaid ebakorrektsusi, alates sellest, et mõnes kohas on kasutatud küpsise nime `__Host-` eesliite asemel `__HOST-`, kuigi küpsiste nimed (ja väärtused) on tõstutundlikud [21], [4].

Töö uurimisprotsessi alguspunktis tundus olevat olulisem anda omapoolne panus agregeeritud tööde kategooriasse, luues üldise küberturbe tehnikaid sisaldava veebilehe asemel ühe fokusseeritud projekti, mis keskenduks just küpsistega seotud ründetehnikatele. Selle asemel, et lihtsalt loetleda tehnikaid, võiks neid rangelt süstematiseerida, luua juurde täpsed kontseptsioonitõestused, mille ümber oleks üldisem küpsistega seotud teadmusbaas, mis annaks võimalusele ristviitamisele.

Menüüpunktide tasemel ideed sellisest keskkonnast võib näha jooniselt 2. Teemasse sügavamale süübides ja algse ideega edasi minnes tekkis arusaam, et kuigi ka selline hästi teostatud keskkond omaks palju väärtust, on hetkel suurem probleem hoopis teine, mis on kõikide loetletud kategooriate ülene.



Joonis 2. Menüütasemel ülevaade algselt plaanitud keskkonnast.

Nimelt võib töid lugedes olla keeruline aru saada, mis skoobist on erinevad ründed teostatavad (kus peab ründaja sihtmärkrakenduse suhtes asuma), kuidas need suhestuvad küpsistega sooritatavate baastoimingutega ja kuidas erinevad keskkonna ning küpsiste seaded neid skoope mõjutavad. Kui skoobid ongi täpsemalt välja toodud, tuleb iga kord tegeleda vaimse gümnastikaga, et sobitada need enda olukorrale. Probleem tuleneb eelkõige tööde staatilisest meediumist.

Võtame näiteks *cookie tossing* (toortõlkes "küpsise viskamise") nimetusega tehnika, tuntud ka kui küpsise varjutamine (ingl *cookie shadowing*) ([15], ptk 4.1.1). Sel puudub range definitsioon, kuid üldpildis räägitakse küpsise varjutamisest kui konkureeriva küpsise kirjutamisest, mil on sama nimi, mis juba eksisteerival ("varjutataval") küpsisel, et server kasutaks ründaja valitud küpsise väärtust ja seega mõjutaks küpsise terviklust (ingl *integrity*). Ründe skoop on *tavaolukorras* terve sama skeemita sait (kuid enamasti mitte sama host ehk nõuab `Domain` atribuudi kasutamist ja nii näiteks alamdomeenilt peadomeenile küpsise "viskamist"). Lisaks kasutatakse enamasti `Path` atribuuti, sest brauserid järjestavad spetsiifilisema teekonnaga küpsised ettepoole ja enamik servereid kasutab esimest küpsist, kui `Cookie` päises on mitu sama nimega küpsist. Üks võimalik mõju on näiteks ohvrile ründaja seansi "kinkimine", ilma et ohver seda märkaks ja nii võib ohver sisestada tundlikke andmeid, mis seotakse hoopis ründaja kontoga. Tuues esile kolme erineva kategooria allikad, mis küpsise varjutamise rünnet kirjeldavad:

- HackTricks Wiki's on selle ründe skoobi kohta kirjutatud *"If an attacker can control a subdomain or the domain of a company [. . .]"*, mis ei ütle midagi ründajale vajaliku protokolli (HTTP/HTTPS) kohta, kas alamdomeeni (*subdomain*) asemel sobib sama saidi raames ka sõsardomeen, kuidas küpsise atribuudid (nt `Secure`) mõjutavad ründe võimalusi jne [31].
- Squarcina jt kirjeldavad teadustöös atribuutide mõju täpsemalt (*"[. . .] cookie tossing requires the `https` capability only for cookies with the `Secure` flag [. . .]"* ja *"[. . .] __Host- prefixed cookies are considered to be unaffected by shadowing attacks from a same-site position.'*) ([15], ptk 4.1.1), aga näited on siiski (vastavalt töö meediumi piirangutele) abstraktsel kujul: sihtmärk `https://site.tld/login/index.php`, ründaja `http://atk.site.tld/` ja kindel etteantud küpsis `sid=good; Path=/`. Sõnaselgelt tuuakse välja, et küpsis varjutatakse ehk peaks juba varem sama nimega eksisteerima.
- Houhou *"Cookie tossing attacks"* [32], just sellele ründele pühendatud artikkel eraisikult, räägib huvitavatest tagajärgedest ja rünnetest, millele küpsiste varjutamine võib aluseks olla, kuid räägib vastupidiselt Squarcina tööle ründe variandist, kus sama nimega küpsis varem *ei* eksisteeri. Lisaks ei kasuta autor ründe defineerimisel isegi saidi mõistet, tuues näidetena muuhulgas ebakorrektseid, PSL-i kuuluvaid domeene (nt `herokuapp.com`, `myshopify.com`).

Samad probleemid kehtivad paljude teiste, tihtipeale küsitavate nimetustega tehnikatega kirjeldamisel nagu küpsise pomm (ingl *cookie bomb* – nii suurte küpsiste kirjutamine, et server tagastab veateate ja tekitab brauseripoolse teenusetõkestuse [19]), küpsise võileib (ingl *cookie sandwich* – `HttpOnly` atribuudiga küpsise lugemine teatud teiste haavatavuste ärakasutamisel [13]), küpsisepurgi üleajamine (ingl *cookie jar overflow*, ka *cookie eviction* – nii paljude küpsiste kirjutamine, et brauser hakkab teatud järjekorras küpsiseid kustutama ([15], ptk 4.1.2)) jne. Lõpuks ei ole kõige tähtsam tehnikale antav nimi, vaid oluline konkreetne operatsioon, mida ründaja saab küpsistega sooritada.

Kui puudub parem pilt küpsistega seotud toimingutest ja seotud rünnetest, võib tagajärjeks olla teadmatus ja selle tõttu tehtud või tegemata otsused nii turvatestijate kui arendajate seas, mis mõjutavad negatiivselt veebirakenduste turvalisust. Probleemi lahendamiseks (või vähemalt leevendamiseks) oleks vaja staatilise sisu asemel dünaamilist tööriista, mis looks raamistiku küpsistega seotud rünnetest ja kaitsetest mõtlemiseks, ilma ründetehnikate nimedesse liialt takerdumata. Nii saab rääkida küpsistega seotud rünnetest täpselt, vastavalt olukorrale, ning näidata, kas ja kuidas erinevad seadistused ning kaitsemeetmed ründeskoope mõjutavad.

2.2 Lahenduse eesmärkide seadmine

Probleemi lahendamiseks on kirjeldatud lahti põhimõtted ja eesmärgid, mida loodav tööriist peaks järgima.

Tööriist peaks:

- selguse loomiseks võtma küpsistega seotud operatsioonid algosadeks, ignoreerides esmalt ründeid kui selliseid. Seejärel oleks võimalik algosad taas siduda kokku suuremaks tervikuks (näiteks võimalikke toiminguid kategoriseerides) ja lisada tagasi juurde rohkem ründaja-kaitsja perspektiivi, nagu teadatuntud nimedega ründetehnikad, möödapääsud piirangutest jms;
- olema interaktiivne ja vastu võtma võimalikult palju asjakohast kasutaja sisendit (URL-id, küpsiste seaded), mille põhjal väljundit kuvada, et tekitada kasutajale olukorrapõhine arusaam. Eeldame, et kasutaja enda sisestatud andmete põhjal, olgu selleks reaalsed või fiktiivsed andmed, on kergem väljundit mõista kui autori

väljamõeldud staatiliste näidetega oleks. Erinevate näidete läbiproovimine võiks kaasa aidata ka abstraktsemal tasemel definitsioonide mõistmisele;

- arvestama erinevate äärejuhtumitega, et anda võimalikult terviklik ülevaade. Mõne situatsiooniga mitte arvestades tuleks see sõnaselgelt välja kirjutada, et kasutaja oleks piirangust teadlik;
- viitama võimalusel spetsifikatsioonidele ja tuua välja, kui mõne brauseri käitumine nendest erineb;
- olema kergelt kasutajatele kättesaadav ja avatud lähtekoodiga, et soovi korral kontrollida lahenduse taga peituvat loogikat;
- sobima neile, kel on algteadmised veebirakenduste toimimisest ja turvalisusest, näiteks elukutselised veebirakenduste läbistustestijad;
- omama väljundit, mis on kergelt jagatav (näiteks URL-ina).

Tööriist ei pea:

- olema ilmtingimata mobiilisõbralik, sest veebirakenduste läbistusteste viiakse autori kogemuse põhjal läbi peamiselt suuremate (arvuti)ekraanide taga;
- asendama olemasolevaid töid, vaid neid täiendama.

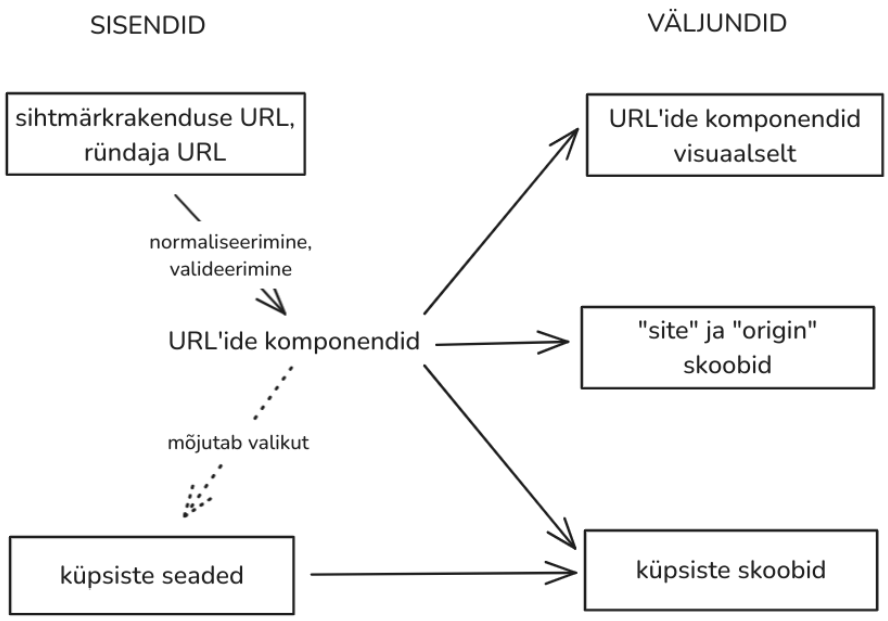
2.3 Disain ja arendus

2.3.1 Arhitektuur ja tehniline pinu

Tööriista osas oli loomulik valik ehitada veebirakendus, sest nii saab tagada, et tulemus on kergelt kasutajatele kättesaadav ja jagatav.

Joonisel 3 on toodud välja suures pildis tööriista arhitektuur sisendite ja väljundite osas. Esimesed kaks väljundit, “URL’ide komponendid visuaalselt” ja “site ja origin skoobid”, on mõeldud vajalike definitsioonide edasiandmiseks. “Küpsiste skoobid” väljund koostatakse sisend-URL’ide ja antud küpsiste seadete põhjal.

Otsustati, et tööriist ei vaja tagarakendust (ingl *backend*), vaid peaks olema vaid staatiliste failide kujul ehk kogu rakendus jookseks vaid kasutaja brauseris. Nii on võimalik täita kergemalt eesmärki “olla interaktiivne ja vastu võtta võimalikult palju asjakohast kasutaja sisendit” (mh suvaliste URL’ide kujul) ja anda vastused koheselt, ka erinevate sisenditega edasi-tagasi mängides. Serveripoolse töötluse puhul, kui päriselt simuleerida täpselt ette



Joonis 3. Tööriista arhitektuur (sisendid ja väljundid).

antud sisendiga olukorrad läbi (näiteks brauserite automaattestidega), võtaks vastuse andmine aega, oleks vaja muretseda serveri turvamise ning rahaliste kulutuste pärast (seevastu staatiliste failide hostimine on tihtipeale väga soodne või tasuta, näiteks GitHub Pages[1] abil). See seab aga ka piirangu: brauserite käitumisest tuleb luua mentaalne mudel, mida siis koodiks muuta.

Et rakenduse arhitektuur ja prototüüpimine viitas kasutajaliidesele, kus ühe sisendi muutmine võib muuta mitut teist komponenti, valiti deklaratiivseks ("reaktiivseks") kasutajaliidese arenduseks React (suures osas selle populaarsuse tõttu [33]). Puhta JavaScripti asemel valiti TypeScript, et aidata aeganõudvaid vigu vältida ja kirjutada rangemat koodi. TypeScript'i abil on võimalik näiteks tekitada küpsise seadetele oma tüüp, mis aitab vältida koodis olukordade tekitamist, kus proovitakse luua küpsise seadete komplekt, mis on vastuolus eesliidete `__Secure-` või `__Host-` eeltingimustega (vt lisa 2).

Public Suffix List'i töötlemiseks valiti aja säästmiseks juba valmisolev kiire ja üldjuhul korrektne (nt PSL *wildcard* sisendit arvestav) tööriist publicsuffixlist.js [34], mis on ainus leitud sobiv PSL-i töötlev JavaScripti teek, millele saab ette anda enda valitud nimekirja kehtivatest tippdomeenidest, mitte ei pea kasutama sisse-ehitatud (ja aegunud) nimekirjasid. See on oluline, et vältida kasutajale vale info andmist (näiteks, kui mõni

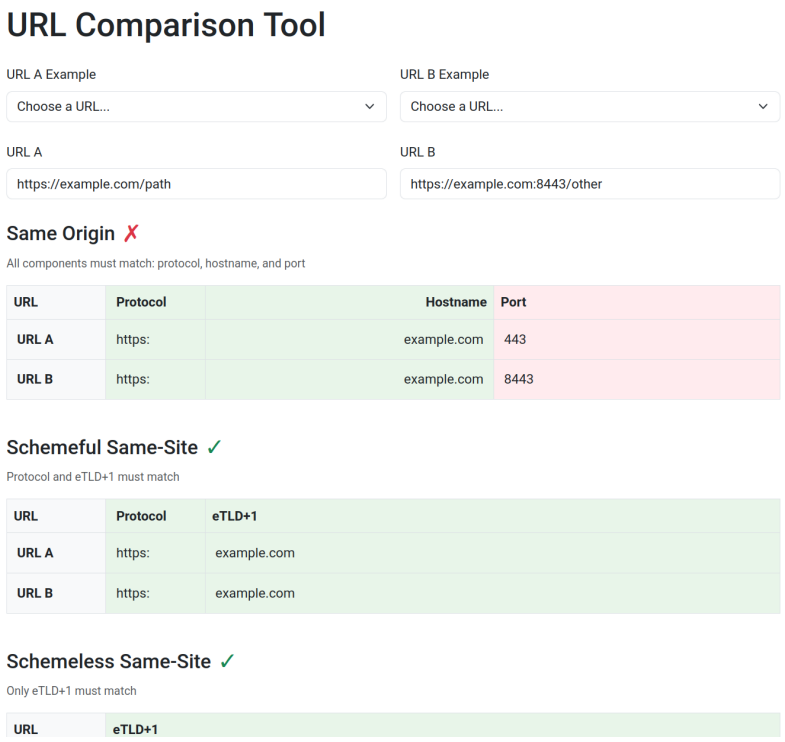
[1] `https://pages.github.com/`

rida on nimekirja juurde tekkinud või ära võetud). Nimekiri ise laaditakse rakenduse avamisel alla esmajärjekorras psl-formatted projekti abil [35], mis pakub andmemahu säästmiseks minimeeritud PSL-i (eemaldatud kommentaarid, ebavajalikud reavahetused). Ebaõnnestumisel proovitakse nimekiri alla laadida muude (ametlike) allikate kaudu.

2.3.2 Prototüüpimine ja iteratiivne arendus

Tähtis osa arendusprotsessist oli prototüüpimine ja tagasiside põhjal iteratiivne (p)arendus.

Esimene testversioon tööriistast, mis tegeles vaid saidi ja `origin` võrdlusega, tehti valmis keelemudelite (ingl *Large Language Model* (LLM)) abiga. Selleks kasutati kordamööda tasuta versioonidega keelemudeleid nagu DeepSeek, OpenAI ChatGPT ja Anthropic Claude. See oli kiire viis kätte saada tunnetus, kuidas tööriista kasutada oleks (isegi, kui see ei töötanud tehniliselt korrektselt). Kuvatõmmis sellest katsetusest on näha jooniselt 4.



Joonis 4. AI abil genereeritud prototüüp algsest tööriistast.

Seejärel alustati tööriista alfaversiooni arendamist (joonis 5), jälgides juba ülalmainitud tehnilist pinu (React, valitud PSL teegid jm). Alfaversioonile andsid tagasisidet antud töö juhendajad. Tagasiside põhjal visandati uued viisid sama info esitamiseks. Visanditest valiti koos juhendajatega välja üks, mis jõudis tööriista beetaversiooni.

Component	Victim	Attacker
scheme	https	http
host	app.example.com	1.dev.example.com
└ eTLD+1 (registrable domain)	example.com	example.com
└ eTLD (public suffix)	com	com
port	443 (default)	8080

Scope	Role	Matcher
✓ **schemelessly same-site** (eTLD+1*)	Victim & Attacker	*://**.example.com:*/**
✗ **(schemefully) same-site** (scheme; eTLD+1*)	Victim	https*://**.example.com:*/**
	Attacker	http*://**.example.com:*/**
✗ **same-origin** (scheme; host; port)†	Victim	https://app.example.com/*
	Attacker	http://1.dev.example.com:8080/*

* if eTLD+1 is null, then host
† same-origin scope can be relaxed with `document.domain` setter if both parties use it; this is being phased out

Joonis 5. Iteratsioon tööriistast: URL-sisenditest genereeritud tabelid.

Beetaversiooni arendusfaasis töötati välja ka küpsiste skoopide mudel, millest räägib täpsemalt järgmine alapeatükk (peatükk 2.3.3). Beetaversioon esitleti intervjuude käigus turvatestijatele (peatükk 3.3). Beetaversiooni kuvatõmmiseid võib näha tulemuste peatükist 3.1.

2.3.3 Mudeli loogika

Saidi ja `origin` skoopide puhul ei olnud erilist probleemi neid kasutaja sisendi põhjal tuletada, sest need on hästi defineeritud ja standardit järgides üsna kergelt implementeeritavad. Arvestada tuli vaid loetud äärejuhtumitega nagu näiteks URL-id, mille host on IP-aadress või kehtiv tippdomeen (seega puudub registreeritav domeen ehk eTLD+1 on väärtusega `null`). Mõnede äärejuhtumite käitlemist võib täpsemalt näha peatükist 3.1.

Töö üks keerulisematest ja kõige tähtsamatest osadest oli küpsiste skoopide loetlemine, kategoriseerimine ja siis igale skoobile täpse mudeli loomine vastavalt küpsise seadetele ja sisend-URL'ile, mis vastaks brauserite käitumisele. See vajas mitu korda erinevate nurkade alt lähenemist, nagu küpsise elutsükli kaardistamist, seal sees võimalikke operatsioonide kaardistamist, spetsifikatsioonide ja brauserite käitumise võrdlemist.

Pidevalt prooviti nii Mozilla Firefox'is kui Google Chromium'is läbi erinevaid olukordi,

kasutades autori varasemalt loodud tööriista `https://response.ee/`, mille abil saab kiirelt suvalisi HTTP vastuseid genereerida (siin eelkõige `Set-Cookie` vastuspäise jaoks) ja neid läbi proovida nii HTTP kui HTTPS peal, erinevatel alamdomeenidel, portidel ja teekondadel. Uurimistöö jaoks katsetamise kergendamiseks tuli response.ee tööriistale lisada TLS koondsertifikaadi tugi (ingl *TLS wildcard certificate*). Teine kasulik tööriist brauserite käsitsi katsetamisel oli brauserite arendaja tööriistad (ingl *developer tools*, lüh *devtools*), mille sees kasutati nii konsooli JavaScriptiga küpsiste manipuleerimiseks kui Application / Storage vahesakki, kust vaadata kõiki kehtivaid küpsiseid ja nende seadeid.

Lõpuks jõuti nimekirjani erinevatest toimingutest, mis on küpsistega üldse võimalikud:

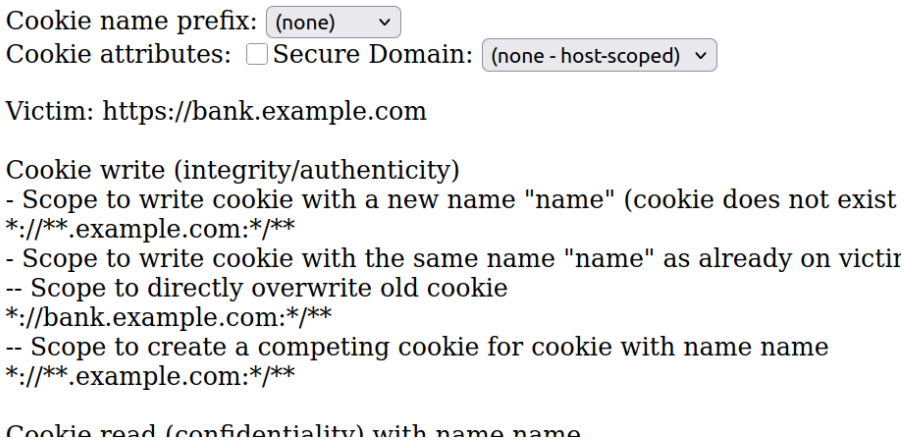
1. uue küpsise kirjutamine, kui varem sama nimega küpsist ei eksisteeri;
2. uue konkureeriva küpsise kirjutamine, kui varem sama nimega küpsis juba eksisteerib;
3. vana küpsise ülekirjutamine;
4. küpsise lugemine;
5. küpsise HTTP päringule kaasa panemine (olenevalt päringu algatajast);
6. küpsise kustutamine otse (võrdne ülekirjutamisega);
7. küpsise kustutamine kaudselt, paljude uute küpsiste kirjutamisega (*cookie jar overflow*);
8. küpsiste kustutamine `Clear-Site-Data` vastuspäise abil (teiste toimingutega võrreldes erijuhtum).

Eraldi elutsükli osana tuleb ära mainida ka küpsise kustumine selle aegudes (`Max-Age` või `Expires` atribuudi järgi, või seansiküpsiste puhul brauseriseansi lõppedes), kuid see on pigem teiste toimingute tagajärg kui ettevõetav toiming.

Seejärel prooviti jagada kõik loetletud operatsioonid omavahel loogilistesse kategooriatesse, mille lõpptulemust võib näha juba valminud tööriistast. Esmalt tundus näiteks loogilisem jagada kõik küpsiste (üle)kirjutamisega seotud operatsioonid ühte gruppi. Katsetades ja kaardistades välja operatsioonide skoobid erinevate seadete puhul, selgus aga, et parem rühmitus on jagada kirjutamisega seotud operatsioonid hoopis kahe teise kategooria vahel: kui juba sellise nimega küpsis eksisteerib ja kui ei eksisteeri. Nii ühtivad toimingute skoobid ka üldpildis paremini. See oli esialgu ootamatu, kuid reaalsuses vastavuses

erinevate RFC6265bis standardis toodud reeglitega. Näiteks olemasoleva sama nimega küpsise puhul, millel on `Secure` atribuut seatud, ei saa täpselt sama nime, domeeni ja teekonnaga küpsist ebaturvaliselt ühenduselt kirjutada ([4] sektsioon 5.7.3, samm 16).

Teatud hetkest oli lihtsam kombinatsioonide loetlemise ja endale erinevatesse struktuuridesse paigutamise asemel hakata kirjutama loogikat valmis pseudokoodina ja siis juba päris koodina ning hakata seda valideerima ning parandama. Joonisel 6 on toodud esimese selline (vigu sisaldav) demo, mis jagas kõik kirjutamisega seotud operatsioonid veel mõtteliselt ühte kategooriasse.



Joonis 6. Demoversioon küpsiste skoope interaktiivselt näitavast tööriistast.

Saades paika baasoperatsioonid ja nende jaotuse, sai hakata lisama iga operatsiooni juurde lisainfot: viiteid standarditele, brauseri bugidele ja/või võimalikele (teadatuntud nimedega) ründetehnikatele. Lisaks on võimalusel ära märgitud alternatiivsed viisid sama tulemuse saavutamiseks (näiteks küpsise ülekirjutamise asemel, millel on tihti kitsam skoop, saab *cookie jar overflow* tehnikaga lasta brauseril vastava küpsise eemaldada ja siis kirjutada (*mitte* ülekirjutada) uue sama nimega küpsise).

3 Tulemused

Loodud tööriist[1] ja selle lähtekood[2] on avalikult veebist kättesaadavad.

Töö tehisel puudub serveripoolne komponent: kogu tööriista loogika jookseb kasutaja brauseris. Lähtekoodi uuendamisel repositooriumi `main` harus käivitatakse automaatselt GitHub Workflow, mis ehitab valmis rakenduse staatilised failid, mida GitHub Pages serveerib.

Töö tulemusi on demonstreeritud probleemi lahendamiseks alapeatükis 3.1. Lisatud on ka kuvatõmmised. Töö tulemuste valideerimiseks võrreldi loodud tööriista varasemate töödega (alapeatükk 3.2) ning viidi läbi kvalitatiivne ekspertuuring turvatestijate seas (alapeatükk 3.3).

Lisaks leiti töö jooksul mitmeid (turva)probleeme seoses väliste osapooltega, mis on kirjeldatud eraldi lisas 3.

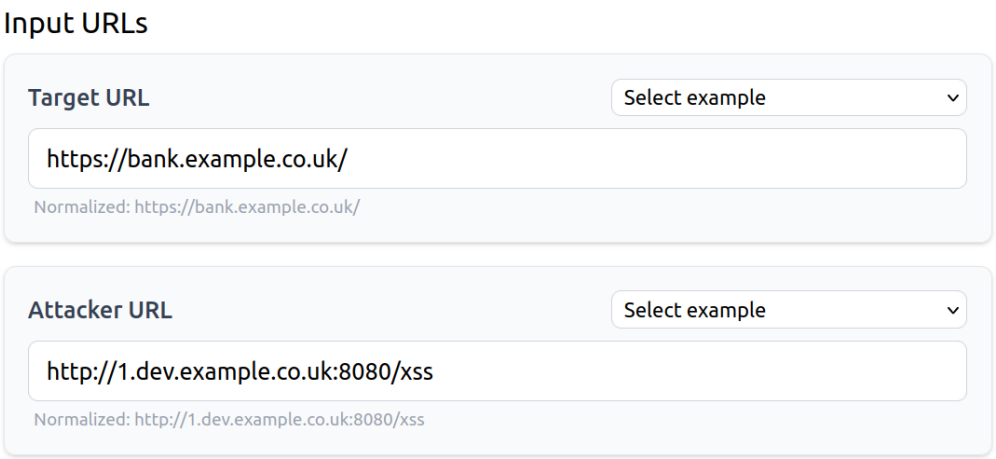
3.1 Tehise demonstratsioon probleemi lahendamiseks

Kujutagem ette olukorda, kus turvatestimisel on rakendus, mis asub aadressil `https://bank.example.co.uk/`. Sihtmärkrakendus kasutab küpsiseid, muuhulgas kasutajate eristamiseks (seansivõtme hoidmiseks). Tuvastatud on sama saidi raames avalik testkeskkond `http://1.dev.example.co.uk:8080/`, mille teekonnal `/xss` leidub XSS-tüüpi viga, mis võimaldab käivitada ründaja valitud JavaScript'i koodi, kui ohver seda lehekülge külastab. Mis mõju saab ründaja (osalise) kontrolli all olevalt URL-ilt sihtmärkrakendusele avaldada? Millised on küpsistega seotud ründeskoobid ja kuidas need muutuksid, kui sihtmärkrakendus kasutaks teistsuguste seadetega küpsiseid?

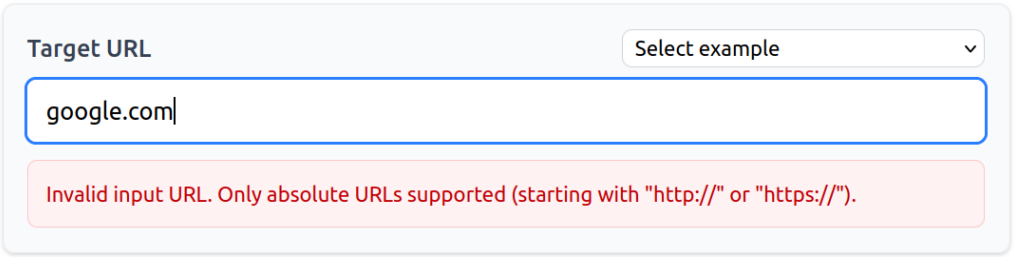
Esmalt laseb kasutajaliides anda sisendit sihtmärkrakenduse URL-i ja ründaja kontrolli all oleva URL-i kohta, kas valides mõne näite või sisestades kasutaja kontekstis teda

[1] `https://ukusormus.github.io/cookies/site-origin-cookie-scopes-visualizer/`
[2] `https://github.com/ukusormus/cookies`

huvitavad URL-id (joonis 7). URL-id normaliseeritakse ja valideeritakse – kui kasutaja üritab sisestada näiteks mittetäieliku aadressi, kuvatakse vastav veateade (joonis 8).



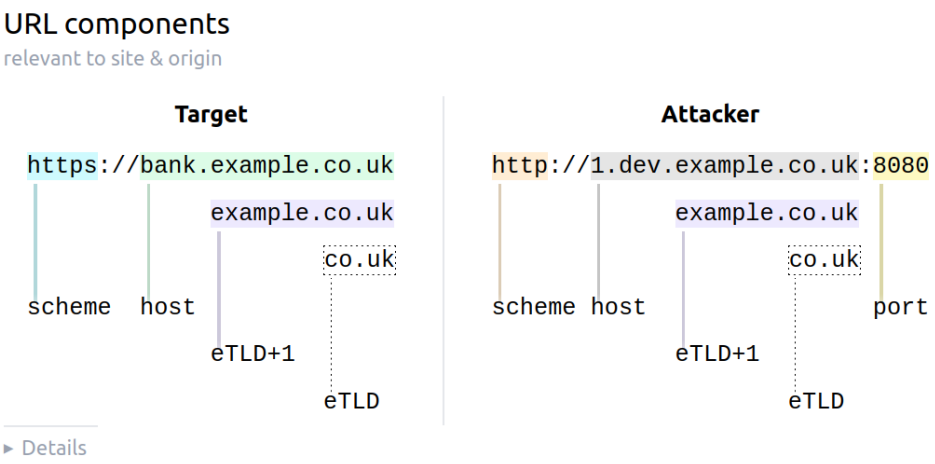
Joonis 7. Sisendid: sihtmärkrakenduse URL ja ründaja kontroll all olev URL.



Joonis 8. Sisendid: näide ebasobiva sisendi korral kuvatavast veateatest.

Seejärel võtab tööriist sisend-URL'id lahti URL komponentideks (joonis 9). Kuvatakse vaid need komponendid, mis on hiljem olulised nii üldiste saidi ja `origin` skoopide kui küpsistega seotud operatsioonide skoopide kuvamisel. Nagu näitest näha, on sama taseme komponendid sama taustavärviga, kui omavad täpselt sama väärtust (siin eTLD+1 ehk registreeritav domeen, lillaka värvitooniga). Korrektselt on ka eristatud kehtiv tippdomeen `co.uk`, mis ei ole konstandina sisse programmeeritud, vaid tuleneb lehekülje laadimisel allalaaditud värskeima Public Suffix List'i töötlemisest.

Siis on ülevaatlikult näidatud, kas ründaja ja sihtmärkrakendus kuuluvad ühe ja sama (skeemita või skeemiga) saidi või `origin` alla (joonis 10). Nii saab kasutaja kohe näha suuremat pilti, mis suhe on esimese ja teise URL-i vahel, mis annab esmase indikatsiooni võimalikkest rünnakutest. Praeguses näites kuuluvad URL-id sama skeemita saidi skoopi. Kui ükski skoop ei ühtiks, puuduks veebilehtedel brauseri turvapoliitika mõttes ühisosa ning rünnete tõenäosus oleks madalam. Esile pole tõstetud lihtsalt skoopidele



Joonis 9. Väljundid: sisend-URL'id oluliste komponentide visualiseerimine.

vastavad komponendid (näiteks `origin` puhul skeem, host, port), vaid näidatud ka regulaaravaldisele sarnast sobitajat, millest saab kergelt tuletada, millised aadressid kuuluksid veel vastavatesse skoopidesse (millised komponendid loevad ja mis ei loe). Siin näites `*://**.example.co.uk:*/**` ütleb meile, et nii sihtmärgi kui ründaja skeemita saidi skoopi kuuluvad kõik URL-id, mille host lõppeb sõnega `example.co.uk` – pole oluline, kas ja palju on alamdomeene, mis on URL skeem, mis on port ja milline on URL teekond.

General site & origin scopes

Target	Attacker	Scope
`*://**.example.co.uk:*/**`		✓ **schemelessly same-site** (eTLD+1) must match[1]
`https://**.example.co.uk:*/**`	`http://**.example.co.uk:*/**`	✗ **schemefully same-site** (scheme, eTLD+1) must match[1]
`https://bank.example.co.uk/**`	`http://1.dev.example.co.uk:8080/**`	✗ **same-origin** (scheme, host, port) must match[2]

▸ Details

Joonis 10. Väljundid: dünaamiline tabel `site` (skeemita ja skeemiga) ning `origin` skoopide kuvamiseks (vastavalt *schemelessly same-site*, *schemefully same-site* ja *same-origin*). Kui skoop sihtmärgi ja ründaja vahel kattub, kuvatakse mõlema tulba peale üks ühine väärtus.

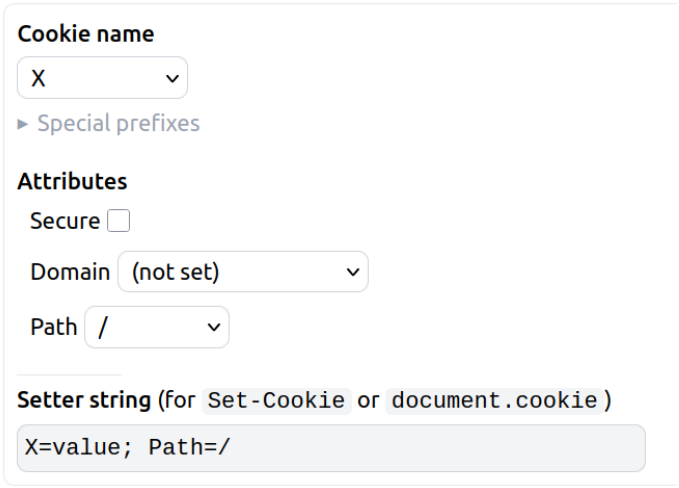
Rakendus arvestab ka korrektselt sisend-URL'ide äärejuhtumitega. Joonisel 11 on näidatud kahte sellist olukorda, kus eelpool mainitud "pole oluline, kas ja palju on alamdomeene" ei kehti. Mõlemal sisendil puudub registreeritav domeen ehk eTLD+1; `uk.com` puhul on tegemist kehtiva tippdomeeniga (kuulub PSL-i, kuid samal ajal hostitakse sisu).

Peale üldist komponentide ja skoopide paika saamist järgneb kasutajaliideses teine (ja

Target	Attacker	
`*://192.0.2.1:*/**`	`*://uk.com:*/**`	✗ (e
`http://192.0.2.1:*/**`	`https://uk.com:*/**`	✗ (s
`http://192.0.2.1/**`	`https://uk.com/**`	✗ (s

Joonis 11. Väljundid: dünaamiline tabel `origin` ja `site` skoopide kuvamiseks kahe äärejuhtumi korral.

viimane) sisendi andmise võimalus – valik erinevate küpsiste seadete vahel (joonis 12). `X` tähistab küpsise nime, mida rakendus kasutab.

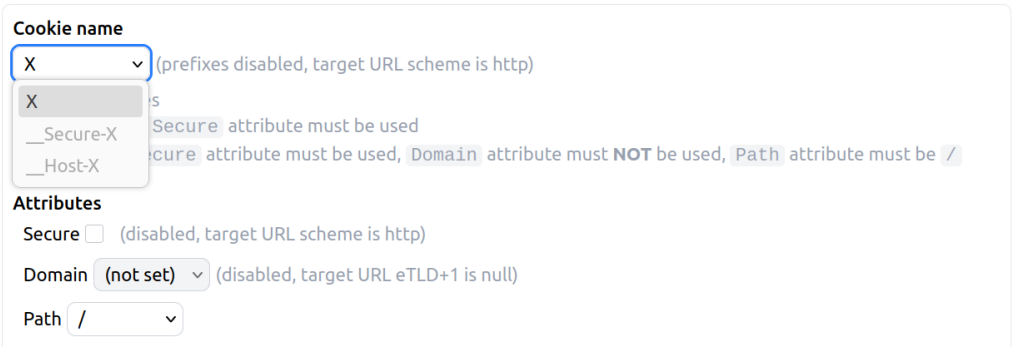


Joonis 12. Sisendid: küpsise sätted tööriista vaikeolekus.

Samamoodi on arvestatud erinevate äärejuhtumitega, kus teatud sätteid ei saa valida, kui eelnev sihtmärkrakenduse URL-i sisend ei ole vastav, et vältida võimatuid olukordi. Joonisel 13 on toodud näide sihtmärkrakenduse sisendi `http://internal-app` korral, kus ei lubata kasutada `Secure` atribuuti (URL skeem ei ole HTTPS) ja selle tõttu ka mitte küpsise nime eesliiteid, mis `Secure` atribuuti nõuavad, ning ka `Domain` atribuut pole valitav, sest URL-il puudub eTLD+1.

Minnes edasi küpsiste vaikesätetega, on nähtav kõige olulisem väljund: mis skoopide all peab ründaja olema, et saada küpsistega tegevusi sooritada. Seda on näidatud kokku viies kategoriseeritud tabelis.

Esimesed kolm tabelit (joonis 14) keskenduvad võimalustele otse sihtmärkrakenduse saidi



Joonis 13. Sisendid: küpsise sätted, sihtmärkrakenduse URL'i äärejuhtumite korral.

küpsistega manipuleerida (lugemine, (üle)kirjutamine, kustutamine). Valitud sisendite põhjal (sihtmärkrakenduse URL, küpsiste sätted) on näha, et ründaja ei saaks juba olemasoleva küpsisega `X` otse manipuleerida, kuid saaks luua uue küpsise `X`, mis saaks rakendust mõjutada (näiteks saaks kirjutada ründaja enda aktiivse seansivõtmega küpsise, et sundida ohvrit kasutama ründaja kontot; nii võib näiteks ohver kogemata sisestada tundlikke andmeid hoopis ründaja konto alla). Tabeleid võib lugeda kui *"skoobist `<Action scope>` võib olla võimalik sooritada `<Action>`; loe täpsemalt `<Note>` alt"*.

Tööriistas on ka kasutatud rakendusesiseseid viiteid (alla joonitud punktiirjoonega), millele klikkides viiakse kasutaja mõne (teise) tabeli rea juurde, mis lihtsustab ühe toimingu juures teise kirjeldamist.

Viimased kaks tabelit (joonis 15) kirjeldavad, millal pannakse sihtmärkrakenduse küpsis HTTP päringule kaasa, sõltuvalt päringu tüübist ja mis suhe on sihtmärkrakenduse ja päringu initseeria vahel. Näiteks on näha, et erinevalt Chrome'st, mis järgib skeemiga saidi definitsiooni, ei takista siin Firefox meie stsenaariumi põhjal ründaja leheküljelt sihtmärkrakenduse lehele HTTP päringut tehes küpsiste kaasapanekut, mis avab tee päringu võltsingu rünneteks (nn `simple request` tüübi puhul) krüpteerimata HTTP liikluse puhul (näiteks aktiivselt võrku kontrolliv ründaja).

Directly interact with an already existing cookie `X` set on target

Action scope	Action	Note
://bank.example.co.uk:/**	Read the cookie's value	• If cookie has the `HttpOnly` attribute set, attacker must have control of a server in scope to read the `Cookie` header from an HTTP request • `Path` is not considered a security feature: for server-side control, just make an HTTP request with the needed path; for non-`HttpOnly` cookies, JavaScript
://bank.example.co.uk:/**	Delete the cookie	• If cookie has `HttpOnly` attribute set, attacker must have control of a server in scope to return `Set-Cookie` header in HTTP response • It is still possible to indirectly delete an `HttpOnly` cookie from JS by triggering cookie eviction, and then overwrite it
	Overwrite the cookie's value	

Create a new cookie that affects target

Action scope	Action	Note
*://**.example.co.uk:*/**	Create completely new cookie with name `X` (does not exist in target's cookie store)	• The scope for cookie bombing is always as wide as just setting a cookie with name `X` (no prefixes or `Secure` attribute needed) • Without `__Secure-` or `__Host-` prefix, the server has no way of telling if the cookie was set with the `Secure` attribute
*://**.example.co.uk:*/**	Create a competing a.k.a. shadowing cookie (exact name `X` already exists)	• Successful result is two cookies with the exact same name, but (possibly) different values • Something in the key of the new cookie must be different from the original (`Path`, `Domain` or `Partitioned` attribute)
*://**.example.co.uk:*/**	Create new cookies to evict the existing cookie `X` (which has not got the `Secure` attribute); a.k.a. cookie jar overflow	• Even `HttpOnly` cookies can be evicted from JS • The order of cookie eviction defines that non-secure cookies are evicted before `Secure` cookies

Purge cookies from the whole schemeless site *://**.example.co.uk:*/**

Action scope	Action	Note
https://**.example.co.uk:*/**	Delete all cookies for the whole schemeless site	• Attacker must have control of a server in scope to add `Clear-Site-Data` header to HTTP response

Joonis 14. Väljundid: millal on võimalik sihtmärkrakenduse küpsiseid mõjutada.

Target's cookie is attached to simple (non-preflighted) HTTP request to target URL (matching target cookie's read scope) from attacker in <scope>

Basis for "CSRF" (or OSRF, On-Site Request Forgery) and XSSI attacks.

▸ About simple requests & JSON

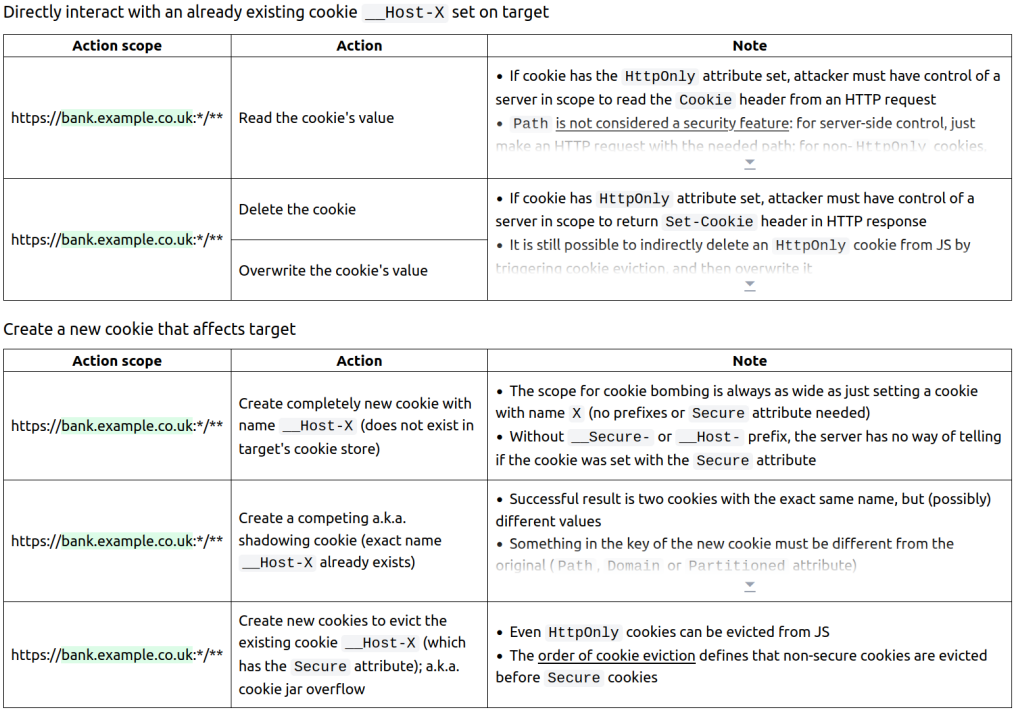
<scope>	Policy applied	Note
https://**.example.co.uk:*/**	*Schemeful* same-site (Chrome)	• Can always be attached by the attacker (the `SameSite` attribute has no effect on same-site requests!) • Browsers like Firefox are expected to follow suit including scheme in site definition at some point
*://**.example.co.uk:*/**	*Schemeless* same-site (Firefox)	
(doesn't match same-site)	Cross-site (depends on definition of site)	• A cookie may or may not be attached - depends if the `SameSite` attribute is set, its value (e.g., `Lax` = only top-level GET requests allowed), victim's browser and its settings (e.g., 3rd-party cookie blocking, separation per top-level site), etc. • Ideas for bypassing restrictions

Target's cookie is attached to non-simple HTTP request to target URL (matching target cookie's read scope) from attacker in <scope>

<scope>	Policy applied	Note
https://bank.example.co.uk/**	Same-origin policy (SOP)	• Credentials (including cookies) are included in same-origin requests by default • Credentials can be explicitly omitted, e.g., in some HTML elements via the `crossorigin` attribute or in JS, Fetch API's `credentials` option or XMLHttpRequest API's `withCredentials` property
(doesn't match same-origin)	Cross-origin	• Cross-origin non-simple requests are preflighted. Note that a CORS-preflight request itself never includes credentials (TLS client certificates are currently included in Chromium, a bug) • The response to the preflight request can indicate credentials are allowed in

Joonis 15. Väljundid: millal pannakse küpsis HTTP päringule kaasa, sõltuvalt päringu tüübist ja päringut initseeriva lehekülje suhtest sihtmärkrakendusega.

Muutes küpsiste seadeid, muutuvad ka skoobid. Valides kõige rangema küpsise seade, `__Host-` eesliite, on näha, et rangemaks (kitsamaks) muutuvad ka skoobid, kust küpsistega operatsioone saab sooritada (joonis 16).



Directly interact with an already existing cookie `__Host-X` set on target

Action scope	Action	Note
https://bank.example.co.uk:*/**	Read the cookie's value	• If cookie has the `HttpOnly` attribute set, attacker must have control of a server in scope to read the `Cookie` header from an HTTP request • `Path` is not considered a security feature: for server-side control, just make an HTTP request with the needed path; for non-`HttpOnly` cookies
https://bank.example.co.uk:*/**	Delete the cookie Overwrite the cookie's value	• If cookie has `HttpOnly` attribute set, attacker must have control of a server in scope to return `Set-Cookie` header in HTTP response • It is still possible to indirectly delete an `HttpOnly` cookie from JS by triggering cookie eviction, and then overwrite it

Create a new cookie that affects target

Action scope	Action	Note
https://bank.example.co.uk:*/**	Create completely new cookie with name `__Host-X` (does not exist in target's cookie store)	• The scope for cookie bombing is always as wide as just setting a cookie with name `X` (no prefixes or `Secure` attribute needed) • Without `__Secure-` or `__Host-` prefix, the server has no way of telling if the cookie was set with the `Secure` attribute
https://bank.example.co.uk:*/**	Create a competing a.k.a. shadowing cookie (exact name `__Host-X` already exists)	• Successful result is two cookies with the exact same name, but (possibly) different values • Something in the key of the new cookie must be different from the original (`Path`, `Domain` or `Partitioned` attribute)
https://bank.example.co.uk:*/**	Create new cookies to evict the existing cookie `__Host-X` (which has the `Secure` attribute); a.k.a. cookie jar overflow	• Even `HttpOnly` cookies can be evicted from JS • The order of cookie eviction defines that non-secure cookies are evicted before `Secure` cookies

Joonis 16. Väljundid: millal on võimalik sihtmärkrakenduse küpsiseid mõjutada, küpsise nime `__Host-` eesliite korral.

Rakendus võimaldab leide teistega jagada, olgu leidudeks tööriista abil avastatud võimalikud ründed või selge põhjendusega soovitused, miks peaks küpsiste seadeid muutma. Selle asemel, et paluda töökaaslasel või kliendil tööriistas manuaalselt samasugused seaded valida, et teine osapool saaks olukorda ise läbi mängida, piisab tööriista URL-i jagamisest, mis hoiab endas alati hetkeolekut. Seda ei saadeta kunagi serveri poolele, sest on kaasas URL *fragment* osas (peale # sümbolit). Kui tööriista avamisel on URL *fragment* olemas, proovitakse laadida sisendid sellest.

Näide sellisest URL-ist, mille seaded on demonstreeritud stsenaariumi lõppseisus:

`https://ukusormus.github.io/cookies/site-origin-cookie-scopes-visualizer/#target=https%253A%252F%252Fbank.example.co.uk%252F&attacker=http%253A%252F%252F1.dev.example.co.uk%253A8080%252Fxss&name=__Host-X`

Seega on täidetud nõue ja eesmärk olla viidatav näiteks turvatesti raportites või koolituse slaididel.

3.2 Võrdlus varasemate töödega

Autor pole teadlik muu sellise interaktiivset tööriista olemasolust, mis näitaks dünaamiliselt küpsistega seotud tegevuste (ründe)skoope vastavalt sisendparameetritele. Võrreldes varasemate küpsiste ründetehnikaid kirjeldavate töödega, keskendub praegune töö just küpsistega teostatavate baasoperatsioonide kaardistamisele, lisades vajadusel nende najale teadatuntud nimedega ründetehnikaid. Seni küpsistega seotud ründetehnikaid kirjeldavate allikate (teadustööde, blogide ja agregaatorite) puhul võib loodud tööriista lugeda täiendavaks – allikad saavad viitada tööriistale ja tööriist allikatele.

Lähemaks võrdlusmomendiks võib lugeda varasemate tööde all toodud kaks interaktiivset tööriista küpsiste käitumisega katsetamiseks. Võrdlus on toodud tabelis 1.

Tabel 1. Võrdlus varasemate tööriistadega küpsiste käitumise katsetamiseks.

Tegevus	**Loodud tööriist**	**setcookie .net**	**set-cookie .info**
Ei vaja tagaserverit	+		
Näitab kasutaja brauseri reaalset käitumist, mitte mudelit		+	+
Küpsiste seadete muutmine	+	+	+
Vabas vormis `Set-Cookie` sisend			+
Valitavad sisend-URL'id	+		
Katsetamine teiste saitidega (*cross-site*)	+		
URL komponentide ja skoopide visualiseerimine	+		
Suunatud pigem arendajatele		+	+
Suunatud pigem turvatestijatele	+		
Ulatuslik rünnete kontekst koos märkmete ja viidetega	+		
Saab näidata, millal küpsis päringule kaasa pannakse (*same-site, simple request*)	+	+	+
Näitab, millal küpsis päringule kaasa pannakse, kõik päringute variandid	+	+	+
Näitab, millal küpsist saab JavaScriptiga lugeda	+	+	+
Kõik muud kategoriseeritud toimingud küpsistega	+		

Ainsaks miinuseks, mis on loodud tööriista arhitektuuri sisse kirjutatud, on see, et tööriist üritab mudeldada brauserite käitumist, mitte ei näita kasutajale, mida (just) tema brauser teeb. Teised kaks tööriista kirjutavad päriselt brauserisse küpsiseid. Teisest küljest annab

see valik aluse paljudele teistele plussidele, nagu näiteks tagaserveri mittevajamine ja võimalus mudeldada käitumist suvaliste URL-idega, mitte ainult sama saidi raames nagu teiste tööriistade puhul. Seda mudeldamise miinust oleks võimalik ka erinevatel viisidel vähendada, näiteks kirjutades automaattestid, mis kõigi suuremate brauserimootorite käitumist kinnitaksid ja nii väljundi usaldusväärsust tõstaks.

Miinuste osas, mis on antud arhitektuuri sees parandatavad: hetkel ei toetata vabas vormis `Set-Cookie` sisendit ja see on ka üks plaanitavatest tulevikuarendustest. Samamoodi on tööriist hetkel suunatud pigem turvatestijatele kui tavaarendajatele, kuid seda saaks vajadusel parendada suurema konteksti andmise ja disainivalikutega (näiteks valikuline info peitmine).

Võrdlusmomedi URL-i(de) visualiseerimise osas saab luua varasema URL-i komponentide visualiseerimise tööriistaga `https://url-parts.glitch.me/`, mis on loodud Google'i töötaja(te) poolt [24]. Võrdlus on toodud tabelis 2. Kui kõrvale jätta hunnik äärejuhtumeid, millega Google'i tööriist ei tegele, siis olulisemad eelised Google'i tööriista ees on kahe URL-i võrdlus (mh vastavate skoopide võrdlus), skeemita saidi skoobiga arvestamine ja alati viimase PSL'i versiooni kasutamine. Google'i tööriist on natuke teise eesmärgiga ja näitab välja ka muid URL'i komponente nagu näiteks "`search`", aga need pole saidi ja `origin` mõistete jaoks olulised.

Tabel 2. Võrdlus varasema tööriistaga URL-i komponentide visualiseerimiseks.

Funktsionaalsus	**Loodud tööriist**	**url-parts .glitch.me**
Interaktiivne	+	+
Toetab enimlevinud variante domeeninimedest	+	+
Lubab kahte URL'i võrrelda	+	
Toob visuaalselt välja kõik URL-i komponendid		+
Toetab PSL-i	+	+
Kasutab alati viimast versiooni PSL'ist	+	
Toetab tippdomeene, mis ei kuulu PSL'i	+	
Toetab hoste, mis kuuluvad ise PSL-i (nt `uk.com`)	+	
Toetab IP-aadresse	+	
Toetab domeeninimesid, mis sisaldavad Punycode'i või mitte-ASCII'd	+	

Tabel jätkub...

Tabel 2 – *Tabel jätkub...*

Funktsionaalsus	**Loodud tööriist**	**url-parts .glitch.me**
Toetab korrektselt normaliseerimata sisend-URL'e (nt `http://AA.co`)	+	
Näitab mõisteid `site`, `origin`	+	+
Toetab skeemita saidi mõistet	+	

3.3 Tööriista valideerimine turvatestijate seas

Eesti küberturbeettevõtte Clarified Security OÜ töötajate seas viidi läbi kvalitatiivne uuring, et valideerida tööriista kasulikkust ja kasutusmugavust ning saada tagasisidet võimalikke parenduste kohta[3].

3.3.1 Intervjueerimise protsess

Kahe päeva jooksul intervjueeriti autori poolt individuaalselt seitset inimest (ühte üle videosilla, kuute kohapeal ettevõtte kontoris). Kõigi osalejate tööülesannete hulka kuulub veebirakenduste läbistustestimine ehk kõik osalejad kuuluvad oma rolli poolest tööriista kasutajate sihtgruppi. Kõik osalejad tutvusid tööriistaga esmakordselt alles intervjuu ajal.

Ekspertintervjuud viidi läbi järgneval kujul: esmalt tutvustati uuringu eesmärki (tööriista valideerimine ja tagasiside saamine) ja uuringu oodatavat tulemit (loodav anonüümne kokkuvõte), siis küsiti nõusolekut heli ja ekraanipildi salvestamiseks ja viimaks selgitati intervjuu struktuuri. Intervjuu struktuur koosnes suuremas osas tööriista iseseisvast proovimisest, samal ajal lastes subjektil kõva häälega mõelda (mida hetkel näen, tunnen, arvan, mis jääb segaseks jne) [36]; lõpuks küsiti lisaküsimusi nagu "kas ja kus võiks tööriistast olla tööl kasu?" ning "mis teeks tööriista paremaks ja/või kasulikkumaks?".

Pärast intervjuude läbiviimist analüüsiti läbi tehtud märkmed ja salvestised, toodi igast intervjuust välja põhilised jutupunktid ning võrreldi, mis teemad käisid intervjuudest korduvalt läbi.

[3]Clarified Security töötajate hulka kuuluvad ka töö autor ja juhendajad (ei kuulu intervjueeritavate hulka).

3.3.2 Intervjuude tulemused

Kõik seitse intervjueeritavat ütlesid, et kasutaksid tööriista tööülesannete täitmisel ja vastasid jaatavalt ka küsimusele "kas soovitaksid tööriista sõbrale?". Põhilisteks olukordadeks, kus tööriista tööl võiks kasutada, toodi välja:

- Kui turvatestimisel esineb olukord, kus (sõsar)rakendused on mitmel alamdomeenil, et paremini mõista, kuidas ühelt alamdomeenilt vea leidmisel on võimalik mõjutada rakendust teisel alamdomeeni.
- Üldise õppematerjali või referentsina, kui teemasid on vaja meelde tuletada või ka uute tehnikatega tutvuda (mitu intervjueeritavat tõid välja enda jaoks uusi avastusi, näiteks nimetud küpsised).

Uuringus osalenud koolitaja mainis, et ta paneks juba praeguse iteratsiooni tööriistast koolituse slaididele ja see oleks koolituse valmistamisel küsimuste lahendamiseks "ülikasulik".

Mitu intervjueeritavat kiitsid tööriista detailsust, info kokkukoondamist ja visuaalset ülevaadet (m.h. URL komponentide visualiseerimist). Kordagi ei toodud välja, et mõni tööriista kui mudeli ennustatav käitumine näiks vale.

Läbivalt tulid tagasisidest välja järgnevad neli soovitust, mis teeks tööriista kasulikkumaks:

1. Anda ette rohkem konteksti legendi või juhendi kujul – tööriist nõuab hetkel arvestataval tasemel käsitletud teemade ja terminite kohta eelnevaid teadmisi (näiteks mainiti, et hetkel ei oleks kindlasti sihtgrupp juunior-arendajad); sisendid ja väljundid on ette antud ilma suurema jututa ja kasu oleks kontekstist, kuidas näiteks tabeli väljundit peaks lugema või kuidas erinevad UI osad omavahel seotud on.
2. Näidata suurema ekraanilaiuse korral sisendeid (URL-id ja küpsise seaded) pidevalt kas ekraani kõrval või ülal, et vältida vajadust üles-alla kerida ja meeles hoida, mida sai sisestatud.
3. Näidata küpsistega tehtavate toimingute tabelite juures, kas ründaja URL kattub vastava toimingu skoobiga või mitte (hetkel tuleb see vastavalt näidatud skoopidele ise tuletada); lahenduseks pakuti välja nii praeguste tabelite täiendamist erinevate taustavärvidega kui eelnevat lühikokkuvõtet stiilis *"ründaja saab teha X ja Y, ei saa teha Z ja W"*, kus näiteks toimingule *X* klikkides viidaks kasutaja vastava

põhjalikuma tabeli veeru juurde.

4. Tekitada ülevaatlikum võrdlusmoment eri küpsiste seadete ja ründaja URL-ide muutmise korral. Osaliselt aitaks selle probleemi lahendamisele kaasa juba eelneva kahe punkti lahendamine, kuid lisaks pakuti välja näiteks võrdlusmaatriksi funktsionaalsust.

Veel mõned olulisemad tagasisidepunktid olid soovitused uute sisendite osas, eelkõige lisada üks märkeruut, kas ründajal on serveripoolne kontroll või mitte ja teine märkeruut küpsiste seadete alla `HttpOnly` atribuudi jaoks. Need muutused võimaldaksid muuhulgas lihtsustada küpsistega tehtavate toimingute tabelites märkmete (*Notes*) veerus kuvatavat teksti, mis praegu neid juhtumeid eraldi tingimustena lahti kirjeldab, kuigi see ei pruugiks vastavalt kasutaja sisendile vajalik olla.

Peale sõnastusparanduste ja väikeste disainisoovide avaldasid kolm inimest seitsmest soovi tumedamates toonides kasutajaliidese variandi järele (ingl *dark mode*).

Kokkuvõtvalt on loodud tööriist juba praegu väärtuslik abimees veebiturvalisusega seotud professionaalidele, olles tehniliselt täpne ja sisukas ning koondades kompaktselt kokku info paljudest allikatest. Samas on veel palju võimalusi tööriista oluliselt paremaks teha, mis puudutavad eelkõige info esitlemist ja pakendamist kasutajaliideses.

4 Diskussioon

4.1 Järeldused tulemustest

Veebitehnoloogia algusaegadest pärit HTTP laiend-mehhanismi uurimine näitas, et aastatega kihtide ja erisuste lisandumine nii küpsiste enda toimimisse kui keskkonda nende ümber on muutnud veebiküpsiste sügavama mõistmise ootamatult keeruliseks – nii keeruliseks, et endiselt on võimalik leida turvavigu brauserite käitumises kui saada positiivne reaktsioon turvatestijatelt, kellele oli loodud tööriistast, mis teadmisi koondab ja uut moodi esitab, teema mõistmisel abi. Kuid isegi siis on loodud tööriistal mitmed olulised piirangud.

4.2 Töö piirangud

Küpsistega seotud toimingute skoopide genereerimise loogika kirjutati suurelt jaolt kahe brauseri käitumist käsitsi testides ja standardeid lugedes ning sellest mudeli luues. Seda on märgitud ka tööriista leheküljel. Programmi usaldusväärsust saaks tõsta, kui kirjutada igale toimingu skoobile vastav automaattest, mis loogikat üle kinnitaks ja vajadusel brauseritevahelised erinevused välja tooks (miks mitte ka muutused ajas). Veel sammu edasi minnes kaoks käsitsi loogika kirjutamine, vaid skoopide genereerimise kood koostatakski automaattestide pealt. Alternatiiv oleks ka teha lõviosa töötlust tagaserveri poolel jooksu pealt, kuid sellel oleks loomulikult omad miinused (finantsilised kulud, sisendi töötlemine ja koodi turvaliselt käivitamine, kasutaja jaoks tulemuste ootamise aeglus jne).

Töös ei tegeletud süvendatult küpsiste käitumisega olukordades, kus üks lehekülg kaasab teist (nt HTML `iframe` elemendi abil). Seda on mainitud ka tööriista leheküljel.

Tööriista küpsiste skoopide all toodi välja tabeli rida, mis käsitleb saidiüleseid päringuid (`cross-site simple request`), kuid selle sees võimalikke kombinatsioonide süstemaatilise kaardistamisega ei tegeletud. Võimalus sellises olukorras küpsist päringuga kaasa panna sõltub muuhulgas `SameSite` atribuudist, selle vaikeväärtusest (brauserite vahel erinev [12]), kolmandate osapoolte küpsiste blokeerimise poliitikatest jne. Sarnast sõnastus

on kasutatud ka vastavas tabeli reas.

4.3 Võimalused edasiseks uurimiseks ja arenduseks

Valminud interaktiivsele tööriistale oleks võimalik juurde teha erinevaid parendusi ja lisafunktsionaalsusi. Esmalt tasuks ette võtta nimekiri intervjuudest enim läbikäinud soovidest ja soovitustest (vaata peatükk 3.3.2). Tööriista teeks mugavamaks võimalus ette anda terve `Set-Cookie` päis ja tuletada just selle vastava küpsisega seotud ründeskoobid. Täielikumaks teeks tööriista sisend HSTS (HTTP Strict Transport Security) erinevate seadete kohta (nagu `includeSubdomains`), et näidata selle mõju ründeskoopide kitsendamisele [37].

Nagu piirangute all mainitud, jäi töö skoopi vaid olukord, kus ohver külastab ründaja linki peataseme saidi (ingl *top-level site*) kontekstis. Välja jäi küpsiste käitumise loogika kaardistamine olukordades, kus saidid kaasavad enda dokumendi sisse kolmandaid osapooli, näiteks pildi või HTML `iframe` elemendina. Ka see valdkond on pidevas muutumises, sest brauserid liiguvad suunas, kus kaasatud kolmandatel osapooltel vaikimisi küpsiseid kirjutada ei lasta. Õhku jäävad küsimused, kuidas mõjutavad ründesituatsioone CHIPS (Cookies Having Independent Partitioned State) koos selle `Partitioned` atribuudiga, Related Website Sets (RWS), Storage Access API ja muud mehhanismid, millega brauserid hetkel aktiivselt katsetavad [38].

Et tööriista ümber on sobiv keskkonna kest MkDocs kujul olemas ja ka võimalikud menüüpunktid algse idee raames kaardistatud, on endiselt võimalus jätkata selle idee arendamisega, dokumenteerides süstemaatiliselt “kõike küpsistest” (eriti turvaperspektiivist).

Public Suffix List-i uurides tekkis lisaks kaks võimalikku ideed edasisteks töödeks. Esiteks ei arvestanud sisuliselt ükski PSL implementatsioon erinevate äärejuhtumitega nagu näiteks punktiga lõppevad täielikud domeeninimed (ingl *Fully Qualified Domain Name* (FQDN)) ja omasid üldiselt muid probleeme nagu võimetus laadida sisse uusi kehtivate domeenide nimekirju. Seega on veel ruumi uuteks, kiireteks, korrektseteks ja laiendatavateks implementatsioonideks. Teiseks võib olla huvitav uurimissuund uue tööriista loomiseks või mõne olemasoleva tööriista täiendamiseks, mille abil oleks võimalik otsida kirjeid kõigist ajalooliselt PSL-is olnud domeenidest. Umbes 2015. aastani leidub GitHub-is oleva koodihoidla ajalugu, varasemad kirjed tuleks üles otsida brauserite koodi

versioonihaldussüsteemidest [39]. Siia juurde saaks lisada ka kirjed uuest Related Website Sets nimekirjast, rääkimata muudest olemasolevatest allikatest. Eesmärk oleks lihtsustada võimalike alamdomeenide kaardistamist, sest suurematel ettevõtetel võib olla mõnel ajaloolisel domeenil vähemturvatud rakendusi.

5 Kokkuvõte

Lõputöö raames loodi avatud lähtekoodiga interaktiivne tööriist, mis aitab aru saada, kuidas saab ründaja kontrolli all olev veebilehekülg mõjutada ette antud sihtmärkrakenduse veebilehekülge ja selle küpsiseid, kui ohver ründaja kontrolli all olevat lehekülge külastab ning kuidas erinevad küpsiste seaded ründeskoobi suurust mõjutavad.

Ekspertintervjuude käigus kinnitati, et loodud tööriistast on kasu veebirakenduste läbistustestijatele. Tööriist aitab veebirakenduste läbistustestidel paremini mõista ründevõimalusi ja kaitsemeetmete mõju. Tööriista arendamise jooksul tuvastati lisaks mitmeid küpsistega seotud turvaprobleeme, mis on kirjeldatud töö lisas.

Kasutatud kirjandus

[1] Henrik Nielsen *et al. Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Juuni 1999. DOI: 10.17487/RFC2616. URL: https://www.rfc-editor.org/info/rfc2616.

[2] Lou Montulli. *The Irregular Musings of Lou Montulli: The Reasoning behind Web Cookies*. The irregular musings of Lou Montulli. 14. mai 2013. URL: https://montulli.blogspot.com/2013/05/the-reasoning-behind-web-cookies.html (vaadatud 13.04.2025).

[3] Michal Zalewski. *The Tangled Web: A Guide to Securing Modern Web Applications*. No Starch Press, november 2011, lk. 60. ISBN: 978-1-59327-388-0. URL: http://archive.org/details/thetangledwebaguidetosecuringmodernwebapplications (vaadatud 13.04.2025).

[4] Steven Bingler, Mike West ja John Wilander. *Cookies: HTTP State Management Mechanism*. Internet Draft draft-ietf-httpbis-rfc6265bis-20. Internet Engineering Task Force, 17. märts 2025. 61 lk. kokku. URL: https://datatracker.ietf.org/doc/draft-ietf-httpbis-rfc6265bis-20 (vaadatud 05.05.2025).

[5] *Curl - HTTP Cookies*. URL: https://curl.se/docs/http-cookies.html (vaadatud 05.05.2025).

[6] WHATWG. *HTML Standard - Web Storage*. URL: https://html.spec.whatwg.org/multipage/webstorage.html (vaadatud 05.05.2025).

[7] WHATWG. *HTML Standard - Site*. URL: https://html.spec.whatwg.org/multipage/browsers.html#obtain-a-site (vaadatud 17.05.2025).

[8] WHATWG. *Fetch Standard - Credentials*. URL: https://fetch.spec.whatwg.org/#credentials (vaadatud 05.05.2025).

[9] R. Fielding, M. Nottingham ja J. Reschke. *RFC9110*. IETF HTTP Working Group Specifications. Juuni 2022. URL: https://httpwg.org/specs/rfc9110.html#credentials (vaadatud 05.05.2025).

[10] *HTTP Authentication - HTTP | MDN*. 13. märts 2025. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Authentication (vaadatud 05.05.2025).

[11] *Cookies Having Independent Partitioned State (CHIPS) - Privacy on the Web | MDN*. 4. aprill 2025. URL: https://developer.mozilla.org/en-US/docs/Web/Privacy/Guides/Privacy_sandbox/Partitioned_cookies (vaadatud 05.05.2025).

[12] Rowan Merewood. *SameSite Cookies Explained*. web.dev. URL: https://web.dev/articles/samesite-cookies-explained#default-behavior-changes (vaadatud 05.05.2025).

[13] Zakhar Fedotkin. *Stealing HttpOnly Cookies with the Cookie Sandwich Technique*. PortSwigger Research. 22. jaanuar 2025. URL: https://portswigger.net/research/stealing-httponly-cookies-with-the-cookie-sandwich-technique (vaadatud 05.05.2025).

[14] Ken Peffers *et al. Design Science Research Process: A Model for Producing and Presenting Information Systems Research*. 4. juuni 2020. DOI: 10.48550/arXiv.2006.02763. arXiv: 2006.02763 [cs]. URL: http://arxiv.org/abs/2006.02763 (vaadatud 05.05.2025). Eelnevalt avaldatud.

[15] Marco Squarcina *et al.* „Cookie Crumbles: Breaking and Fixing Web Session Integrity". Teoses: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, august 2023, lk. 5544. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/squarcina.

[16] Xiaofeng Zheng *et al.* „Cookies Lack Integrity: Real-World Implications". Teoses: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, august 2015, lk. 707–721. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/zheng.

[17] Ankur Sundara. *Cookie Bugs - Smuggling & Injection*. arxenix's blog. 5. mai 2023. URL: https://blog.ankursundara.com/cookie-bugs/ (vaadatud 14.05.2025).

[18] filedescriptor. *The cookie monster in your browsers*. URL: https://hitcon.org/2019/CMT/slide-files/d1_s3_r0.pdf (vaadatud 14.05.2025).

[19] Huli. *Interesting and Practical Cookie Bomb | Beyond XSS*. URL: https://aszx87410.github.io/beyond-xss/en/ch4/cookie-bomb/ (vaadatud 14.05.2025).

[20] Jorian Woltjer. *Cross-Site Request Forgery (CSRF) | Practical CTF*. 3. aprill 2025. URL: https://book.jorianwoltjer.com/web/client-side/cross-site-request-forgery-csrf (vaadatud 14.05.2025).

[21] *Cookies Hacking - HackTricks*. URL: https://book.hacktricks.wiki/en/pentesting-web/hacking-with-cookies/index.html (vaadatud 14.05.2025).

[22] *Set-Cookie - HTTP | MDN*. 8. aprill 2025. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Set-Cookie (vaadatud 16.05.2025).

[23] Eiji Kitamura. *"Same-Site" and "Same-Origin" | Articles*. web.dev. URL: https://web.dev/articles/same-site-same-origin (vaadatud 16.05.2025).

[24] Sam Dutton. *Samdutton/Url-Parts*. 28. aprill 2025. URL: https://github.com/samdutton/url-parts (vaadatud 16.05.2025).

[25] *Format · Publicsuffix/List Wiki*. URL: https://github.com/publicsuffix/list/wiki/Format (vaadatud 18.05.2025).

[26] WHATWG. *URL Standard - Origin*. URL: https://url.spec.whatwg.org/#concept-url-origin (vaadatud 17.05.2025).

[27] WHATWG. *URL Standard - Public Suffix*. URL: https://url.spec.whatwg.org/#host-public-suffix (vaadatud 17.05.2025).

[28] WHATWG. *URL Standard - Registrable Domain*. URL: https://url.spec.whatwg.org/#host-registrable-domain (vaadatud 17.05.2025).

[29] Chris Buckley. *Cookie Test*. URL: https://setcookie.net/ (vaadatud 17.05.2025).

[30] Kendrick Grünberg. *set-cookie.info*. URL: https://set-cookie.info/ (vaadatud 15.05.2025).

[31] *Cookie Tossing - HackTricks*. URL: https://book.hacktricks.wiki/en/pentesting-web/hacking-with-cookies/cookie-tossing.html#cookie-tossing (vaadatud 18.05.2025).

[32] Thomas Houhou. *Cookie Tossing: Self-XSS Exploitation, Multi-Step Process Hijacking, and Targeted Action Poisoning*. URL: https://www.thomashouhou.com/post/cookie-tossing-attacks/ (vaadatud 18.05.2025).

[33] *Technology | 2024 Stack Overflow Developer Survey*. URL: https://survey.stackoverflow.co/2024/technology (vaadatud 18.05.2025).

[34] Raymond Hill. *Gorhill/Publicsuffixlist.Js*. 23. aprill 2025. URL: https://github.com/gorhill/publicsuffixlist.js (vaadatud 18.05.2025).

[35] William Harrison. *Wdhdev/Psl-Formatted*. 18. mai 2025. URL: https://github.com/wdhdev/psl-formatted (vaadatud 18.05.2025).

[36] *Think Aloud Protocol*. Teoses: *Wikipedia*. 19. märts 2025. URL: https://en.wikipedia.org/w/index.php?title=Think_aloud_protocol&oldid=1281228576 (vaadatud 14.05.2025).

[37] *Strict-Transport-Security - HTTP | MDN*. 13. märts 2025. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Strict-Transport-Security (vaadatud 17.05.2025).

[38] *Cookies Having Independent Partitioned State (CHIPS) | Google Privacy Sandbox*. URL: https://privacysandbox.google.com/cookies/chips (vaadatud 05.05.2025).

[39] *Commits · Publicsuffix/List*. GitHub. URL: https://github.com/publicsuffix/list/commits/main/?since=2015-01-01&until=2015-06-30 (vaadatud 17.05.2025).

[40] Haridus- ja Teadusministeerium. *EENet » Domeeniteenus*. URL: https://www.eenet.ee/EENet/domeeniteenus.html (vaadatud 04.05.2025).

[41] WHATWG. *HTML Standard - Sandboxed Flag*. URL: https://html.spec.whatwg.org/#sandboxed-origin-browsing-context-flag (vaadatud 14.05.2025).

[42] Paul Vorbach. *Npm-Stat: Psl*. URL: https://npm-stat.com/charts.html?package=psl&from=2025-04-01&to=2025-04-30 (vaadatud 14.05.2025).

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

Mina, Uku Sõrmus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Veebiküpsistega seotud ründeskoopide visualiseerimine", mille juhendajad on Sten Mäses ja Elar Lang
    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.

04.06.2025

[1] 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 – Näide TypeScript'i kasutamisest küpsise seadete esitamiseks

TypeScript'i kasutamine küpsise seadete esitamiseks TypeScript'i tüübina, et aidata vältida koodis olukordade tekitamist, kus luuakse `__Host-` või `__Secure-` eesliitega küpsis, mille vastavad eeltingimused ei ole täidetud.

```
interface BaseCookieSettings {
    baseName: string;
    path: string;
    isSecure: boolean;
    domain: string | null;
}

export type CookieSettings =
| (BaseCookieSettings & {
    prefix: "__Host-";
    isSecure: true;
    path: "/";
    domain: null;
})
| (BaseCookieSettings & {
    prefix: "__Secure-";
    isSecure: true;
})
| (BaseCookieSettings & {
    prefix: null;
});
```

Lisa 3 – Töö käigus tuvastatud vead välistes osapooltes

Töö käigus teemadesse süübides avastati autori poolt mitu viga välistes osapooltes, mis ka vastavalt raporteeriti. Vead võib üldjoones liigitada kas turvaprobleemideks või võimalikeks tehnilise sõnastuse parendusteks.

Firefox ja Chromium

Põhiliselt testitud brauserites, Mozilla Firefox'is ja Google Chromium'is (mis on aluseks Google Chrome'ile, Microsoft Edge'ile jne), leiti mitu küpsistega seotud probleemi, mis omavad ka mõju turvalisusele (konfidentsiaalsus, terviklus). Lõputöö esitamise ajal pole veel siin enumereeritud turvaraportid avalikud, kuid tavaliselt avalikustatakse raportid peale probleemide parandamist ja väikest ajapuhvrit.

Firefox'ist leiti ka üks kõrvalekalle tulevasest RFC6265bis spetsifikatsioonist ja Chromium'i käitumisest, mis *ei oma* turvamõju ja on seega avalik:
`https://bugzilla.mozilla.org/show_bug.cgi?id=1964920`

Turvaprobleemid, mis esinesid ainult Firefox'is (töö esitamise ajaks parandatud):

- `https://bugzilla.mozilla.org/show_bug.cgi?id=1951746`
- `https://bugzilla.mozilla.org/show_bug.cgi?id=1951750`
- `https://bugzilla.mozilla.org/show_bug.cgi?id=1964216`

Seni kehtivad turvamõjuga probleemid, mis esinevad nii Firefox'is kui Chromium'is ja võib lugeda ka möödapanekuteks spetsifikatsiooni tasandil (esimesed kaks on iseseisvalt taasavastatud leiud, mida on varem märgatud ja/või mainitud, kuid pole enamasti raporteeritud):

- `https://issues.chromium.org/issues/415981645` ja
  `https://bugzilla.mozilla.org/show_bug.cgi?id=1964767`
- `https://issues.chromium.org/issues/416099594` ja

https://bugzilla.mozilla.org/show_bug.cgi?id=1964945

- https://issues.chromium.org/issues/416196166 ja
  https://bugzilla.mozilla.org/show_bug.cgi?id=1966397

Haridus- ja Teadusministeerium

Public Suffix List'i all eesti domeene lähemalt uurides tuvastati, et nimekirjas olevad domeenid jaotuvad teatud üksikute asutuste vahel. Haridus- ja Teadusministeeriumi (EENet) alla kuuluvad nimekirjas mitu domeeni: `edu.ee`, `lib.ee` ja `org.ee`.

Leides vastava EENeti lehekülje [40], kus nende domeenide alla asutustele registreerimist pakutakse, on näha, et PSL-ist on aga üks pakutav domeen puudu. Tsitaat leheküljelt: *"Akadeemiliste domeenide `edu.ee`, `lib.ee`, `org.ee` ja `vil.ee` registreerimine"*.

See tähendab, et kõik `vil.ee` alla kuuluvad üksteisest sõltumatud asutused (enamasti erinevad koolid, nt `vjk.vil.ee` ja `abjag.vil.ee`) arvestatakse brauseri turvapoliitikas küpsiste mõttes ühe (skeemita) saidi alla – kehtivaks tippdomeeniks loetakse `ee` ja registreeritavaks domeeniks `vil.ee`. Seda vastupidiselt PSL-i kuuluva `edu.ee` puhul, mis tõttu on `pg.edu.ee` ja `www.voru.edu.ee` eraldi saidid, mille tippdomeen on mõlemal `edu.ee` ja registreeritavad domeenid vastavalt `pg.edu.ee` ja `voru.edu.ee`. Vaata ka joonist 17.

Joonis 17. Firefox toob aadressiribal registreeritava domeeni eraldi esile: (a) `pg.edu.ee` puhul on selleks `pg.edu.ee`, (b) `www.voru.edu.ee` puhul `voru.edu.ee`, (c) `vjk.vil.ee` puhul `vil.ee`, (d) `abjag.vil.ee` puhul samuti `vil.ee`.

Mõju mõttes tähendab see, et kui ühe asutuse leheküljel leidub näiteks XSS-tüüpi viga (või on lehekülg pahatahtlik või kompromiteeritud) ja ohver seda lehekülge külastab, saab see külastus omada ohvri kaudu ja ohvri jaoks mõju ka teistele suvalistele asutustele `vil.ee` alamdomeenidel. Mõju võib tähendada päringuvõltsingu rünnete (CSRF) lihtsustamist,

küpsisepommi kirjutamist (käideldavuse kadu) või isegi konfidentsiaalsuse kadu, kui mõni küpsis peaks olema kirjutatud atribuudiga `Domain=vil.ee`.

Loodud tööriistaga saab täpsemalt küpsistega võimalikke toimingute skoope näha siit: `https://ukusormus.github.io/cookies/site-origin-cookie-scopes-visualizer/#target=https%253A%252F%252Fabjag.vil.ee%252F&attacker=https%253A%252F%252Fvjk.vil.ee%252F`

Ründe tõenäosust võib hinnata aga pigem madalapoolseks, sest `vil.ee` all registreeritud domeene on üsna vähe (esmaste veebiotsingute järgi kümne, maksimaalselt paarikümne ringis), leheküljed ei ole tõenäoliselt kriitilise tähtsusega (enamasti Viljandimaal tegutsevad koolid) ning ründaja motivatsiooni võiks lugeda tunnetuslikult pigem madalaks, meelitada ohver külastama ühte `vil.ee` all olevat domeeni, et mõjutada ohvri jaoks mõnda teist, kuhu ohver on juba autenditud ja/või mida veel plaanib kasutada. Eeltingimuseks on ka juba olemasolev probleem, mille kaudu ründaja saab JavaScripti koodi käivitada või serverist HTTP vastusepäiseid tagastada.

Emaili teel `turvas@eenet.ee` saadi vastavate kontaktidega ühendust. Hetkel teadaoleva info põhjal kaalutakse prioriteete määravate aspektide (kasutajaskond, sensitiivne info, ajakriitilisus jm) võimalusena lahendada olukord ka lihtsa teavitusena klientidele, andes soovi korral võimaluse vahetada `vil.ee` näiteks `edu.ee` või `ee` domeeni vastu.

Vue playground

Töö käigus küpsisepommi (ingl *cookie bomb*) ründetehnika kohta uurides avastati, et Vue JavaScripti raamistiku kodulehe alamdomeenil `play.vuejs.org` on nn mänguväljak, kus saab Vue raamistikuga programmeerimist järele proovida. Keskkond pakub võimalust proovitud koodi lingina jagada (lisatakse URL-i *fragment* osasse kokkupakitult ja base64-kodeeritult); lingi avamisel käivitub saaja brauseris lingis sisalduv JavaScripti kood koheselt. Tuvastati, et `vuejs.org` ei kuulu Public Suffix List alla ja puuduvad muud piirangud, mis takistaks kirjutamast ohvri brauserisse küpsiseid `**.vuejs.org:*` skoopi ehk võimalik on tekitada brauseripoolne teenusetõkestus igaühele, kes ründaja konstrueeritud linki külastab. Muid võimalikke ründevektoreid (näiteks võimalust päringuvõltsingu rünneteks) pole siinkohal selle saidi raames uuritud.

Vue kodulehelt leitud turvakontaktile `security@vuejs.org` raporteeriti probleemist 19. märtsil ja saadeti korduskirjad 24. märtsil ja 10. aprillil, kuid ühelegi kirjale kolmest vastust ei saadud.

Lühidalt, kontseptsioonitõestus:

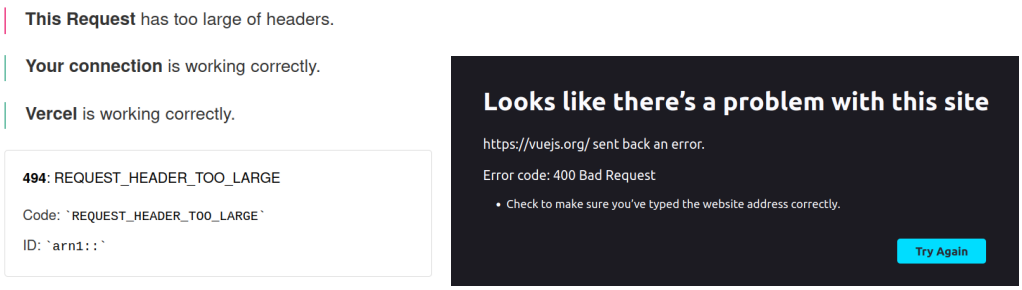
1. Avada brauseris uus inkognito aken (soovituslik samm testijale – nii on lihtsam hiljem mitte muretseda küpsiste kustutamise pärast).
2. Külastada järgmist URL-i:
   `https://play.vuejs.org/#eNp9kE9PwkAQxb/KZOMBArYl4kFKSVA5aKIS9b gHmmUoW9rdzf7Bmqbf3W0b0IPhtJP33mZ+b2qyVCo4OiQzMjdMc2XBoHVqQcVO ahgUaIFDAlHsn3kCk6idRqMh1FQAbCVzJQobMCkPHH1wc1XzJrmqKVlSEmhUmN rBNIqiYRPDOrX7JIzhUZYpF4nfm5tA6iyGl7S6XmaY3EXRJgYIQ2jXS1F8w+QW Si6cRTOG3BnPI4ClBqloqJiHPfWCjIk1TIodz4LcSOELdYSUMFkqXqB+U5ZLYS iZ9eytlxaF/HruNKsdjk862yM7/KPnpmo1StYaDeojUnL2bKoztL29+njFys9n s5RbV/j0BfMdjSxcy9jH7p3Yeuw/uY72qVRSWy6yT7OqLApzKtWCtsmmy1Pir/ twofov7k0w7f75g5LmB9KCqf8=`
3. Märgata, et kõik `vuejs.org` alla kuuluvad domeenid annavad neid külastades veateate (vaata joonis 18), muuhulgas `play.vuejs.org` ise, `vuejs.org`, `blog.vuejs.org` ja `news.vuejs.org`.

Käivitatav kood:

```
<script setup>
for (let i = 0; i <= 100; i++) {
  document.cookie = `${i}=${"A".repeat(4000)}; Path=/; Domain=vuejs.org
      ; Max-Age=900`;  // for only 15 minutes, just in case
}
</script>
```

Võimalikeks lahendusteks probleemile pakuti saadetud e-kirjas muuhulgas välja:

- Luua eraldiseisev domeen (stiilis `vueplayground.example`), et mänguväljakul käivitatav kood ei saaks teisi `vuejs.org` alamdomeene mõjutada ja lisada praeguselt domeenilt alaline suunamine uuel lehele.
- Lisada `vuejs.org` PSL-i.



(a) (b)

Joonis 18. Brauseris avanev vaade pärast seda, kui ohver on ründaja lingi avanud: (a) `blog.vuejs.org`, (b) `vuejs.org` külastamisel.

- Kasutada HTTP `Content-Security-Policy` turvapäist väärtusega `sandbox allow-scripts`. Väärtuses `sandbox` kasutamine loob brauserites unikaalse `"null" origin`'i, millelt ei tohiks saada HTML'i standardi järgi küpsistega interakteeruda [41]; seda kinnitab ka esmane testimine, Firefox'is veateate kujul *"Document.cookie setter: Forbidden in a sandboxed document without the 'allow-same-origin' flag"* (analoogne veateade Chrome'is).

Muud ettepanekud

Olgu loetletud mõned muud välistele osapooltele uurimistöö perioodil raporteeritud ettepanekud.

Märgati, et inglisekeelses Vikipeedias (Wikipedia) on veebiküpsistega seotud artiklis juba esimeses lõigus võimalusi tehnilist sõnastust muuta täpsemaks. Artikli aruteluleheküljele loodi vastavad teemad:

- `https://en.wikipedia.org/wiki/Talk:HTTP_cookie#c-UkuSormus-20250410053800-%22Created_by_a_web_server%22_in_the_first_paragraph_may_not_always_be_correct`
- `https://en.wikipedia.org/wiki/Talk:HTTP_cookie#c-UkuSormus-20250410054000-%22user's_web_browser%22_v.s._other_types_of_client`
- `https://en.wikipedia.org/wiki/Talk:HTTP_cookie#c-UkuSormus-20250410054100-New_browser-side_CookieStore_API`

Public Suffix List (PSL) koodihoidla all:

- Pakuti välja vajadus standardimplementatsiooniks:
  `https://github.com/publicsuffix/list/issues/2443`
- Märgati väikest tehnilist viga esimeses lauses, mis esines nii `publicsuffix.org` veebilehel kui PSL koodihoidla `README` failis üle kümne aasta (parandatud):
  `https://github.com/publicsuffix/list/issues/2451`

Node.js `psl` pakk, millel on `npm-stat.com` andmetel ainuüksi 2025. aaasta aprillikuus ligi 136 miljonit allalaadimist [42], ei käitu vastavalt PSL algoritmile punktiga lõppevate domeenide osas:
`https://github.com/lupomontero/psl/issues/350`

WHATWG URL standardi kohta leiti, et selgemini saaks sõnastada `host` ja `hostname` mõisteid ning vastavaid URL API funktsioone; lisaks leidub võimalik vasturääkivus domeeninimede parsimise osas:

- `https://github.com/whatwg/url/issues/869`
- `https://github.com/whatwg/url/issues/870`
- `https://github.com/whatwg/url/issues/871`