VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdai

Web Scraping Prevention Methods

Bakalauro baigiamasis darbas

Atliko:	Domantas Slėnys	(parašas)
Darbo vadovas:	lekt. Aurimas Šimkus	(parašas)
Darbo recenzentas:	lekt. Karolis Uosis	(parašas)

Vilnius – 2022

Santrauka

Šiame darbe analizuojamos automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdų galimybės. Probleminė sritis yra skirstoma į penkias dalis. Pirmiausiai, apibrėžiama interneto roboto sąvoka, akcentuojant skirtumus tarp saityno peržiūros roboto (angl. *web crawler*) ir saityno duomenų ištraukėjo (angl. *web scraper*). Toliau, pagal apibrėžtas interneto robotų savybes, įvertinamos egzistuojančios interneto robotų valdymo priemonės. Skyriaus rezultatai apima išskirtus kriterijus bei sprendžiamas problemas, siekiant praktiškai ir efektyviai taikyti prevenciją. Po kriterijų išskyrimo pateikiama prevencijos samprata, gilinamasi į saityno duomenų ištraukėjo veikimo principus ir atliekama teorinė analizė pasirinktų prevencijos būdų. Nustatyta, kad padengti visus duomenų ištraukimo etapus, prevencijos būdai turi būti kombinuojami. Siekiant įvertinti prevencijos būdų galimybes ir iš praktinės pusės, analizuojamos egzistuojančios komercinės prevencijos priemonės. Galiausiai, apibrėžiamos automatizuoto duomenų surinkimo prevencijos prototipo įgyvendinimo detalės, integruojant vieną iš pasirinktų komercinių prevencijos priemonių. Šiuo atveju tikslas yra padengti visus duomenų ištraukimo etapus ir parodyti, kad prevencija gali būti praktiškai taikoma kiekviename iš analizuojamų etapų. Darbo rezultatai apima atliktus palyginimus bei įgyvendintą automatizuoto duomenų surinkimo prevencijos prototipą. Darbo metu nustatyta, kad prevencija gali būti praktiškai taikoma prieš neetiškus interneto robotus.

Raktiniai žodžiai: automatizuoto duomenų surinkimo prevencija, neetiški interneto robotai, paskirstytas interneto robotų tinklas, besisukantys įgaliotieji serveriai, saityno duomenų ištraukėjas.

Summary

This paper analyzes the possibilities of prevention against website scraping. The problem area is divided into five parts. First, the concept of an internet robot is defined by emphasizing the differences between a web crawler and a web scraper. Next, the existing control tools for internet robots are evaluated according to the characteristics defined during internet robots comparison. The results of the chapter include the identified criterias and the problems to be solved in order to practically and effectively apply prevention. After the definition of criterias the concept of prevention is presented, the working principles of a web scraper are explored and the theoretical analysis of the selected prevention methods is performed. It has been found that to cover all stages of web scraping, prevention techniques need to be combined. Existing commercial prevention measures are analyzed in order to assess the practical possibilities of prevention. Finally, the details of implementing a web scraping prevention prototype by integrating one of the selected commercial prevention measures are defined. In this case, the aim is to cover all stages of web scraping and to show that prevention can be applied to each of the stages analyzed practically. The results of the work include the comparisons made and the implementation of a prototype for the web scraping prevention. In the work it is concluded that prevention can be practically applied against unethical internet robots.

Keywords: web scraping prevention, bad web robots, distributed web robot networks, rotating proxy, web scraper.

TURINYS

ĮVADAS 6
1. Automatizuoto duomenų surinkimo iš tinklalapių būdai 8
1.1. Saityno peržiūros robotas 8
1.2. Saityno duomenų ištraukėjas 9
1.3. Palyginimas tarp saityno peržiūros roboto ir duomenų ištraukėjo 10
1.4. Apibendrinimas 10
2. Prevencijos kriterijai ir sprendžiamos problemos 12
2.1. Interneto robotų valdymas 12
2.2. Paskirstyto duomenų surinkimo keliama problema 12
2.3. Duomenų ištraukimo etiškumas 13
2.4. Apibendrinimas 14
3. Automatizuoto duomenų surinkimo prevencija 15
3.1. Prevencijos samprata 15
3.2. Saityno duomenų ištraukėjo architektūros komponentai 15
3.2.1. Interneto naršyklės valdymo sąsaja 15
3.2.2. Įgaliotasis serveris 16
3.2.3. HTML analizatorius 17
3.3. Prevencijos būdų palyginimo kriterijai 17
3.4. Prevencijos būdų analizė 18
3.4.1. Trijų sąrašų filtravimo technika 18
3.4.2. „Honeypot“ 19
3.4.3. Užklausos šaltinio indikatorių analizė 20
3.4.4. Atsitiktinis tinklalapių turinio atkūrimas 21
3.5. Prevencijos būdų palyginimas 23
3.6. Apibendrinimas 24
4. Komercinės automatizuoto duomenų surinkimo prevencijos priemonės 25
4.1. Pasirinkimo kriterijai 25
4.2. „DataDome“ 25
4.3. „PerimeterX“ 27
4.4. „Kasada“ 28
4.5. Prevencijos priemonių palyginimas 29

4.6. Apibendrinimas 30
5. Automatizuoto duomenų surinkimo prevencijos prototipas 31
5.1. Prototipo veikimo principas 31
5.2. HTML turinio ir pakopinių stilių šablono apdorojimas 32
5.2.1. HTTP užklausų vykdymo pagrindas 32
5.2.2. Abstrakčios sintaksės medžio sudarymas 32
5.2.3. Atsitiktinių reikšmių generavimas ir kodavimas 33
5.2.4. HTML dokumento susiejimas su įterptais resursais 34
5.2.5. Pakopinių stilių šablono modifikavimas 35
5.3. Integravimo ir plečiamumo galimybių įvertinimas 36
5.4. Naudojamos priemonės ir bibliotekos 37
5.5. Apibendrinimas 38
REZULTATAI IR IŠVADOS 39
TOLIMESNI DARBAI 41
ŠALTINIAI 42
SAVOKŲ APIBRĖŽIMAI 46

ĮVADAS

Duomenų perteklius bei ribotas vartotojų laikas šiais laikais sudaro tendenciją, kad dažniausiai aplankomos interneto svetainės yra tos, kurios laiku ir tikslingai pateikia aktualiausią turinį vartotojui. Vartotojo veiksmus, renkantis interneto svetainę, galima paaiškinti per informacijos ieškojimo teoriją (angl. *information-foraging theory*). Teorija teigia – jeigu žmogus turi klausimą, į kurią interneto svetainę eiti, tai jis nuspręs įvertinęs, kaip tikėtina, ar tinklalapis pateiks atsakymą į klausimą bei kaip ilgai užtruks tą atsakymą rasti [Bud20]. Vadinasi, ar tai būtų turinio originalumas, ar poreikis iš vartotojo pusės, daugybė interneto svetainių turi tarpusavyje konkuruoti su tikslu užtikrinti pranašumą. Vienas iš tokių būdų – surinkti aktualius duomenis iš panašaus turinio svetainių, neretai agreguojant tai į bendrą visumą. Nors tradiciniai būdai kaip iškirpti ir įklijuoti (angl. *cut&paste*) yra galimi ir naudojami, tačiau siekiant efektyviai analizuoti didelius duomenų srautus atsiranda poreikis surinkimo procesą automatizuoti. Pavyzdžiui, elektroninės komercijos srityje sekti konkurentų kainas realiu laiku. Šiam tikslui pasiekti kuriami komerciniai įrankiai, kurie specializuojasi automatizuoto duomenų surinkimo iš tinklalapių srityje.

Bendru atveju galima išskirti du automatizuoto duomenų surinkimo būdus: saityno peržiūros robotas (angl. *web crawler*) bei saityno duomenų ištraukėjas (angl. *web scraper*). Abu interneto robotai automatiškai vykdo užklausas į interneto svetainės serverį, apdoroja bei analizuoja gautus duomenis daug greičiau nei renkant rankiniu būdu. Šiuo atveju toks interneto svetainės serverio apkrovimas gali nenumatytai sukelti šalutinius poveikius kaip kaštų išaugimą, pavyzdžiui, kai naudojamos išorinės paslaugos, kurios neriboja skaičiavimo resursų. Tai nėra retenybė, kadangi, pagal 2020 metų kibernetinio saugumo įmonės „Imperva“ statistiką, ketvirtadalis viso interneto srauto yra žalingi robotai [Has21]. Be to, kai kurioms svetainėms, stipriai priklausančioms nuo savo patentuoto turinio, poreikis išlikti konkurencingiems verčia susimąstyti apie galimus prevencijos būdus tokiai interneto robotų veiklai. Taigi, siekiant tikslingai taikyti prevenciją, būtinos priemonės, kurios leistų atskirti tikrus vartotojus nuo interneto robotų.

Šiame darbe atliekama mokslinės literatūros ir lyginamoji analizė su tikslu įvertinti automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdų taikymo galimybes tiek iš teorinės, tiek iš praktinės pusės. Papildomai praktine dalimi siekiama įvertinti prevencijos būdų pritaikymo ribas, pateikiant ir įgyvendinant duomenų surinkimo prevencijos prototipą. Taip pat atsižvelgiama į darbo prielaidą, kad komercinių prevencijos priemonių įgyvendinimo detalės yra slepiamos, todėl kartu siekiama ir įvertinti jų plečiamumo galimybes integruojant į prototipą.

Darbo tikslas – išnagrinėti skirtingus automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdus ir įvertinti jų taikymo galimybes.

Tikslui pasiekti keliami uždaviniai:

1. Apžvelgti automatizuoto duomenų surinkimo sampratą, būdus, jų paskirtį bei veikimą.
2. Identifikuoti esminius kriterijus ir sprendžiamas problemas prevencijos taikymui analizuojant egzistuojančias interneto robotų valdymo priemones.
3. Apžvelgti ir pagal pasirinktus kriterijus palyginti automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdus.
4. Apžvelgti ir pagal pasirinktus kriterijus palyginti egzistuojančias komercines automatizuoto duomenų surinkimo prevencijos priemones.
5. Išnagrinėjus komercines prevencijos priemones, įvertinti ir praktiškai parodyti jų plečiamumo galimybes, įgyvendinant nagrinėtus prevencijos būdus.

1. Automatizuoto duomenų surinkimo iš tinklalapių būdai

Šiame skyriuje bus apžvelgiami skirtingi automatizuoto duomenų surinkimo iš tinklalapių būdai, kaip saityno peržiūros robotas bei saityno duomenų ištraukėjas. Skyrius skirtas įsigilinti į probleminę sritį, pateikiant įvardintų būdų apibrėžimus, skirtumus bei panašumus.

1.1. Saityno peržiūros robotas

Saityno peržiūros robotas (angl. *web crawler*), bendriau apibrėžiamas kaip interneto robotas, yra terminas, apibūdinantis bet kokio tipo programą, kuri automatiškai ir rekursyviai keliauja per interneto svetaines [Mck16]. Rekursyviam keliavimui peržiūros robotas naudoja HTML[1] nuorodas, apibrėžtas tinklalapio turinyje. Siekiant padengti visą interneto svetainės turinį, peržiūros robotas turi aplankyti visas pasiekiamas nuorodas pradedant nuo pasirinkto taško (tinklalapio). Dažnai nuorodos gali būti ciklinio pobūdžio ar nuvesti į išorines svetaines, todėl atsiranda poreikis saugoti aplankytų ir neaplankytų peržiūrimos svetainės tinklalapių sąrašą su tikslu terminuotai užbaigti darbą. Taigi, tinklalapių paieškos operaciją galima apibendrinti kaip procesą pereinantį orientuotą grafą su sąlyga, kad tinklalapis yra laikomas mazgu ir nuoroda – briauna [UKD14]. Pradinis svetainės tinklalapių sąrašas susideda iš pasirinkto vieno arba kelių tinklalapių URL[2] adresų. Pasirinktu adresu peržiūros robotas vykdo HTTP[3] užklausą, išsaugo gautą HTML dokumentą ir iš dokumento įtraukia į tinklalapių sąrašą dar neaplankytų nuorodų adresus. Adresai gali būti reliatyvūs, todėl, norint užtikrinti tinklalapių sąrašo unikalumą, reliatyvūs adresai verčiami absoliučiais, įtraukiant domeno vardą. Užklausos vykdymo ir URL adresų išgavimas tęsiasi tol, kol tinklalapių sąrašas nėra tuščias arba kažkokia kita sąlyga priverčia procesą sustoti [UKD14].

Vienas iš pagrindinių saityno peržiūros roboto panaudos atvejų yra pritaikymas paieškos varikliuose. Kaip pavyzdį galima pateikti „Googlebot“ peržiūros robotą, kuris yra skirtas indeksuoti interneto svetaines. Šio roboto tikslas – peržiūrėti kuo daugiau interneto svetainės tinklalapių, neviršijant serverio numatyto tinklo pralaidumo [Goo21a]. „Googlebot“ gali vykdyti „JavaScript“ kodą, tačiau ne visada tai atlieka iš karto, kadangi atvaizdavimo etapas yra brangus vykdymo laiko atžvilgiu [Rog19]. Surinkti duomenys toliau „Google“ serveriuose apdorojami ir agreguojami su kitomis interneto svetainėmis tam, kad indeksuotos svetainės būtų tikslingai pasiekiamos, vykdant

[1] HTML – hiperteksto ženklinimo kalba (angl. *HyperText Markup Language*)

[2] URL – universalus ištekliaus adresas (angl. *Uniform Resource Locator*)

[3] HTTP – hipertekstų siuntimo protokolas saityno duomenims (angl. *HyperText Transfer Protocol*)

paiešką atitinkamoje sistemoje, šiuo atveju, „Google Search“. Vadinasi, peržiūros robotas turi būti pritaikytas apdoroti didžiulius kiekius informacijos efektyviai. Pabrėžiama, kad prieš pradedant vykdyti užklausą į bet kokį interneto svetainės tinklalapį, peržiūros robotas turi save identifikuoti kartu su užklausa pateikiant vartotojo agento eilutę (angl. *user agent*). Atitinkamai vartotojo agento eilutė gali būti naudojama „robots.txt“ faile tam, kad apibrėžti peržiūros taisykles specifiniams interneto robotams [Goo21b] (plačiau apie „robots.txt“ failą rašoma 2.1 poskyryje).

Googlebot Desktop	Googlebot
Pilna vartotojo agento eilutė	• `Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)` • `Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; Googlebot/2.1; +http://www.google.com/bot.html) Chrome/W.X.Y.Z Safari/537.36` • `Googlebot/2.1 (+http://www.google.com/bot.html)`

1 Pav. „Googlebot“ peržiūros roboto pateikiama vartotojo agento eilutė [Goo21b]

1.2. Saityno duomenų ištraukėjas

Skirtumas tarp duomenų ištraukimo (angl. *web scraping*) ir duomenų peržiūros (angl. *web crawling*) yra didžiąją dalimi neaiškus, todėl daugelis autorių bei programuotojų abu terminus vartoja pakaitomis [BB18]. Saityno duomenų ištraukėjas yra panašus į peržiūros robotą tuo, kad sutampa abiejų sistemų pirmasis vykdomas žingsnis – duomenų gavyba. Bendrais bruožais peržiūros roboto veikimas sudaro dalį saityno duomenų ištraukėjo ir atvirkščiai. Tačiau skirtumas išryškėja apibrėžiant abiejų sistemų tikslus. Peržiūros robotas siekia padengti visą ar net kelias interneto svetaines, retai kada atsižvelgiant į tinklalapių turinį (turinio aktualumą), kai duomenų ištraukėjas atvirkščiai orientuotas į aiškiai apibrėžtą interesų sritį – ar tai būtų internetinės parduotuvės katalogas, ar eilė straipsnių iš naujienų svetainių. Taigi, saityno duomenų ištraukėjas yra suprojektuotas išgauti nestruktūrizuotus duomenis iš interneto taip, kad būtų galima juos išsaugoti ir įvertinti [Mck16].

Norint ištraukti dominantį turinį, tuo pačiu vengiant nereikalingos informacijos, ištraukėjas turi būti iš anksto sukonfigūruotas. Vadinasi, žinios apie interneto svetainės turinio organizavimą lemia duomenų ištraukėjo tikslumą. Neretai visą duomenų ištraukimo procesą siekiama automatizuoti, todėl struktūros žinojimas yra itin svarbi duomenų ištraukimo dalis. Priklausomai nuo pasirinktos interneto svetainės, priešingai nei peržiūros robotas, duomenų ištraukėjas neišvengiamai turės vykdyti svetainės skriptus. To priežastis – dinaminės interneto svetainės. Dinaminėje interneto svetainėje pateikiama informacija keičiasi priklausomai nuo stebinčiojo vartotojo, paros laiko, laiko juostos, vartotojo

gimtosios kalbos ir kitų veiksnių [Hop20a]. Todėl atsiranda poreikis imituoti vartotojo veiklą. Šiam poreikiui pasiekti saityno duomenų ištraukėjai identifikuojasi kaip interneto naršyklės ir neretai naudoja papildomas paslaugas kaip įgaliotuosius serverius (angl. *proxy server*). HTTP užklausos, vykdomos per įgaliotuosius serverius, gali būti užmaskuojamos įgaliotojo serverio IP adresu (plačiau apie įgaliotuosius serverius rašoma 3.2.2 punkte).

1.3. Palyginimas tarp saityno peržiūros roboto ir duomenų ištraukėjo

Šiame poskyryje siekiama apibendrinti nagrinėtus automatizuoto duomenų surinkimo iš tinklalapių būdus, pabrėžiant tarpusavio skirtumus bei panašumus.

1 lentelė. Automatizuoto duomenų surinkimo iš tinklalapių būdų palyginimas

Saityno peržiūros robotas	Saityno duomenų ištraukėjas
Saityno peržiūra dažniausiai vykdoma dideliu mastu	Duomenų surinkimas gali būti vykdomas bet kokiu mastu
Siekia padengti visą ar net kelias interneto svetaines	Orientuotas į aiškiai apibrėžtą interesų sritį
Išsaugomi gauti HTML dokumentai dažniausiai indeksavimo tikslais	Išgaunami nestruktūrizuoti duomenys saugojimo ir įvertinimo tikslais
Aktualu išvengti duomenų dublikavimo	Į duomenų dublikavimą nėra atsižvelgiama
Atsižvelgiama į poveikį tinklo pralaidumui	Nebūtinai atsižvelgiama į poveikį tinklo pralaidumui
Identifikuojasi pateikdamas atpažįstamo interneto roboto vartotojo agento eilutę	Imituoja tikro vartotojo veiklą pateikdamas interneto naršyklės vartotojo agento eilutę
Iš anksto žinoma tik apie interneto svetainės nuorodas (pradinį tašką peržiūrai)	Iš anksto žinoma apie interneto svetainės turinio organizavimą
Svetainės skriptai vykdomi ribotai, t. y. orientuojamasi į našumą	Dažniausiai būtina vykdyti svetainės skriptus tinklalapio turiniui pasiekti
HTTP užklausos vykdomos iš atitinkamo peržiūros roboto serverio	Naudojasi įgaliotaisiais serveriais įvykdyti HTTP užklausai

1.4. Apibendrinimas

Apžvelgus skirtingus automatizuoto duomenų surinkimo iš tinklalapių būdus matoma aiški atskirtis tarp duomenų ištraukėjo ir peržiūros roboto. Nors abu interneto robotai identifikuoja save

vykdant užklausą į interneto svetainės serverį, pastebima, kad skirtumas tarp duomenų ištraukėjo ir tikro interneto vartotojo yra gana nežymus, t. y. siekiama imituoti vartotojo veiklą. Nenumatytais atvejais interneto svetainės serveris neturi galimybės suvaldyti roboto, kurio negali identifikuoti. Vadinasi, taikyti prevencijai, būtina žinoti esminius kriterijus bei sprendžiamas problemas su tikslu praktiškai ir efektyviai suvaldyti interneto robotus.

2. Prevencijos kriterijai ir sprendžiamos problemos

Šiame skyriuje bus nagrinėjamos egzistuojančios interneto robotų valdymo priemonės. Skyriaus tikslas yra identifikuoti esminius kriterijus ir sprendžiamas problemas, siekiant praktiškai ir efektyviai taikyti prevenciją interneto robotams.

2.1. Interneto robotų valdymas

Siekiant užtikrinti, kad duomenų surinkimas neapkrautų interneto svetainės serverio, svetainės savininkas gali šakniniame kataloge apibrėžti tekstinį failą, pavadinimu „robots.txt“. Šio failo turinys apibrėžiamas remiantis robotų išskyrimo protokolu. Protokolo tikslas yra pateikti instrukcijas apie interneto svetainę interneto robotams [Kos20]. Pavyzdžiui, „Disallow“ direktyva galima uždrausti aplankyti specifinį tinklalapį ar dalį interneto svetainės. Tačiau būtina atkreipti dėmesį į tai, kad tiek peržiūros robotai, tiek duomenų ištraukėjai gali ignoruoti „robots.txt“ failą [Kos20]. Todėl nagrinėjamus automatizuoto duomenų surinkimo iš tinklalapių būdus galima priskirti į vieną iš dviejų kategorijų: etiški arba žalingi. Šiuo atveju etiškais yra laikomi paieškos varikliai ar bet koks kitas interneto robotas, kuris laikosi robotų išskyrimo protokolo. Taigi, vienas iš kriterijų, kada atliekama žala interneto svetainės savininkui, t. y. turi būti taikoma prevencija, kai duomenys yra renkami iš uždraustų interneto svetainės dalių.

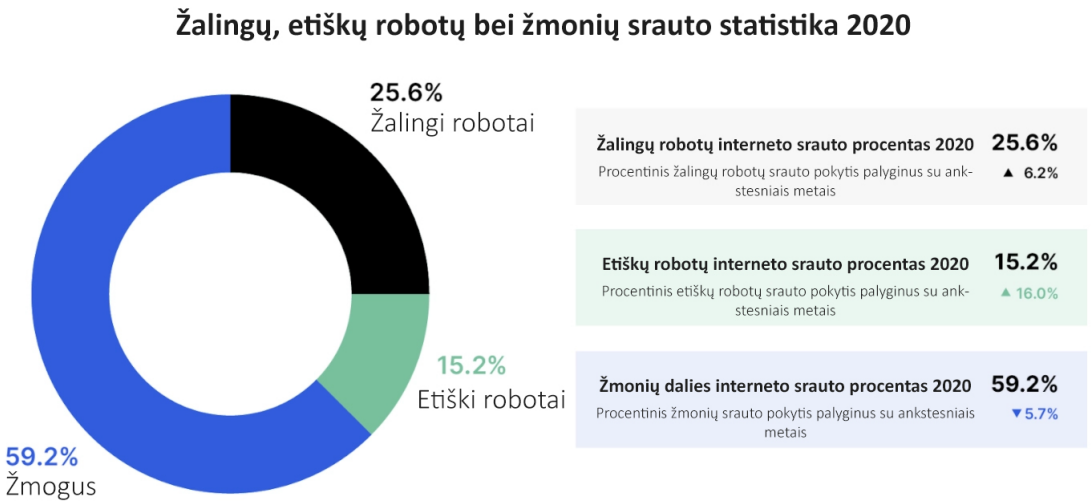
```
Sitemap: https://www.svetaine.pvz.lt/sitemap.xml
Sitemap: https://svetaine.pvz.lt/sitemap.xml

User-agent: *                    User-agent: Googlebot
Disallow: /tmp/                  Disallow: /neindeksuojamas/turinys
Disallow: /admin/
Disallow: /ajax/                 User-agent: Unwelcomebot
Disallow: /talpykla/             Disallow: /
Disallow: /aplankalas/
Disallow: /failas.html
Disallow: /nuotrauka.png
Disallow: /įdėtinis/aplankalas
```

2 Pav. „Disallow“ direktyvų pavyzdys

2.2. Paskirstyto duomenų surinkimo keliama problema

Atsižvelgiant į automatizuoto duomenų surinkimo prigimtį, HTTP užklausos į interneto svetainės serverį turi būti vykdomos efektyviai. Vienas iš būdų efektyviai vykdyti užklausas yra paskirstyti duomenų surinkimo procesą. Pavyzdžiui, paskirstytas peržiūros robotas yra suprojektuotas taip, kad panaudotų perteklinį tinklo mazgų pralaidumą ir skaičiavimo išteklius su tikslu peržiūrėti

pasirinktas interneto svetaines [TO04]. Vis dėlto, tokiu atveju atsiranda rizika, kad gali sumažėti interneto svetainės tinklalapių peržiūrėjimo intervalai, dėl ko bus sukelta DDoS[4] ataka, t. y. trikdoma prieiga įprastiems vartotojams. Taip pat tam nepadeda faktas, kad, pagal 2020 metų kibernetinio saugumo įmonės „Imperva“ statistiką, ketvirtadalis viso interneto srauto yra žalingi robotai [Has21]. Vienas iš galimų sprendimų būtų į 2.1 poskyryje minėtą „robots.txt“ failą įtraukti „Crawl-Delay“ direktyvą, tačiau ši direktyva yra nestandartinis protokolo praplėtimas, todėl ją kiekvienas interneto robotas gali interpretuoti dviprasmiškai. Pavyzdžiui, „Googlebot“ peržiūros robotas šios direktyvos nepalaiko. Taigi, atsiranda poreikis gebėti atskirti interneto robotų tinklus, nepaveikiant įprastų vartotojų srauto interneto svetainėje.



3 Pav. Žalingų, gerų robotų bei įprastų vartotojų srautas 2020 metais [Has21]

2.3. Duomenų ištraukimo etiškumas

Siekiant atsakyti į duomenų ištraukimo etiškumo klausimą, reikia gilintis į teisėtumo aspektą. Nors etika ir teisė yra skirtingi dalykai, tačiau viena kitą papildančios teisinės sistemos paprastai apima akivaizdžiausias etikos problemas, susijusias su konkrečia praktika [KJS20]. Viena iš teisių – deliktų teisė (angl. *Tort*) – jau buvo paminėta pavyzdyje 2.2 poskyryje, kai yra trikdoma prieiga įprastiems vartotojams. Šiuo atveju atsiranda atsakomybė atlyginti žalą. Tačiau kitos teisės, kaip naudojimo sąlygos, dažnai apibrėžiamos interneto svetainės poraštėje (angl. *footer*), ar bendresniu atveju autorių teisės, retai kada internete įrodo interneto svetainės savininko turinio nuosavybę. Pavyzdžiui, interneto svetainėje vartotojų sugeneruotas turinys nėra nuosavybės objektas, taip kaip nėra draudžiama iš autorių teisių saugomų duomenų kurti suvestines [KJS20]. Taigi, iki dabar nėra jokios konkrečios

[4] DDoS – paskirstyta paslaugos trikdymo ataka (angl. *Distributed Denial-of-Service*)

teisės, kuri būtų skirta duomenų ištraukimui, dėl ko šis procesas iš teisinės pusės laikomas „pilka“ zona.

Kitas būdas nagrinėti duomenų ištraukimo etiškumą yra detalizuojant veiksmus, kuriais siekiama išgauti dominantį turinį. Priešingai nei peržiūros robotas, duomenų ištraukėjas gali tiesiogiai vykdyti užklausas su pasirinktų parametrų kombinacijomis į interneto svetainės serverį. Šiuo atveju tikslas – filtruoti rezultatus. Pavyzdžiui, surinkti duomenis apie įmonę, jos darbuotojus ieškant profesiniuose socialiniuose tinkluose potencialių profilių, atitinkančių apibrėžtus kriterijus. Taip pat, kad pasiektų autorizacijos reikalaujantį turinį, duomenų ištraukėjas gali pateikti netikras formas bei pildyti suklastotus duomenis. Vadinasi, atsiranda poreikis gebėti apriboti interneto robotų veiksmus, kurie iš savininko pusės būtų laikomi neetiškais.

2.4. Apibendrinimas

Išnagrinėjus automatizuoto duomenų surinkimo probleminę sritį, matoma, kad tiek peržiūros robotai, tiek duomenų ištraukėjai gali būti neetiški. Atitinkamai žala interneto svetainės savininkui gali būti laikoma nauda duomenų surinkėjui. Įvertinta, kad iš teisinės pusės duomenų surinkimas nėra tiesiogiai reglamentuojamas. Standartinės interneto priemonės, kaip „robots.txt“ failas, negarantuoja, kad interneto svetainę aplankantys robotai jų laikysis. Vadinasi, atsiranda poreikis interneto svetainės savininkui užtikrinti, kad vertingas interneto svetainės turinys būtų apsaugomas, t. y. taikoma prevencija neetiškiems interneto robotams, kurie bando tokią informaciją išgauti ir panaudoti savais tikslais.

3. Automatizuoto duomenų surinkimo prevencija

Šiame skyriuje bus nagrinėjama, kas yra automatizuoto duomenų surinkimo iš tinklalapių prevencija: aprašomi saityno duomenų ištraukėjo architektūros komponentai, siekiant suprasti prevencijos įgyvendinimo etapus, bei apžvelgiami egzistuojantys prevencijos būdai. Skyriaus tikslas yra įvertinti bei palyginti išskirtus prevencijos būdus.

3.1. Prevencijos samprata

Šiame darbe prevencija yra formalios technikos, metodai, kombinuojami žingsniai, kuriais siekiama dalinai arba maksimaliai apsunkinti automatizuoto duomenų surinkimo procesą. Norint tai pasiekti nėra diferencijuojama tarp skirtingų duomenų surinkimo būdų; turi būti gilinamasi į prevencijos etapus pradedant nuo užklausų vykdymo iki interneto robotų elgesio analizės bei išgauto turinio apdorojimo sunkumo. Bendru atveju prevencijos tikslas yra apsunkinti interneto svetainės duomenų gavimo procesą skriptams bei interneto robotams, tuo pačiu užtikrinant prieigą tikriems vartotojams bei paieškos variklių naudojamiems peržiūros robotams [Jon20]. Pabrėžiama, kad prevencija turi būti taikoma tik neetiškiems interneto robotams.

3.2. Saityno duomenų ištraukėjo architektūros komponentai

Šiame poskyryje siekiama suprasti, kokie komponentai sudaro saityno duomenų ištraukėją. Kaip 1.2 poskyryje nustatyta, duomenų ištraukėjo ir peržiūros roboto pirmasis vykdomas žingsnis – duomenų gavyba – sutampa, tačiau duomenų ištraukėjas įgyvendina papildomą funkcionalumą, kuris sumažina skirtumą tarp tikro vartotojo ir duomenų ištraukėjo. Vadinasi, norint suprasti, kokiais etapais gali būti vykdoma prevencija, būtina turėti aiškų supratimą, kokias funkcijas individualūs komponentai atlieka.

3.2.1. Interneto naršyklės valdymo sąsaja

Interneto naršyklės valdymo sąsaja (angl. *web driver*) – yra nuotolinio valdymo sąsaja, kuri leidžia kontroliuoti vartotojo agentus (angl. *user agent*) bei atlikti introspekciją [Con21]. Introspekcija laikoma valdymo sąsajos galimybė stebėti apdorojamą HTML dokumentą iš tikro vartotojo perspektyvos. Tokiu būdu siekiama išspręsti dinaminės interneto svetainės problemą. Pavyzdžiui, užkraunant pakopinių stilių šabloną (angl. *CSS*[5]), „JavaScript“ failus bei kitus saityno duomenų

[5] CSS – pakopinių stilių šablonas (angl. *Cascading Style Sheets*)

šaltinius (angl. *web assets*). Dažniausiai interneto naršyklės valdymo sąsajos naudojamos automatizuotų testų vykdymui [Con21]. Taip pat valdymo sąsaja atlieka identifikavimosi funkciją (pateikia vartotojo agento eilutę). Palyginimui pateikiamos pavyzdinės įprastos interneto naršyklės ir interneto roboto vartotojo agentų eilutės:

```
1 Mozilla/5.0 (compatible; YandexAccessibilityBot/3.
  0; +http://yandex.com/bots)

1 Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/53
  7.36
```

4 Pav. „Yandex“ paieškos variklio roboto (viršuje) ir „Chrome“ interneto naršyklės vartotojo agento (apačioje) eilučių palyginimas [Moz21a]

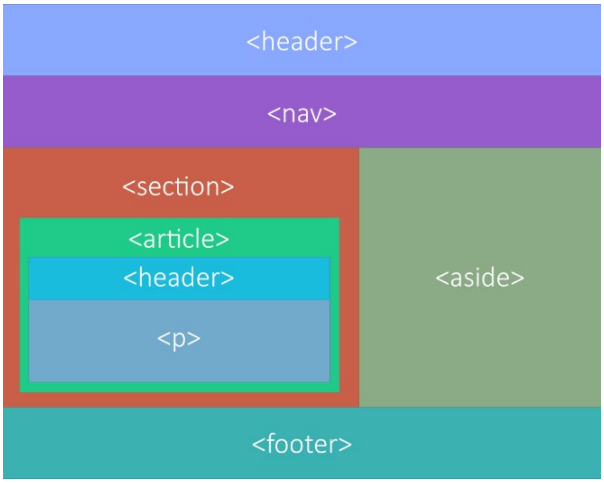
Matoma, kad interneto roboto vartotojo agento eilutėje, dviejose iš kabliataškiais atskirtų žymių, nurodomas žinomas interneto roboto vardas bei nuoroda į aprašymą, o interneto naršyklės eilutėje įvardinama platforma. Papildomai, sutampanti dalis parodo, kad abu, eilutes pateikę šaltiniai, įgyvendina nurodytos interneto naršyklės funkcionalumą. Kadangi valdymo sąsajos vienas iš tikslų yra imituoti tikrą vartotoją, identifikuojamasi ne kaip interneto robotas, bet kaip interneto naršyklė, t. y. neetiškai – prarandama pirminė galimybė apriboti leidžiamus veiksmus interneto robotams.

3.2.2. Įgaliotasis serveris

Neretai neužtenka tiesiog identifikuotis kaip interneto naršyklei siekiant efektyviai vykdyti duomenų ištraukimą. Problema iškyla, kai interneto svetainės serveriai tinklo paslaugų lygyje įgyvendina IP adreso blokavimo funkcionalumą. IP adreso blokavimo funkcionalumo tikslas yra kiekvienam svetainės lankytojui apibrėžti HTTP užklausų vykdymo normą, kurią peržengus prieiga arba dalinai, arba pilnai apribojama. Tam išvengti naudojamas įgaliotasis serveris (angl. *proxy server*). Įgaliotasis serveris yra tarpinis serveris, kurio tikslas atskirti galutinių vartotojų vykdymo klientus nuo jų naršomų vietų [Pet21]. Tokie įgaliotieji serveriai, kurie veikia kaip tarpininkai tarp interneto svetainės ir užklausos šaltinio yra vadinami persiuntimo įgaliotaisiais serveriais (angl. *forward proxy*). Šiame darbe bus akcentuojamas vienas iš porūšių: besisukantys įgaliotieji serveriai (angl. *rotating proxy*). Iš pavadinimo besisukantys įgaliotieji serveriai įvardina įgaliotojų serverių tinklą, kurio tikslas – sklaidyti HTTP užklausų vykdymą pastoviai keičiant IP adresą. Pastovus IP adresų keitimas riboja interneto svetainės serverio galimybes atskirti interneto robotų tinklus nuo įprastų vartotojų srauto. Taigi, duomenų ištraukimo procesas paskirstomas bei išvengiama IP adreso blokavimo problemos.

3.2.3. HTML analizatorius

Sėkmingai įvykdžius HTTP užklausą, gautas HTML dokumentas turi būti apdorojamas. Apdorojimui naudojamas HTML dokumento analizatorius (angl. *html parser*). Kiekvienas analizatorius turi būti pritaikomas specifinei interneto svetainei, iš kurios būtų ištraukiami duomenys. Pavyzdžiui, atvejais, kai žinomos prielaidos, kurios palengvintų duomenų ištraukimą, kaip žinios apie interneto svetainės turinio organizavimą. Bendru atveju HTML dokumento analizavimas apima tokenizavimą – HTML žymių pradžios ir pabaigos tokenų radimą, įskaitant atributų vardus bei vertes, ir medžio konstravimą [Moz21b]. Vadinasi, analizatoriaus tikslas yra sudaryti hierarchinį HTML žymių medį, pagal kurį būtų galima identifikuoti interesų sritį bei ištraukti nuorodas į kitus susijusius interneto svetainės tinklalapius. Toliau atliekamos papildomos HTML žymių medžio ar konkretaus poaibio transformacijos, remiantis žiniomis apie interneto svetainę. Įvardinamos kelios iš plačiai taikomų transformacijų: reguliariojo reiškinio teksto eilutės atitikmens ieškojimas (angl. *regex matching*), dokumento objektų modelio analizavimas (angl. *DOM parsing*) bei specifinių semantinių elementų išskyrimas.



5 Pav. HTML dokumento semantinių elementų hierarchijos pavyzdys

3.3. Prevencijos būdų palyginimo kriterijai

Šiame poskyryje apibrėžiami bendri kriterijai, pagal kuriuos bus palyginti skirtingi automatizuoto duomenų surinkimo prevencijos būdai. Kriterijai buvo pasirinkti remiantis nagrinėtomis problemomis (žr. 2.1, 2.2, 2.3 poskyrius). Dalis kriterijų buvo išvedami atsižvelgiant į analizuotus saityno duomenų ištraukėjo architektūros komponentus. Tikslas yra įvertinti analizuojamo prevencijos būdo taikymo sritį, mastą, kokiu etapu vykdoma prevencija ir efektyvumą priešpriešinei prevencijai (iš interneto roboto pusės).

2 lentelė. Automatizuoto duomenų surinkimo prevencijos būdų palyginimo kriterijai

Kriterijus	Paaiškinimas
Įgyvendinamumas	Kriterijus padengia šias sritis: ar reikalingos specifinės žinios, įranga, palaikymas naudojamam prevencijos būdui
Prevencijos mastas	Kokiai interneto svetainės daliai taikoma prevencija
Prevencijos etapas	Kada atliekama prevencija pradedant nuo pirmos užklausos vykdymo iki keliavimo per interneto svetainės tinklalapių nuorodas bei HTML dokumento analizavimo
Saugumas	Kriterijus įvardina ar egzistuoja apėjimo būdai taikomai prevencijai ir kaip lengva tai apeiti, išvengti
Įtaka tikriems vartotojams	Kaip taikomas prevencijos būdas paveikia tikrų vartotojų srautą interneto svetainėje
Taikymas rinkoje	Ar praktiška taikyti prevencijos būdą, paplitimas esamoje rinkoje

3.4. Prevencijos būdų analizė

Šiame poskyryje atliekama pasirinktų prevencijos būdų analizė. Atitinkamai aprašomi technikų, metodų, kombinuojamų žingsnių veikimo principai bei apibendrinama, kaip tikslingai individualūs prevencijos būdai tenkina apibrėžtus kriterijus. Būdai buvo pasirinkti iš šiame darbe nagrinėjamų šaltinių, kurie padengia skirtingus prevencijos etapus, tarp jų išskiriant tuos, kurie yra pritaikomi praktikoje.

3.4.1. Trijų sąrašų filtravimo technika

A. Haque ir S. Singh pateikia trijų sąrašų filtravimo techniką kaip vieną iš galimų sprendimų automatizuoto duomenų surinkimo prevencijai [HS15]. Atitinkamai technika priklauso nuo juodojo, pilkojo bei baltojo sąrašų sudarymo. Juodąjį sąrašą sudaro užblokuoti IP adresai, pilkąjį – IP adresai, kurie potencialiai priklauso duomenų ištraukėjui, t. y. pasižymėjo įtartina veikla, ir baltąjį sąrašą sudaro patikimi IP adresai [HS15]. Interneto svetainės lankytojas, nepriklausomai, ar tai tikras vartotojas, ar interneto robotas, patekęs į pilkąjį sąrašą turi spręsti pateikiamus tiuringo testus iki tol, kol bus įtrauktas į baltąjį sąrašą. Į juodąjį sąrašą įprastai iš karto įtraukiami identifikuoti interneto robotai, kurie ignoruoja „robots.txt“ failą, o perėjimas iš pilkojo sąrašo į bet kurį kitą sąrašą vyksta naudojant robotų atskyrimo metodus [HS15]. Siekiant tinkamai pasirinkti robotų atskyrimo metodą,

būtinas supratimas apie interneto svetainės auditoriją: ar orientuojamasi į sugrįžtančius vartotojus, ar naujus. Taip pat priklausomai nuo pasirinkto tiuringo testo, pateikiamos užduoties sudėtingumas tiesiogiai lemia prevencijos efektyvumą, identifikuojant interneto robotus. Pavyzdžiui, pilkasis sąrašas gali būti labai ilgas, dėl ko tampa itin svarbu neįtraukti tikrų vartotojų arba būtų neigiamai paveikiama vartotojo patirtis (angl. *UX*[6]) [Jaw17]. Vadinasi, trijų sąrašų filtravimo technika reikalauja specifinių žinių, supratimo, renkantis robotų atskyrimo metodus bei tiuringo testą.

Trijų sąrašų filtravimo technika taikoma visai arba pasirinktai interneto svetainės daliai. Prevencija pradedama taikyti nuo pirmos HTTP užklausos vykdymo, kai patikrinama priklausomybė vienam iš trijų sąrašų, ir tęsiama tol, kol vartotojas arba užblokuojamas, arba įtraukiamas į baltąjį sąrašą. Saugumo prasme technika nėra tinkama, kai duomenų ištraukimą vykdo paskirstytas interneto robotų tinklas dėl besisukančių įgaliotųjų serverių naudojimo. Vadinasi, labiau orientuojamasi į pavienius neetiškus interneto robotus. Pasirinktam robotų atskyrimo metodui reikalinga atskira posistemė, kuri galėtų kaupti reikiamus duomenis analizei. Posistemė negarantuoja, kad tikri vartotojai nebus klaidingai identifikuojami kaip robotai, todėl taikomas prevencijos būdas gali stipriai paveikti vartotojų srautą. IP adresų blokavimas yra plačiai naudojamas informacinių technologijų rinkoje, tačiau informacijos apie trijų sąrašų naudojimą nebuvo rasta.

3.4.2. „Honeypot“

„Honeypot“ (tiesioginis vertimas „medaus puodas“) įvardina techniką, kurios tikslas identifikuoti neteisėtus bandymus ištraukti interneto svetainės informaciją ar kitaip rinkti duomenis iš uždraustų interneto svetainės dalių. Bendru atveju išskiriamos dvi aptariamos technikos rūšys: aktyvūs medaus puodai, kurie bando „sugauti“ nepageidaujamą srautą, ir pasyvioji rūšis, kuri naudojama tik aptikimui [Mck16]. Pagrindinė technikos idėja į HTML dokumentą įtraukti įprastam vartotojui nematomas nuorodas. Nuoroda laikoma nematoma, jeigu HTML dokumento stiliaus maketas apibrėžia turinį už vizualiai matomos zonos vartotojui arba išjungiamas žymės vaizdavimas, t. y. dokumento kodas verčiamas (angl. *rendering*) į interaktyvų tinklalapį taip, tarsi žymės nebūtų. Pavyzdžiui, pakopinių stilių šablone pasirinktam elementui nurodant *display: none*, dėl ko elementas ir visi jo palikuonys nebebus paskelbti naudojant ekrano skaitymo technologiją [Moz21c]. Interneto robotai, kurie neatsižvelgia į nuorodos kontekstą, tokią nuorodą įtraukia į neaplankytų tinklalapių sąrašą. Kai ateina eilė vykdyti užklausą į nematomą nuorodą, interneto svetainės serveris gali

[6] UX – vartotojo patirtis (angl. *User Experience*)

identifikuoti, kad užklausą įvykdė interneto robotas. Aptikus interneto robotą, galima imtis papildomų priemonių, pavyzdžiui, įtraukimo į juodąjį sąrašą ar tyčinio suklastotų duomenų pateikimo [Jon20].

Palyginus su trijų sąrašų filtravimo technika, medaus puodo technika nereikalauja papildomų žinių, supratimo įgyvendinant sprendimą. Prevencija įprastai taikoma specifinei interneto svetainės daliai, kurioje gausu vertingo turinio, ir pradedama tik nuo HTTP užklausos vykdymo nematoma nuoroda, todėl iki to taško ištraukti duomenys nėra apsaugomi. Medaus puodo technikos gali išvengti didžioji dalis komercinių interneto robotų tinklų [URR+17]: atsiranda papildomas poreikis peržiūrėti HTML dokumentą kartu su pakopinių stilių šablonu. Pavyzdžiui, jeigu interneto robotas randa tinklalapio elementą su paslėpto lauko žyma, galima tokio stiliaus išteklių neįtraukti į neaplankytų tinklalapių sąrašą [Mck16]. Taip pat nėra daroma jokia įtaka tikriems vartotojams, nes nematomos nuorodos vartotojams nėra tiesiogiai pasiekiamos. Peržvelgus skirtingus prevencijos būdus pastebima, kad medaus puodo technika įprastai naudojama kartu su kitais prevencijos būdais kaip praplėtimas su tikslu padengti skirtingas interneto svetainės sritis.

3.4.3. Užklausos šaltinio indikatorių analizė

Užklausos šaltinio indikatorių analizė (angl. *fingerprinting*) yra technika, skirta išgauti duomenis iš kliento užklausų vykdymo konteksto, pavyzdžiui, interneto naršyklės su tikslu sukurti unikalų identifikatorių. Šiuo atveju duomenys nėra saugomi kliento pusėje (angl. *stateless*), pavyzdžiui, slapukuose, todėl suvaržymai susiję su BDAR[7] nėra taikomi. Įrenginio „piršto antspaudu“ laikomas sistemos atributų rinkinys, kurį sudaro retai arba nuspėjamai besikeičiantys indikatoriai kaip įrenginio ekrano matmenys, įdiegtos programinės įrangos versijos ar įdiegtų šriftų sąrašas [AJN+13]. Bendru atveju indikatoriai yra surenkami naudojantis aplikacijų programavimo sąsajomis. Viena iš plačiausiai naudojamų yra „JavaScript“ programavimo sąsaja. „JavaScript“ sąsaja galima pasiekti interneto naršyklės resursus kaip navigatoriaus ir ekrano objektus. Atitinkamai navigatorius yra objektas, kuriame saugoma informacija apie naršyklės kūrėjus, specifinę versiją bei palaikomus papildinius (angl. *plugins*), MIME[8] tipus, o ekrano objekte – informacija apie kliento ekrano matmenis, spalvų bei pikselių gylį [AJN+13]. Pabrėžiama, kad dėl „Adobe Flash“ palaikymo nutraukimo 2020 m. gruodžio 31 d. interneto naršyklės grąžinamos navigatoriaus savybės yra apribotos dėl vartotojų privatumo užtikrinimo. Vadinasi, atsiranda poreikis išskirti papildomus

[7] BDAR – bendrasis duomenų apsaugos reglamentas

[8] MIME – daugiafunkciniai interneto pašto plėtiniai (angl. *Multipurpose Internet Mail Extensions*)

požymius vietoj papildinių ar MIME tipų. Papildomi požymiai gali būti išskiriami remiantis interneto naršyklės plėtiniais (angl. *extensions*). Pagrindinis skirtumas tarp papildinio ir plėtinio: papildinys žino tik apie tinklalapį, kuriame yra, t. y. nėra žinoma apie naršyklę ar kas yra vykdoma kituose tinklalapiuose [Tax17]. Plėtiniai negali būti išgaunami naudojant „JavaScript“ sąsają, tačiau potencialiai gali būti identifikuojami pagal jų šalutinius poveikius [AJN+13]. Vienas iš pavyzdžių: ar naudojama reklamos blokavimo priemonė. Papildomai išskiriami pažangūs interneto naršyklės pirštų antspaudai kaip įrangos naudojamo procesoriaus loginių branduolių skaičius, grafinė korta, garso kontekstas (angl. *AudioContext*) bei „Canvas“ ir „WebGL“ sugeneruotos maišos (angl. *hash*). Atitinkamai „Canvas“ ir „WebGL“ maišos sugeneruojamos pagal duomenis kaip kiekvienas pikselis atvaizduojamas, įskaitant ir įdiegtus šriftus, o garso kontekstas sudaromas sugeneruojant trumpą garso įrašą (angl. *sample*), remiantis „Web Audio“ sąsaja su tikslu unikaliai identifikuoti pagal naudojamą garso plokštę [EFF20]. Taigi, remiantis įvardintais indikatoriais galima sudaryti unikalų kliento identifikatorių, kuris nėra susietas su IP adresu, todėl IP adresui pakitus įmanoma identifikuoti pokytį bei imtis atitinkamų priemonių.

Nors užklausos šaltinio indikatorių analizė reikalauja papildomų žinių, supratimo įgyvendinant sprendimą, prevencija gali būti efektyviai taikoma visai interneto svetainei. Prevencija pradedama taikyti po pirmos HTTP užklausos. Jeigu indikatoriai rodo, kad yra neatitikimas tarp pateikiamos interneto naršyklės informacijos arba naujo IP adreso identifikatorius sutampa su jau žinomu, galima užblokuoti prieigą potencialiam interneto robotui arba pateikti tiuringo testą. Taigi, saugumo prasme, remiantis vykdomų užklausų indikatorių analize, galima suvaldyti paskirstyto interneto robotų tinklo problemą dėl besisukančių įgaliotųjų serverių naudojimo. Priklausomai nuo naudojamų indikatorių aptariama technika gali praktiškai nepaveikti įprasto tikrų vartotojų srauto interneto svetainėje. Remiantis 2021 IEEE saugumo ir privatumo simpoziumo metu pateikta statistika, užklausų šaltinio indikatorių analizė naudojama daugiau nei tarp 10% populiariausių 100 000 interneto svetainių bei ketvirtadalyje 10 000 populiariausių svetainių [IES21].

3.4.4. Atsitiktinis tinklalapių turinio atkūrimas

Didžioji dalis pateiktų prevencijos būdų orientuojasi į analizes, sprendimus, kurių tikslas identifikuoti interneto robotą bei vienaip ar kitaip sustabdyti duomenų ištraukimą užklausos vykdymo lygyje. Siekiant padengti visus duomenų ištraukėjo komponentus, analizuojama atsitiktinio tinklalapių turinio atkūrimo technika, kuri sutelkia dėmesį į duomenų ištraukėjo analizatorių. W. Wang, Y. Zheng ir kiti bendraautoriai pateikia „WebRanz“ techniką, kuri atsitiktinai suskirsto URL

adresus ir turinį tinklalapiuose, kad turinio pagrindu veikiantys interneto robotai nebegalėtų naudoti požymių atitikties (angl. *pattern matching*) su tikslu identifikuoti bei manipuliuoti DOM[9] elementus [WZX+16]. Pabrėžiama, kad elementai nėra keičiami vieni kitais, dėl ko galėtų išsiskirti stebima tinklalapio interpretacija ar funkcionalumas. Technika įgyvendinama dviem etapais serverio ir kliento pusėje siekiant padengti tiek statinę HTML dokumento dalį, tiek dinamiškai sugeneruojamą turinį vykdymo metu. Serverio pusėje apdorojamas statinis HTML dokumento turinys kartu su susijusiu pakopinių stilių šablonu: ID bei klasės atributams atsitiktinai priskiriamos sugeneruotos reikšmės, o URL adresai užšifruojami remiantis viešojo rakto kriptografija. Tais atvejais, kai HTML elementai turi tuos pačius ID atributus, priskiriama ta pati atsitiktinai sugeneruota reikšmė [WZX+16]. Kadangi URL adresai užšifruojami serverio pusėje, atsiranda poreikis turėti skaidrų tarpinį serverį (angl. *transparent proxy*). Skaidraus tarpinio serverio tikslas – atkoduoti ir peradresuoti kliento užklausą kartu su slapukais bei grąžinti tikro šaltinio užklausos atsakymą [WZX+16]. Papildomai, siūloma skaidriame tarpiniame serveryje kartu atlikti ir saityno duomenų šaltinių podėliavimą (angl. *caching*). Toliau kliento pusėje perrašomi ID bei klasių atributų selektoriai, kaip ir reikšmių gavimo, priskyrimo ir DOM elementų kūrimo funkcijos, su tikslu įtraukti vienas prie vieno žemėlapį, sudarytą serverio pusėje, generuojant atsitiktines reikšmes iš tikrų. Taigi, automatizuotas duomenų ištraukėjas, kuris dominantį turinį identifikuoja remiantis HTML žymių atributais, neturės atskaitos taško kaip tik HTML žymių hierarchiją.

```
1.<style type="text/css">
2.   .office-sessions-widget .video-item {...}
3.</style>
4.<div class="office-sessions-widget">
5.  <div class="video-item">...</div>
6.</div>
```

```
11.<style type="text/css">
12.  .U7n231k .hcd1nc {...}
13.</style>
14.<div class="U7n231k">
15.  <div class="hcd1nc">...</div>
16.</div>
```

6 Pav. Klasės atributams atsitiktinai sugeneruotų reikšmių pavyzdys [WZX+16]

Įgyvendinant atsitiktinio tinklalapių turinio atkūrimo techniką, reikalingos specifinės žinios, supratimas renkantis atsitiktinių reikšmių generatorių bei kaip yra perrašomos „JavaScript“ funkcijos. Technika taikoma visoms interneto svetainės dalims. Prevencija prasideda po pirmos HTTP užklausos įvykdymo, kai duomenų ištraukėjo analizatorius bando ištraukti dominantį turinį. Saugumo prasme, kiekvienos užklausos į tą patį tinklalapį metu, sugeneruojamos naujos atributų bei URL adresų reikšmės, todėl bandyti atkurti reikšmių žemėlapį nėra praktiška [WZX+16]. Pastebima, kad taikomos technikos posistemė gali būti taikoma bet kokiai interneto svetainei, t. y. laisvai jungiama su egzistuojančiomis sistemomis (angl. *loose coupling*). Tikriems vartotojams nėra daroma įtaka,

[9] DOM – dokumento objektų modelis (angl. *Document Object Model*)

kadangi interneto svetainės stebima tinklalapio interpretacija ar funkcionalumas nesikeičia. Peržvelgus didžiųjų kompanijų kaip „Google“, „Facebook“ šaltinių kodą (angl. *source code*) pastebima, kad naudojamos CSS atributų selektorių sumažinimo technikos (angl. *minification*) su tikslu sumažinti paketo dydžius, kas yra atitikmuo aptariamos technikos tikslui – sugluminti (angl. *obfuscation*) interneto robotą atsitiktinai atkuriant tinklalapių turinį – skirtumas tik ketinimuose.

3.5. Prevencijos būdų palyginimas

Šiame poskyryje pateikiamas nagrinėtų automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdų palyginimas pagal 3.3 poskyryje apibrėžtus kriterijus.

3 lentelė. Nagrinėtų automatizuoto duomenų surinkimo prevencijos būdų palyginimas

	Trijų sąrašų filtravimo technika	**„Honeypot“ (medaus puodo technika)**	**Užklausos šaltinio indikatorių analizė**	**Atsitiktinis tinklalapių turinio atkūrimas**
Įgyvendina-mumas	Reikalingos specifinės žinios, atskira posistemė	Sprendimas nereikalauja papildomų žinių, supratimo	Reikalingos specifinės žinios renkantis indikatorius	Reikalingos žinios renkantis atsitiktinių reikšmių generatorių
Prevencijos mastas	Visai arba daliai interneto svetainės	Specifinei interneto svetainės daliai	Visai interneto svetainei	Visai interneto svetainei
Prevencijos etapas	Nuo pirmos HTTP užklausos	Nuo HTTP užklausos į nematomą nuorodą	Po pirmos HTTP užklausos	Bandant ištraukti duomenis iš HTML dokumento
Saugumas	Priklauso nuo pasirinktos posistemės	Komerciniai interneto robotų tinklai gali išvengti	Prevencija efektyvi prieš paskirstytų interneto robotų tinklus	Išvengiama turinį ištraukiant pagal poziciją HTML dokumento hierarchijoje
Įtaka tikriems vartotojams	Gali stipriai paveikti vartotojų srautą	Nedaro įtakos	Priklauso nuo pasirinktų indikatorių	Nedaro įtakos

Taikymas rinkoje	Duomenų nerasta	Kombinuoja kartu su kitais prevencijos būdais	Plačiai naudojama rinkoje	Panaši technika yra naudojama, bet nenustatyta, kad būtų paplitusi

Pagal 3 lentelę matoma, kad lengviausiai įgyvendinama prevencijos technika yra „honeypot“, tačiau atitinkamai nuo to skiriasi ir prevencijos mastas. Kaip aptarta 3.4.2 punkte, medaus puodo technika taikoma ten, kur gausu vertingo turinio. Priešingu atveju pateikiama daugiau požymių, pagal kuriuos būtų galima identifikuoti, kurių HTML žymių reiktų vengti. Kiekviena iš aptartų technikų prevenciją atlieka skirtinguose duomenų ištraukimo etapuose. Trijų sąrašų filtravimo technika – prieš pasiekiant interneto svetainės turinį, analizuojant lankytojų elgesį. Medaus puodo technika – kai aplankoma nematoma nuoroda HTML dokumente. Užklausos šaltinio indikatorių analizė – kai nuolatinių lankytojų indikatoriai išsiskiria nuo, pavyzdžiui, IP adreso. Atsitiktinis tinklalapių turinio atkūrimas – turinio išgavimo metu. Pastebima, kad kiekviena iš aptartų technikų potencialiai viena kitą papildo padengiant galimus prevencijos etapus. Prasčiausiai saugumo prasme išsiskyrė medaus puodo technika, kadangi dažnai į ją atsižvelgiama įgyvendinant priešpriešinę prevenciją, o iš tikro vartotojo perspektyvos – trijų sąrašų filtravimo technika; visi lankytojai privalo spręsti tiuringo testus iki kol bus įtraukti į baltąjį sąrašą, t. y. tikėtina, kad tikri vartotojai galimai paliks interneto svetainę. Užklausos šaltinio indikatorių analizė yra plačiausiai praktikoje naudojama technika kaip prevencijos būdas automatizuotam duomenų surinkimui iš tinklalapių. Pabrėžiama, kad atsitiktinio tinklalapių turinio atkūrimo technika naudojama tik dalinai, kadangi sugeneruotos reikšmės nėra atnaujinamos, keičiamos. Taigi, nors kiekviena technika turi savo silpnybių bei stiprybių, galima teigti, kad kombinuojant prevencijos būdus išskirtos silpnybės gali būti minimizuojamos kitų prevencijos būdų su tikslu padengti kuo daugiau skirtingų interneto svetainės dalių.

3.6. Apibendrinimas

Išnagrinėjus automatizuoto duomenų surinkimo prevencijos sampratą bei saityno duomenų ištraukėjo komponentus buvo išskirti prevencijos būdų palyginimo kriterijai. Kriterijai buvo naudojami palyginti pasirinktus prevencijos būdus. Pagal gautus palyginimo rezultatus galima spręsti, kad siekiant padengti kuo daugiau interneto svetainės sričių, prevencijos būdai turi būti tarpusavyje kombinuojami. Taigi, siekiant apsisaugoti nuo automatizuoto duomenų surinkimo iš tinklalapių, neužtenka įgyvendinti individualių prevencijos būdų.

4. Komercinės automatizuoto duomenų surinkimo prevencijos priemonės

Šiame skyriuje bus nagrinėjamos pasirinktos komercinės duomenų surinkimo prevencijos priemonės. Pasirinkimo kriterijai apima sąlygas kaip viešai prieinama informacija apie įgyvendinimo detales bei variacija tarp taikomų prevencijos būdų. Pagal kriterijus pasirinktos trys priemonės, o kitos nenagrinėjamos priemonės yra laikomos kaip alternatyvos. Skyriaus tikslas yra įvertinti 3 skyriuje apibrėžtų prevencijos būdų praktišką pritaikymą palyginant pasirinktas prevencijos priemones.

4.1. Pasirinkimo kriterijai

Šiame poskyryje apibrėžiami bendri kriterijai, pagal kuriuos bus palyginamos komercinės automatizuoto duomenų surinkimo prevencijos priemonės. Šiuo atveju prevencijos būdai yra tiesiogiai siejami su 3.3 poskyryje apibrėžtais prevencijos būdų palyginimo kriterijais.

4 lentelė. Automatizuoto duomenų surinkimo prevencijos priemonių palyginimo kriterijai

Kriterijus	Paaiškinimas
Prevencijos būdai	Kokiais būdais yra užtikrinama interneto robotų prevencija
Prevencijos taikymas	Kaip įgyvendinama prevencija identifikavus neetišką interneto robotą
Prevencijos pagrindas	Kriterijus įvardina analizuojamos priemonės procesus, kurie leidžia nuspręsti, kada reikalingas prevencijos taikymas
Prevencijos dimensija	Integravimosi taškas apsaugomai interneto svetainei

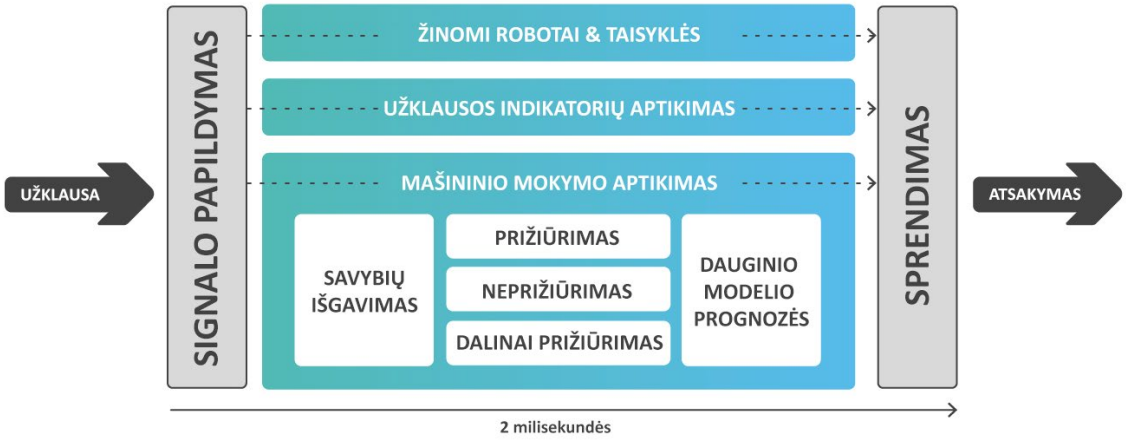
4.2. „DataDome“

„DataDome“ yra paslauginė programinė įranga (angl. *software as a service*) kaip robotų apsaugos sprendimas, skirtas elektroninės komercijos, mobiliųjų aplikacijų ir API sąsajų apsaugojimui. Programinė įranga pateikiama kaip integruojamas modulis ir palaikoma trylikoje skirtingų serverio pusės technologijų (pavyzdžiui, nginx, microsoft IIS, node.js, java ir kitos). Taip pat integracija atliekama ir kliento pusėje, įterpiant „JavaScript“ žymę, atsakingą už naujų HTML elementų sukūrimą ir globalių objektų konfigūravimą. Mobiliosioms aplikacijoms yra skirtas programavimo komplektas (angl. *SDK*). Autoriai, interneto robotų ir tikrų vartotojų atskyrimo problemą apibrėžia pateikdami pažangiausias priešpriešinės prevencijos technikas. Nuoseklių HTTP, TCP[10] ir TLS[11] užklausų indikatorių klastojimas, paskirstytų interneto robotų tinklų, naudojančių

[10] TCP – persiuntimo valdymo protokolas (angl. *Transmission Control Protocol*)

[11] TLS – transporto lygmens protokolas, skirtas saugumui užtikrinti TCP/IP tinkluose (angl. *Transport Layer Security*)

besisukančius įgaliotuosius serverius su patikimais gyvenamosios vietovės (angl. *residential*) IP adresais, taikymas, duomenų ištraukimo vykdymas lėtai ir paskirstytai, tikroviškų pelės judesių ir klaviatūros paspaudimų imitavimas, naudojant generatyvinius konkurencinius tinklus (angl. *GAN*), ir duomenų ištraukimo fermų naudojimas, sudarytų iš realių įrenginių [Dat21]. Vadinasi, taikoma prevencija turi atsižvelgti į įvardintus aspektus arba prevencijos taikymas tampa neveiksmingu. „DataDome“ pateikia savo modelį kaip sprendimą įvardintoms problemoms. Šiuo atveju „DataDome“ prevencijos taikymo veikimą sudaro penki aspektai. Pirmiausiai, surenkami užklausos indikatoriai bei kiti signalai. Toliau iš anksto atpažintiems svetainės lankytojams suteikiama prieiga prie serverio. Priešingu atveju atliekama interneto robotų užklausos indikatorių ir mašininio mokymo atpažinimų procesai. Galiausiai, arba suteikiama prieiga, arba pateikiamas tiuringo testas, užblokuojama. Signalai apie vartotojus sudaromi tiek serverio, tiek kliento pusėje. Pavyzdžiui, prie IP adreso pridedama papildoma trečiųjų šalių informacija kaip autonominės sistemos pavadinimas, IP adreso šalis, įskaitant atvirkštinio DNS[12], susieto su IP adresu, testavimą, siekiant patikrinti etiškus robotus, tokius kaip „Googlebot“ [Dat21]. Surinkti užklausos indikatoriai yra siunčiami į serverį ir apdorojami. Užklausos indikatorių atpažinimo metu tikrinami neatitikimai tarp tikėtinų rezultatų ir esamų, lyginant su išanalizuotų robotų elgesio duomenų baze. Pavyzdžiui, lankytojas apsimeta, kad naudoja „Chrome“, bet nesiunčia antraščių, kurios turėtų būti „Chrome“ [Dat21]. Mašininis mokymasis skirstomas į tris kategorijas. Prižiūrimas mokymasis įvertina įvestį pagal turimus praeities duomenis, neprižiūrimas mokymasis – identifikuoja išskirtis (angl. *outliers*), t. y. žymų nuokrypį nuo tikėtinų duomenų, ir dalinai prižiūrimas mokymasis naudojamas, kai duomenų ištraukimo procesas dalinai vykdomas su žmonių pagalba, pavyzdžiui, su tiuringo testų sprendimo fermų pagalba.



7 Pav. „DataDome“ prevencijos taikymo veikimo aspektai [Dat21]

[12] DNS – srities vardų struktūra (angl. *domain name system*)

4.3. „PerimeterX“

„PerimeterX“, kaip kibernetinio saugumo įrangos tiekėjas, nuo kitų prevencijos priemonių išsiskiria viešu faktu, kad naudoja medaus puodo techniką atskirti interneto robotų tinklus. Šiuo atveju naudojama pasyvaus tipo medaus puodo technika kaip papildomas požymis priimant sprendimus. „PerimeterX“ platformos architektūra yra suskaidyta į tris veikimo sritis. Atitinkamai, į kliento pusės užklausos indikatorių surinkimą ir agregavimą, rizikos balo sudarymą, remiantis mašininiu mokymusi ir elgesio analize, bei reagavimo į grėsmes politikos vartų įgyvendinimą, remiantis antrame žingsnyje gautais rezultatais [BS21]. Pabrėžiama, kad, analizuojant agreguotus duomenis, vykdomos ekspromtinės (lot. *Ad hoc*) užklausos. Ekspromtinės užklausos sukuriamos, kai kyla klausimų, kurių neįmanoma išspręsti naudojant iš anksto nustatytus duomenų rinkinius [Sno21]. Tokiu būdu siekiama efektyviai palyginti užklausos indikatorius su saugomais duomenimis apie klientą. Taip pat „PerimeterX“ sukūrė savo tiuringo testą, pavadinimu „žmogaus iššūkis“. Poreikis tokiam įrankiui apibrėžiamas kaip didėjanti atskirtis tarp mašininio mokymosi ir kompiuterinio regėjimo sričių pažangos ir progresyviai sunkėjančių tiuringo testų, pateikiamų tikriems vartotojams. Šiuolaikinėmis technologijomis automatizuotas tiuringo testų sprendimas pranoksta tikrus žmones, kai kuriose srityse, ir robotų tinklai neretai turi tiuringo testų sprendimo fermas, operuojamas tikrų žmonių [Ami21]. Taigi, tiuringo testams vis sunkėjant, neigiamai paveikiama vartotojo patirtis, o interneto robotai įgyvendina vis tobulėjančius problemų sprendimus. „PerimeterX“ tiuringo testas reikalauja vartotoją paspausti mygtuką ir laikyti. Autorių teigimu, vieno paspaudimo užtenka surinkti visus reikiamus duomenis nustatyti ar testą išsprendė žmogus, ar robotas su aukštu pasitikėjimo laipsniu [Ami21]. Be to, testas leidžia identifikuoti ar buvo naudojamos sprendimo fermos, matuojant išsprendimo laiką bei kitokias vartotojo sąveikas. Nors neįvardinta kaip, tačiau medaus puodo technika yra kartu integruojama į „žmogaus iššūkio“ tiuringo testą. Autoriai įvardina, kad medaus puodo technika, apjungta kartu su kriptografiniais sprendimais, leidžia apkrauti interneto robotų išteklius, apsunkinant tinklalapio atvaizdavimo etapą [BS21]. Pavyzdžiui, naudojamas kriptografinis darbo įrodymas (angl. *proof of work*), kuris reikalauja išeikvoti apibrėžtą skaičiavimo resursų skaičių prieš leidžiant užbaigti kitus veiksmus, kaip paspaudimą. Tokiu atveju dideliu mastu atliekamos duomenų ištraukimo atakos proporcingai didina kaštus duomenų gavybos procesui. Pridedama, kad be medaus puodo technikos, papildomai įgyvendinama ir peradresavimo, apgaulingo turinio pateikimo technikos.

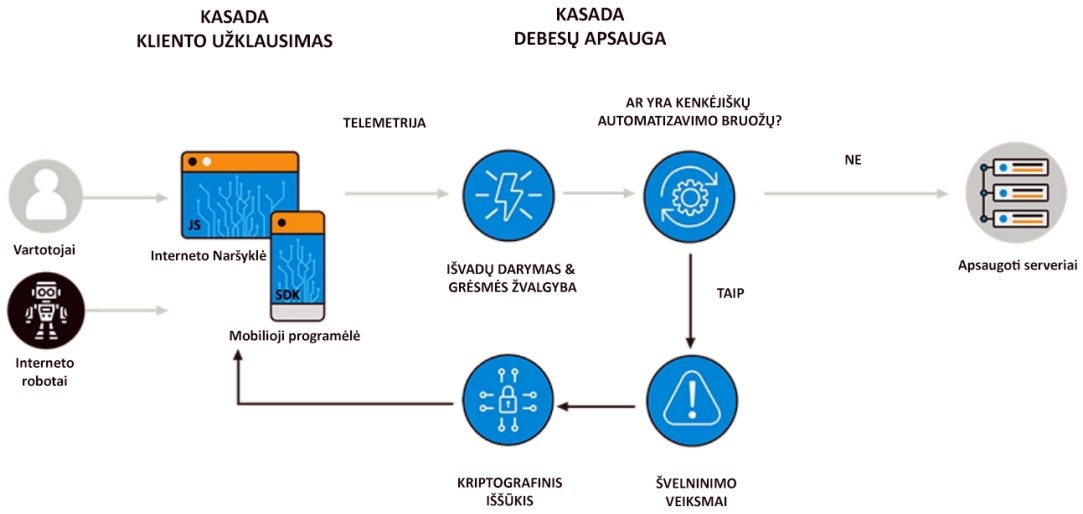
8 Pav. „Žmogaus iššūkio“ tiuringo testo vizuali reprezentacija [Ami21]

4.4. „Kasada“

„Kasada“ yra skaitmeninio srauto vientisumo sprendimas, skirtas interneto svetainėms, mobiliesiems įrenginiams ir API sąsajoms. Sprendimo autoriai „Kasada“ įvardina kaip alternatyvą įprastai taikomai prevencijai duomenų gavybos proceso metu. Šiuo atveju teigiama, kad ugniasienių, rizikos balų sudarymas bei elgesio analizė netenkina bent vienos iš įvardinamų sąlygų: prevencijos efektyvumo arba duomenų apdorojimo našumo. Siekiant išspręsti šias problemas, „Kasada“ vadovaujasi nulinio pasitikėjimo filosofija (angl. *zero trust philosophy*). Nulinio pasitikėjimo filosofija yra saugumo modelis, grįstas griežtos prieigos kontrolės ir nepasitikėjimo niekuo principu kiekvienos užklausos metu, net ir atvejais, kai klientas jau įrodė, kad nėra robotas [Clo21]. Prevencija yra grįsta kriptografinių iššūkių pateikimu naudojant skaidrų tarpinį serverį, t. y. taikoma bandant įvykdyti užklausą, naudojant HTML formas, AJAX[13] ir API sąsajas. „Kasada“ sprendimas siekia interneto robotus įtikinti, kad duomenų gavybos procesas nėra užbaigtas, pateikiant nežymų, bet sudėtingą matematinį iššūkį dar prieš užkraunant interneto svetainę, taip didinant reikiamus kaštus išspręsti problemai [Whi19]. Pabrėžiama, kad jokie interneto svetainės klientai nėra blokuojami. Potencialiai identifikavus interneto robotą, kriptografinių iššūkių sudėtingumas atitinkamai yra didinamas. Taigi, interneto robotas nežino, kad išnaudojami jo skaičiavimo resursai, galimai net ir pažeidžiant (angl. *bricking*) duomenų gavybos įrangą. Nustatyti ar klientas yra robotas, surenkami pažangūs užklausos šaltinio indikatoriai. Indikatoriai leidžia su kiekviena užklausa iš karto priimti sprendimą, kokių švelninančių veiksmų (angl. *mitigative actions*) reikia imtis. Bendru atveju „Kasada“ sprendimas yra orientuotas į duomenų ištraukimo proceso apsunkinimą taip, kad proporcingai ir išaugtų reikiami skaičiavimo kaštai. Pridedama, kad be kriptografinių iššūkių, sprendimas įgyvendina netikrų užklausų pateikimą, tyčinį užklausos šaltinio peradresavimą ir

[13] AJAX – asinchroninio JavaScript ir XML technologija (angl. *asynchronous JavaScript and XML*)

„Kasada“ patentuotą kodo sugluminimo (angl. *obfuscation*) įrankį su savo nuosavu interpretatoriumi (angl. *interpreter*).



9 Pav. „Kasada“ prevencijos taikymo veikimo aspektai [Kas22]

4.5. Prevencijos priemonių palyginimas

Šiame poskyryje pateikiamas nagrinėtų komercinių automatizuoto duomenų surinkimo prevencijos priemonių palyginimas pagal 4.1 poskyryje apibrėžtus kriterijus.

5 Lentelė. Nagrinėtų komercinių prevencijos priemonių palyginimas

	„DataDome“	**„PerimeterX“**	**„Kasada“**
Prevencijos būdai	Užklausos šaltinio indikatorių analizė	Medaus puodo technika, užklausos šaltinio indikatorių analizė	Užklausos šaltinio indikatorių analizė
Prevencijos taikymas	Prieigos blokavimas, tiuringo testai	Prieigos blokavimas, patentuotas tiuringo testas, kriptografinis iššūkis	Prieiga nėra blokuojama, patentuotas kriptografinis iššūkis
Prevencijos pagrindas	Elgesio analizė, remiantis mašininiu mokymusi	Elgesio analizė, remiantis mašininiu mokymusi	Nulinio pasitikėjimo filosofija
Prevencijos dimensija	Skaidrus tarpinis serveris	Skaidrus tarpinis serveris	Skaidrus tarpinis serveris

Pagal 5 lentelę matoma, kad skaidraus tarpinio serverio naudojimas ir užklausos šaltinio indikatorių analizė dominuoja kiekvienoje iš nagrinėtų komercinių automatizuoto duomenų surinkimo

prevencijos priemonių. Prevencijos taikymas, naudojant tarpinį serverį, leidžia priemones laisvai jungti su egzistuojančiomis kliento sistemomis (angl. *loose coupling*). Taip pat dislokuoti papildomus skaidrius tarpinius serverius pagal individualius poreikius. Didžiąją dalimi prevencija yra taikoma pateikiant tiuringo testą arba blokuojant prieigą. Tokiais atvejais priemonės turi numatyti kaip bus elgiamasi klaidingai teigiamai (angl. *false positive*) įvertinus tikrą interneto svetainės vartotoją kaip interneto robotą. Pavyzdžiui, atvejais, kai klientas naudoja seną interneto naršyklės versiją arba blokuoja pilnai ar dalinai „JavaScript“, „WebGL“ vykdymą. Pastebima, kad tiuringo testai praranda savo svarbą, atsižvelgus į pažangą mašininio mokymosi ir kompiuterinio regėjimo srityse. Tam nepadeda faktas, kad naudojamos tiuringo testų fermos, operuojamos tikrų žmonių. Šias problemas „PerimeterX“ bando spręsti su savo patentuotu „žmogaus iššūkio“ tiuringo testu. Bendru atveju tiuringo testai yra arba taikomi ribotinai kraštutiniais atvejais, arba keičiami kriptografiniais iššūkiais. Kriptografinis iššūkis, priešingai nei tiuringo testas, nėra naudojamas identifikacijai, bet progresyviai apsunkinti duomenų gavybos procesą. Taigi, tenkinamas 2.2 poskyryje apibrėžtas poreikis gebėti atskirti interneto robotų tinklus, nepaveikiant įprastų vartotojų srauto interneto svetainėje. Prevencijos pagrindas didžiąją dalimi yra orientuotas į interneto svetainės lankytojų elgesio analizę. Mašininis mokymas, šiuo atveju naudojamas analizuoti didelius kiekius informacijos, susijusios su klientais. Alternatyvūs sprendimai, kaip „Kasada“, išsiskiria nuo kitų priemonių bendru požiūriu į prevenciją, kad interneto robotai neturi būti blokuojami, bet įtikinami išnaudoti savo skaičiavimo resursus. Pagrindinis iššūkis, taikant nulinio pasitikėjimo filosofiją, apibrėžiamas kaip reikalavimas korektiškai identifikuoti interneto robotus. Pabrėžiama, kad, šiuo metu, analizuojant komercines prevencijos priemones nebuvo rasta atvejų, kur būtų įgyvendinama atsitiktinio tinklalapio atkūrimo technika. Taigi, siekiant praktiškai taikyti prevencijos būdus, prevencijos sprendimai turi gebėti adaptuotis tiek prie esamų, tiek prie naujų duomenų surinkimo ir priešpriešinės prevencijos technikų kiekviename prevencijos etape.

4.6. Apibendrinimas

Apžvelgus pasirinktas komercines automatizuoto duomenų surinkimo prevencijos priemones, pastebima, kad praktikoje nėra taikoma prevencija HTML dokumento analizavimo metu. Atvejais, kai pažangūs neetiški interneto robotai nėra sustabdomi iki duomenų gavybos pabaigos, tolimesnis duomenų ištraukimas nereikalauja papildomų pastangų. Šiuo atveju galima sėkmingai analizuoti interneto svetainės turinio organizavimą su tikslu patobulinti duomenų ištraukimo procesą ir pasiruošti kitai atakai. Vadinasi, atsiranda poreikis turėti galimybę atsitiktinai atkurti tinklalapio turinį.

5. Automatizuoto duomenų surinkimo prevencijos prototipas

Šiame skyriuje siekiama įvertinti komercinių prevencijos priemonių plečiamumo galimybes, pasirenkant vieną iš analizuotų įrankių ir integruojant į skaidrų tarpinį serverį. Poreikis prototipui apibrėžiamas 4.6 poskyryje. Norint pabrėžti plečiamumo galimybes, pasirinkta įgyvendinti ir pagal poreikį pritaikyti atsitiktinio tinklalapių turinio atkūrimo techniką. Technika pasirinkta dėl dviejų priežasčių. Apžvelgus komercines prevencijos priemones nustatyta, kad atsitiktinis tinklalapių turinio atkūrimas nėra taikomas praktikoje ir antrą – nėra padengiami visi nagrinėti prevencijos etapai. Taigi, skyriaus tikslas parodyti, kad prevenciją įmanoma taikyti kiekviename duomenų ištraukimo etape praktiškai ir kartu praplėsti nagrinėtų įrankių galimybes pasirinktu prevencijos būdu.

5.1. Prototipo veikimo principas

Sukurtas prototipas turi tenkinti dvi sąlygas: turi būti lengvai jungiamas su egzistuojančiomis kliento sistemomis (angl. *loose coupling*) (plačiau žr. 4.5) ir vykdyti prevenciją, nepaveikiant įprastų vartotojų srauto interneto svetainėje (plačiau žr. 2.2). Vadinasi, įgyvendinamas atsitiktinis tinklalapių turinio atkūrimas turi užtikrinti, kad nesikeistų tiek vizuali HTML dokumento reprezentacija, tiek teikiamas funkcionalumas. Šiuo atveju prevencija buvo taikoma apsunkinant turinio ištraukimo procesą dinamiškai keičiant ID, klasių atributų selektorių reikšmes bei URL adresus į atsitiktines reikšmes. Klientų užklausų apdorojimui ir sugeneruotų interneto svetainės tinklalapių podėliavimui (angl. *caching*) buvo naudojamas skaidrus tarpinis serveris. Taigi, skaidrų tarpinį serverį galima išskirti kaip izoliuotą prevencijos tašką tarp interneto svetainės vartotojo ir tikro teikiamos paslaugos serverio. Prevencijos taikymas skirstomas į du etapus. Pirmą etapą – HTML dokumento bei susijusio pakopinių stilių šablono apdorojimą, ir antrą – interneto svetainės srauto stebėjimą pagal apibrėžtus požymius. Antras etapas buvo įgyvendinamas integruojant prototipą kartu su „PerimeterX“ komercine prevencijos priemone. Šiam etapui įgyvendinti, iš nagrinėtų prevencijos priemonių, buvo pasirinktas „PerimeterX“ sprendimas dėl teorinės galimybės padengti kiekvieną nagrinėtą prevencijos etapą, tačiau siūlomas sprendimas nepriklauso nuo pasirinkto įrankio. Pabrėžiama, kad šiuo metu nėra alternatyvių atvirojo kodo sprendimų komercinėms prevencijos priemonėms, kurių šaltinio kodas būtų viešai prieinamas.

Prototipą sudaro du serveriai – abu įgyvendinti, naudojant „Express.js“ interneto programų tiekimo karkasą. HTTP užklausų turinio, antraščių ir slapukų apdorojimas yra realizuojamas, naudojant tam skirtas tarpines programines įrangas (angl. *middleware*). Pirmas serveris yra interneto

svetainės leidėjas (angl. *publisher*) ir antras – skaidrus tarpinis serveris. Interneto svetainės leidėjas atsakingas už duomenų nuskaitymą, podėliavimą bei atsitiktinį tinklalapių turinio atkūrimą, o skaidrus tarpinis serveris už užklausų dekodavimą, peradresavimą bei komercinės prevencijos priemonės integravimą su tikslu analizuoti HTTP užklausų požymius. Pabrėžiama, kad vienintelė privaloma leidėjo serverio atsakomybė yra duomenų nuskaitymas, o visa kita gali būti atliekama skaidriame tarpiniame serveryje (žr. 5.3 poskyrį). Atsitiktiniam tinklalapių turinio atkūrimui naudojama pavyzdinė, duomenų ištraukimui skirta, interneto svetainė (`https://books.toscrape.com/`). Toliau aprašomi įgyvendinti prototipo veikimo žingsniai, su kokiomis problemomis susidurta ir kaip tai buvo sprendžiama.

5.2. HTML turinio ir pakopinių stilių šablono apdorojimas

Šiame poskyryje bus aprašomos W. Wang, Y. Zheng ir kitų bendraautorių pateiktos „WebRanz“ [WZX+16] technikos įgyvendinimo detalės. Pabrėžiama, kad siekiama išsiplėsti nuo teorinio „WebRanz“ technikos pagrindo ir pasiūlyti praktišką sprendimą, taikant techniką. Antrinis tikslas yra išanalizuoti, su kokiomis problemomis yra susiduriama, t. y. įvertinti pritaikymo ribas.

5.2.1. HTTP užklausų vykdymo pagrindas

Įvykdžius HTTP užklausą į interneto svetainės serverį, reikalingi saityno duomenų šaltiniai bei HTML dokumentas yra pateikiami klientui tiesiogiai į interneto naršyklę. Šiuo atveju kiekvienam šaltiniui yra siunčiama atskira užklausa, t. y. viena užklausa neįmanoma grąžinti daugiau nei vieno resurso. Tačiau remiantis HTTP/1.1 standartu [IETF14], klientas turi galimybę palaikyti pastovų ryšį su serveriu su tikslu vienu metu išsiųsti kelias užklausas. Serveris gali duomenis apdoroti paraleliai, tačiau atsakymai turi būti grąžinami ta pačia tvarka kaip ir buvo pateiktos užklausos. Taigi, pirmiausiai interneto naršyklė gauna HTML dokumentą ir tada pradeda atvaizdavimo procesą į interaktyvų puslapį. Atvaizdavimo metu, išsiunčiamos papildomos užklausos kiekvienam įterptam resursui, pradedant nuo metaduomenų, saugomų `<head>` elemente, ir baigiant paskutiniu įterptu resursu. Pasinaudojant šiuo veikimo principu, įgyvendinamas atsitiktinis tinklalapių turinio atkūrimas. Toliau aprašoma specifiškai prototipo tinklalapių turinio atkūrimo dalies įgyvendinta koncepcija.

5.2.2. Abstrakčios sintaksės medžio sudarymas

Gavus užklausą į atitinkamą HTML dokumentą, pradedamas atsitiktinis tinklalapio turinio atkūrimas. Pagal pateiktą adresą nuskaitomas HTML dokumento failas kaip simbolių eilutė. HTML

dokumentas taip pat gali būti saugomas operatyvioje atmintyje, vietoj skaitymo iš diskinio kaupiklio (angl. *disk storage*). Nuskaityta simbolių eilutė yra toliau verčiama (angl. *parsed*) į dokumento objektų modelį. Vertimo procesas apima leksinę analizę, tokenizavimą ir medžio konstravimą. Dažnai leksinė analizė – tokenų leksemų apibrėžimas, ir tokenizavimas – tokenų išskyrimas, yra laikomi kaip vienas žingsnis. Sukonstruotas modelis reprezentuoja abstrakčios sintaksės medį (angl. *abstract syntax tree;* toliau – AST). Pavyzdinė medžio viršūnių (mazgų) struktūra yra pateikiama 10 paveikslėlyje. Pabrėžiama, kad sukonstruotas medis gali būti verčiamas atgal į HTML dokumentą, nepakeičiant originalios struktūros, turinio. Vadinasi, atliekamos modifikacijos neturi iškraipyti nesusijusių HTML elementų. Pagrindiniai AST mazgų tipai yra „tekstas“, „komentaras“, „apdorojimo instrukcija“ ir „elementas“. Šiuo atveju dėmesys buvo skiriamas „elemento“ atributams. Sukonstravus dokumento objektų modelio medį, specifiniai HTML elementai, atributai ir jų reikšmės buvo identifikuojami, vykdant rekursyvią paiešką į gylį, bei modifikuojami, priskiriant naujas reikšmes. Pagal „WebRanz“ techniką, pagrindinis tikslas yra perrašyti HTML elementų klasių bei ID atributų reikšmes į atsitiktines bei modifikuoti dokumento turinyje apibrėžtas nuorodas. Toliau apdorojami kiekvienas iš atributų bei nuorodos atskirai (apdorojimo tvarka nėra svarbi).

```
<ref *1> Document {
  parent: null, prev: null, next: null, startIndex: null, endIndex: null,
  children: [
    ProcessingInstruction {
      parent: [Circular *1], prev: null, next: [Element], startIndex: null, endIndex: null,
      data: '!DOCTYPE html', name: '!doctype', type: 'directive'
    },
    Element {
      parent: [Circular *1], prev: [ProcessingInstruction], next: null, startIndex: null, endIndex: null,
      children: [
        ...
        children: [
          <ref *1> [
            Text {
              parent: Element {
                parent: [Element], prev: [Element], next: null, startIndex: null, endIndex: null,
                children: [Circular *1],
                name: 'body', attribs: {}, type: 'tag'
              },
              prev: null, next: null, startIndex: null, endIndex: null,
              data: 'HTML5 pavyzdys', type: 'text'
            }
          ]
        ]
      ],
      name: 'html', attribs: {}, type: 'tag'
    }
  ],
  type: 'root'
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Pavyzdinis puslapis</title>
  </head>
  <body>
    HTML5 pavyzdys
  </body>
</html>
```

10 Pav. Pavyzdinė AST viršūnių (mazgų) struktūra. Tarpų ženklai (angl. *whitespace*) į pavyzdį neįtraukiami dėl pavyzdžio glaustumo

5.2.3. Atsitiktinių reikšmių generavimas ir kodavimas

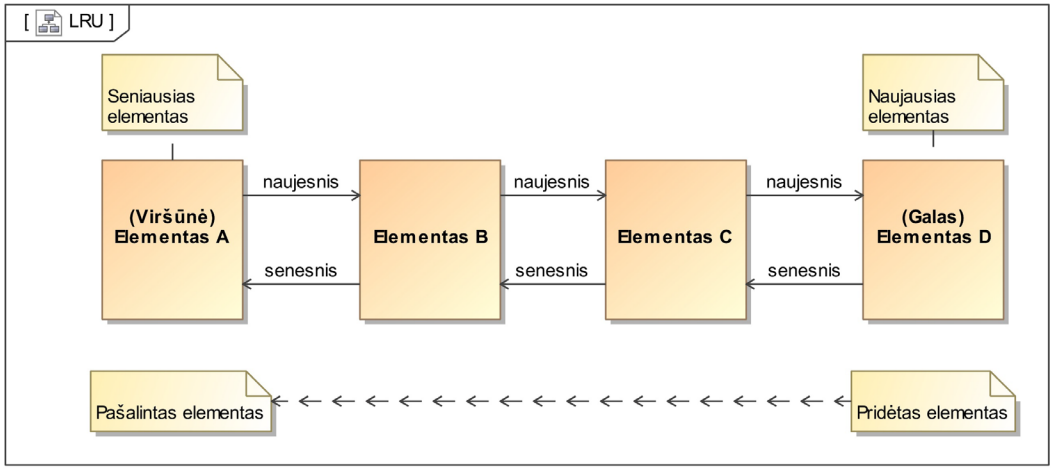
Priešingai nei „WebRanz“ autorių pateiktoje technikoje, atsitiktinės reikšmės buvo generuojamas tiek URL adresams, tiek atributams, remiantis viešojo rakto kriptografijos šifravimu.

Pasirinktas Rivest–Shamir–Adleman (toliau – RSA) algoritmas. Pasirinkimo priežastis yra, šifruojant žinutę, papildoma galimybė naudoti užpildymo schemą (angl. *padding scheme*). Schemos tikslas yra užtikrinti semantišką saugumą. Pavyzdžiui, taikant užpildymo schemą, galima, šifruojant tą pačią žinutę, gauti skirtingas reikšmes. Taigi, dvi užklausos į tą patį interneto svetainės tinklalapį grąžins skirtingai sugeneruotus atributus, nuorodas. Pastebima, kad prieš generuojant atsitiktines reikšmes, būtina reliatyvius URL adresus pakeisti jų absoliučiais analogais. Problema išryškėja, vykdant užklausą į interneto svetainės serverį. Šiuo atveju interneto naršyklė (klientas), HTML dokumente apibrėžtus reliatyvius URL adresus verčia absoliučiais pagal esamo tinklalapio URL adresą. Kadangi tiek esamo tinklalapio URL adresas, tiek nuorodos į kitus tinklalapius yra užšifruotos, klientas to padaryti negali. Taip pat nuorodos, vedančios į kitas interneto svetaines, nėra šifruojamas dėl prieš tai minėtų priežasčių. Pabrėžiama, kad atsitiktinai sugeneruotos reikšmės URL adresams negali turėti tokių simbolių kaip pliuso (+), pasvirojo brūkšnio (/) ir lygybės (=), kadangi tai yra rezervuoti simboliai, turintys prasmę pagal kontekstą. Taigi, kodavimui (angl. *encoding*) buvo naudojamas Base64URL standartas. Base64URL, taip pat, kaip ir Base64, yra sudarytas iš 64 simbolių kodavimui ir vieno neprivalomo specialaus simbolio, skirto užpildyti eilutę iki reikalaujamo ilgio. Užpildymas reikalaujamas tik tada, kai reikia dekoduoti dvi arba daugiau sujungtų (angl. *concatenated*) koduočių. Skirtumas tarp Base64 ir Base64URL yra tai, kad Base64URL keičia pliuso simbolį (+) į brūkšnį (–) ir pasvirąjį brūkšnį (/) į pabraukimą (_). Lygybės simbolis (=) nėra būtinas dekoduoti eilutei, tačiau gali būti keičiamas procentų simboliu (%). Vadinasi, atsitiktinai sugeneruotos reikšmės gali būti priskiriamos URL adresams, naudojant Base64URL kodavimą, ir ID, klasių atributų selektoriams, naudojant fiksuotą dalį koduotės, pavyzdžiui, pirmus dešimt simbolių. Pabrėžiama, kad nėra poreikio priskirti pilnos koduotės ID, klasių selektoriams, kadangi originalios reikšmės yra saugomos vienas prie vieno žemėlapyje.

5.2.4. HTML dokumento susiejimas su įterptais resursais

HTTP užklausų veikimo principas diktuoja, kad vienu metu galima grąžinti tik vieną resursą. Taigi, apdorojus HTML dokumento ID bei klasių atributų reikšmes, būtina turėti galimybę tas pačias reikšmes priskirti su dokumentu susijusiam pakopinių stilių šablonui. Tam pasiekti buvo pasirinkta kiekvieną reikšmę saugoti vienas prie vieno žemėlapyje, kuris atitinkamai saugomas mažiausiai neseniai naudotos (angl. *Least Recently Used;* toliau – LRU) duomenų struktūros podėlyje (angl. *cache*). LRU veikimo principas yra vaizduojamas 11 paveikslėlyje. Pagrindinė idėja yra turėti apibrėžto dydžio saugyklą, kur seniausiai naudotas elementas yra pašalinamas, pasiekus maksimaliai

leidžiamą saugyklos dydį. Šiuo atveju buvo pasirinkta kaip saugyklos raktą laikyti laiko žymą (angl. *timestamp*), kada klientas padarė pirmą užklausą į pasirinkto tinklalapio HTML dokumentą. Vadinasi, siunčiant papildomas užklausas kiekvienam įterptam resursui, klientas privalo į HTTP užklausą įtraukti ir sugeneruotą laiko žymą. Laiko žyma yra grąžinama kartu su HTML dokumentu kaip slapukas HTTP užklausos antraštėje. Kadangi slapuko reikšmė yra naudojama užkrauti tik vienam specifiniam interneto svetainės tinklalapiui, gyvavimo laikas gali būti apibrėžtas nežymiai trumpas. Sutikimo dėl tokio slapuko saugojimo nėra reikalaujama gauti iš kliento. T. y. tenkinamos būtinos sąlygos, kad slapukas yra naudojamas tik su tikslu vykdyti informacijos perdavimą tinklu ir griežtai reikalingas teikti paslaugą, kurios vartotojas išreikštinai reikalauja [Coo22]. Priskyrus atsitiktinai sugeneruotas reikšmes pagal viešojo rakto kriptografiją, į dokumento `<head>` elementą yra įtraukiami papildomi skriptai, reikalingi, pagal „WebRanz“ techniką, perrašyti „Javascript“ atributų selektorius, naudojamus dinamiškai įterpiamam turiniui. Pabrėžiama, kad įtraukiami skriptai prieš tai yra transformuojami (angl. *obfuscated*) į naują reprezentaciją, kurią sunkiau tiek modifikuoti, tiek perpanaudoti savais tikslais.



11 Pav. LRU duomenų struktūra grįsto podėliavimo veikimo principas

5.2.5. Pakopinių stilių šablono modifikavimas

Atlikus HTML dokumento transformaciją, abstrakčios sintaksės medis yra verčiamas atgal į simbolių eilutę ir grąžinamas klientui. Kitas žingsnis apima pakopinių stilių šablono modifikavimą. Gavus užklausą į atitinkamą pakopinių stilių šabloną, iš užklausos ištraukiamas laiko žymą nurodantis slapukas. Laiko žymos pagalba išgaunami klasių bei ID selektorių vienas prie vieno žemėlapiai iš LRU podėlio. Toliau, nuskaityta simbolių eilutė verčiama į abstrakčios sintaksės medį ir vykdoma rekursyvi paieška į gylį. Konkrečiai buvo analizuojami dviejų tipų AST objektai: taisyklės,

nurodančios specifinius selektorius, ir „@media“ užklausos, skirtos pritaikyti skirtingus stilius priklausomai nuo įrenginio savybių, pavyzdžiui, ekrano matmenų. Taip pat ta pati logika gali būti pritaikoma ir kitoms užklausoms, kaip „@supports“, „@page“ ir t. t. Bendru atveju stipriai rekomenduojama vienam interneto svetainės tinklalapiui skirti vieną pakopinių stilių šabloną. To priežastis yra tai, kad modifikuojami tik HTML dokumente nurodyti ID bei klasių selektoriai. Pabrėžiama, kad kiekvienas selektorius gali turėti sudėtingą logiką, todėl papildomai buvo sudaromas abstrakčios sintaksės medis selektoriaus tokenams, kurie apibrėžia elementų, turinčių klasių, ID atributus, vaizdavimą. Vienas iš pagrindinių iššūkių, taikant „WebRanz“ techniką, yra pakopinių stilių šablone apibrėžtų atributų selektorių modifikavimas. HTML dokumento generavimo metu, ID bei klasės atributams sugeneruotos atsitiktinės reikšmės yra skirtos sutampančioms selektorių poroms. Taigi, atributų selektoriai, kurie elementams priskiriami pagal tam tikro atributo reikšmę ar jos dalį, tampa neefektyvūs. „WebRanz“ technika šio aspekto nepadengia. Toliau pateikiamas realizuotas sprendimas. Prieš atsitiktinai atkuriant tinklalapių turinį, nurodytiems HTML dokumento elementams buvo pridėti pagal poreikį pritaikomi (angl. *custom*) „data–*“ atributai. Pavyzdinis sprendimas pateikiamas 12 paveikslėlyje. Tokiu būdu ieškoma vertė gali pilnai pakeisti visų tipų atributų selektorius ir būti tiesiogiai siejama su priskirtu atributu. Kadangi toks modifikavimas atliekamas tik vieną kartą, iki kol nepasikeičia konkretaus tinklalapio statinis turinys, modifikuoti failai gali būti saugomi operatyvioje atmintyje.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Pavyzdinis puslapis</title>
  </head>
  <body>
    <div class="row">
      <div class="col-sm-8 h1">
        <small> HTML5 pavyzdys </small>
      </div>
    </div>
  </body>
</html>

.row [class*="col-"].no-margin {
  margin-left: 0;
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Pavyzdinis puslapis</title>
  </head>
  <body>
    <div class="row">
      <div class="col-sm-8 h1" data-tkyzky>
        <small> HTML5 pavyzdys </small>
      </div>
    </div>
  </body>
</html>

.row [data-tkyzky].no-margin {
  margin-left: 0;
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Pavyzdinis puslapis</title>
  </head>
  <body>
    <div class="LRyuPkkD5o">
      <div class="tXyifX5egc h1" data-tkyzky>
        <small> HTML5 pavyzdys </small>
      </div>
    </div>
  </body>
</html>

.LRyuPkkD5o [data-tkyzky].no-margin {
  margin-left: 0;
}
```

12 Pav. Klasės selektoriaus perrašymas, vykdant paiešką pagal atributų selektorius

5.3. Integravimo ir plečiamumo galimybių įvertinimas

Remiantis bendro lankstumo taško – skaidraus tarpinio serverio – privalumais, galima teigti, kad įgyvendintas atsitiktinio tinklalapių turinio atkūrimo sprendimas gali būti pilnai izoliuotas nuo svetainės leidėjo (angl. *publisher*) serverio, nekeičiant pasiūlytų įgyvendinimo principų. Priešingai

nei įvardinama „WebRanz“ technikos aprašyme, HTML dokumentai bei susiję pakopinių stilių šablonai yra apdorojami atskirai (ne poromis). Taip pat vienas prie vieno žemėlapiai yra laikomi individualiuose LRU podėliuose (angl. *cache*). Taigi, toks atskyrimas leidžia užtikrinti dvipusį srautą tarp interneto svetainės lankytojų ir leidėjo nieko nežinant apie taikomą prevenciją. Konkrečiai tinklalapiai gali būti laikomi skaidraus tarpinio serverio operatyvioje atmintyje, o dekoduotos užklausos su antraštėmis peradresuojamos leidėjui. Be to, pasiūlyto sprendimo nereikia iš anksto konfigūruoti priklausomai nuo svetainės. Vadinasi, galima tiek praplėsti komercinių prevencijos priemonių įgyvendinimą, tiek taikyti prevenciją izoliuotai. Praktiškumo aspektas yra argumentuojamas pateikto sprendimo paprastumu bei naudojamomis viešojo kodo bibliotekomis (žr. 5.4 poskyrį). Pabrėžiama, kad integravimo prasme prevencijos taikymo tvarka yra svarbi, t. y. HTTP užklausos turi būti pirmą dekoduojamos ir tada peradresuojamos. Taigi, apjungus komercines prevencijos priemones kartu su pasiūlytu atsitiktinio tinklalapių turinio atkūrimo sprendimu, galima padengti visus nagrinėtus prevencijos etapus ir taip maksimaliai apsunkinti duomenų ištraukimo procesą laiko ir kainos atžvilgiu.

5.4. Naudojamos priemonės ir bibliotekos

Šiame poskyryje išvardinamos naudotos viešojo kodo bibliotekos, siekiant parodyti, kad atsitiktinio tinklalapių turinio atkūrimo techniką galima įgyvendinti be patentuotų įrankių, kurių įgyvendinimo detalės yra slepiamos.

- „htmlparser2“ – biblioteka, atsakinga už HTML dokumento simbolių eilutės vertimą (angl. *parsing*) į dokumento objektų modelį (`https://github.com/fb55/htmlparser2`).
- „cheerio“ – biblioteka, atsakinga už HTML dokumento specifinių elementų, atributų ir jų reikšmių identifikavimą, vykdant rekursyvią paiešką į gylį, bei naujų reikšmių priskyrimą. (`https://github.com/cheeriojs/cheerio`).
- „lru-cache“ – biblioteka, atsakinga už HTML dokumento atsitiktinai sugeneruotų ID, klasių selektorių reikšmių vienas prie vieno žemėlapių podėliavimą (`https://github.com/isaacs/node-lru-cache`).
- „javascript-obfuscator“ – biblioteka, atsakinga už dinamiškai įterpiamų skriptų, pagal “WebRanz” techniką, transformavimą (angl. *obfuscated*) į naują reprezentaciją, kurią sunkiau tiek modifikuoti, tiek perpanaudoti savais tikslais. (`https://github.com/javascript-obfuscator/javascript-obfuscator`).

- „css“ – biblioteka, atsakinga už pakopinių stilių šablono simbolių eilutės vertimą (angl. *parsing*) į abstrakčios sintaksės medį (`https://github.com/reworkcss/css`).
- „css-what“ – biblioteka, atsakinga už selektorių simbolių eilutės vertimą (angl. *parsing*) į abstrakčios sintaksės medį (`https://github.com/fb55/css-what`).
- „perimeterx-node-express“ – „PerimeterX“ biblioteka, skirta integruotis su prototipo tarpiniu skaidriu serveriu kaip tarpinė programinė įranga (angl. *middleware*) (`https://github.com/PerimeterX/perimeterx-node-express`).

5.5. Apibendrinimas

Apibrėžus ir įgyvendinus automatizuoto duomenų surinkimo prevencijos prototipą galima teigti, kad prevencijos taikymas kiekviename duomenų ištraukimo etape yra praktiškai pasiekiamas. Tiek iš teorinės, tiek iš praktinės pusės yra apibrėžti ir įgyvendinti atitinkami būdai, priemonės, kuriuos naudodamas interneto svetainės savininkas gali atskirti tikrus vartotojus nuo interneto robotų. Be to, įvertinta, kad skaidraus tarpinio serverio naudojimas užtikrina duomenų surinkimo prevencijos priemonių plečiamumo galimybes. Taigi, nustatyta, kad egzistuojančios prevencijos priemonės gali būti praplečiamos, siekiant padengti visus duomenų ištraukimo etapus. Įgyvendintas prototipas yra pasiekiamas „GitHub“ repozitorijoje: `https://github.com/alwaysintune/html-css-randomizer`

REZULTATAI IR IŠVADOS

Darbe pasiekti **rezultatai**:

1. Apžvelgti skirtingi automatizuoto duomenų surinkimo iš tinklalapių būdai:
   - Pristatytas saityno peržiūros robotas – veikimas, dažniausi panaudos atvejai, bei saityno duomenų ištraukėjas. Įvardinta, kokias problemas sprendžia saityno duomenų ištraukėjas. Pabrėžtas skirtumas tarp duomenų ištraukimo ir duomenų peržiūros atitinkamai išskiriant duomenų surinkimo tikslus.
   - Palyginti automatizuoto duomenų surinkimo iš tinklalapių būdai, apibendrinant jų skirtumus bei panašumus.
2. Apžvelgti bei palyginti automatizuoto duomenų surinkimo iš tinklalapių prevencijos būdai, įvertinant prevencijos taikymo galimybes iš teorinės pusės:
   - Detalizuotos duomenų ištraukėjo architektūros komponentų (interneto naršyklės valdymo sąsajos, įgaliotojo serverio, HTML analizatoriaus) funkcijos, veikimo sritis duomenų ištraukimo procese.
   - Išanalizuoti ir palyginti skirtingi prevencijos būdai (trijų sąrašų filtravimo technika, „honeypot“ (liet. *medaus puodas*), užklausos šaltinio indikatorių analizė, atsitiktinis tinklalapių turinio atkūrimas), įvardinant veikimą bei individualiai apibendrinant atitikimą apibrėžtiems kriterijams.
3. Apžvelgtos bei palygintos komercinės automatizuoto duomenų surinkimo prevencijos priemonės, įvertinant prevencijos taikymo galimybes iš praktinės pusės:
   - Pateikta kaip komercinės prevencijos priemonės sprendžia problemas, susijusias su pažangiausiomis priešpriešinės prevencijos technikomis.
   - Išanalizuotos ir palygintos skirtingos komercinės prevencijos priemonės („DataDome“, „PerimeterX“, „Kasada“), įvardinant veikimą bei individualiai apibendrinant atitikimą apibrėžtiems kriterijams.
4. Apibrėžtas ir įgyvendintas automatizuoto duomenų surinkimo prevencijos prototipas:
   - Įgyvendinta atsitiktinio tinklalapių turinio atkūrimo technika, naudojant skaidrų tarpinį serverį ir viešai prieinamas priemones, bibliotekas. Sprendimas apima abstrakčios sintaksės medžio sudarymą, atsitiktinių reikšmių generavimą, kodavimą į atitinkamą formatą ir reikšmių priskyrimo taisykles.

- Integruota „PerimeterX“ komercinė prevencijos priemonė kaip tarpinė programinė įranga (angl. *middleware*). Parodyta galimybė padengti visus nagrinėtus prevencijos duomenų ištraukimo etapus ir įvertintos duomenų surinkimo prevencijos priemonių plečiamumo galimybės.

Toliau pateikiamos darbo **išvados**:

1. Šiuo metu automatizuotą duomenų surinkimą riboti iš teisinės pusės yra sudėtinga. Nustatyta, kad techninių prevencijos būdų taikymas yra efektyvesnis sprendimas.
2. Atlikus analizę, nustatyta, kad prevencijos tikslas neturėtų būti visiškas duomenų ištraukimo apribojimas. Užtenka apsunkinti automatizavimą taip, kad ištraukti duomenis iš interneto svetainės būtų nepraktiška (laikas ir kaina).
3. Kadangi interneto naršyklė apie vartotoją pateikia pakankamai daug informacijos, galima teigti, kad unikalių identifikatorių sudarymas gali būti efektyviai naudojamas kaip alternatyva IP adresams identifikuojant interneto robotus.
4. Atsižvelgus į pažangą mašininio mokymosi ir kompiuterinio regėjimo srityse, įvertinta, kad tiuringo testai praranda savo svarbą ir turi būti keičiami alternatyvomis, pavyzdžiui, kriptografiniais iššūkiais.
5. Įvertinta, kad skaidrus tarpinis serveris yra būtina priemonė norint praplėsti komercines prevencijos priemones, taip įgyvendinant trūkstamą funkcionalumą.
6. Įgyvendinus automatizuoto duomenų surinkimo prevencijos prototipą, galima teigti, kad atsitiktinio tinklalapių turinio atkūrimo technika gali būti praktiškai naudojama rinkoje.
7. Darbo metu nustatyta, kad siekiant padengti kuo daugiau skirtingų interneto svetainės dalių, taikomi prevencijos būdai turi būti kombinuojami.

Apibendrinant gautus rezultatus ir išvadas, galima teigti, kad prevencija gali būti praktiškai taikoma prieš neetiškus interneto robotus.

TOLIMESNI DARBAI

Kaip tolimesnė ateities darbų kryptis siūloma įvertinti automatizuoto duomenų surinkimo prevencijos būdų taikymo galimybes specifiniam interneto svetainių porūšiui: kliento pusėje atvaizduojamoms interneto svetainėms. Šiuo atveju, praktiškai parodyti, pavyzdžiui, atsitiktinio tinklalapių turinio atkūrimo technikos galimybes dinamiškai keičiant API sąsajų grąžinamų objektų savybių raktus, reikšmes. Taip pat kartu taikant užklausos šaltinio indikatorių analizę atlikti autorizacijos funkcijai ir galimai pritaikant kriptografinį iššūkį su tikslu maksimaliai apsunkti duomenų ištraukimo procesą laiko ir kainos atžvilgiu.

ŠALTINIAI

[AJN+13] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, Bart Preneel. *FPDetective: Dusting the Web for Fingerprinters,* https://www.esat.kuleuven.be/cosic/publications/article-2334.pdf, 2013 (tikrinta 2022-05-28)

[ALS+06] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers – Principles, Techniques & Tools. Second Edition*. Pearson, 2006

[Ami21] Elad Amit. *PerimeterX Human Challenge Now Default Option for Bot Defender,* https://www.perimeterx.com/resources/blog/2021/perimeterx-human-challenge-now-default-option-for-bot-defender/, 2021 (tikrinta 2022-05-28)

[BB18] Seppe vanden Broucke, Bart Baesens. *Practical Web Scraping for Data Science*. Apress, 2018

[BS21] Itay Binder, Caty Sarle. *Expert Q&A: How to use honeypots to lure and trap bots,* https://www.perimeterx.com/resources/blog/2021/expert-q-a-how-to-use-honeypots-to-lure-and-trap-bots/, 2021 (tikrinta 2022-05-28)

[Bud20] Raluca Budiu. *Information Scent: How Users Decide Where to Go Next*, https://www.nngroup.com/articles/information-scent/, 2020 (tikrinta 2022-05-28)

[Cis20] Cisco. *What are Autonomous System Numbers?*, https://www.thousandeyes.com/learning/glossary/as-autonomous-system, 2020 (tikrinta 2022-05-28)

[Clo21] Cloudflare. *What is Zero Trust security?*, https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/, 2021 (tikrinta 2022-05-28)

[Con21] World Wide Web Consortium. *WebDriver*, https://www.w3.org/TR/webdriver/, 2021 (tikrinta 2022-05-28)

[Coo22] CookieYes. *Cookie Consent Exemptions: Strictly Necessary Cookies*, https://www.cookieyes.com/blog/cookie-consent-exemption-for-strictly-necessary-cookies/, 2022 (tikrintas 2022-05-28)

[Dat21] DataDome. *How DataDome works*, https://datadome.co/learning-center/how-datadome-works/, 2021 (tikrintas 2022-05-28)

[EFF20] Electronic Frontier Foundation. *Cover your tracks: See how trackers view your browser,* https://coveryourtracks.eff.org/, 2020 (tikrinta 2022-05-28)

[Goo21a] Google. *Googlebot*, https://developers.google.com/search/docs/advanced/crawling/googlebot, 2021 (tikrinta 2022-05-28)

[Goo21b] Google. *Overview of Google crawlers*, https://developers.google.com/search/docs/advanced/crawling/overview-google-crawlers, 2021 (tikrinta 2022-05-28)

[Has21] Erez Hasson. *Bad Bot Report 2021: The Pandemic of the Internet*, https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/, 2021 (tikrinta 2022-05-28)

[Hop20a] Computer Hope. *Dynamic website*, https://www.computerhope.com/jargon/d/dynasite.htm, 2020 (tikrinta 2022-05-28)

[Hop20b] Computer Hope. *Regex*, https://www.computerhope.com/jargon/r/regex.htm, 2020 (tikrinta 2022-05-28)

[HS15] Afzalul Haque, Sanjay Singh. *Anti-Scraping Application Development*, https://www.researchgate.net/profile/Sanjay-Singh-22/publication/281098321_Anti-Scraping_Application_Development/links/55d7501808ae9d65948d8a73/Anti-Scraping-Application-Development.pdf, 2015 (tikrinta 2022-05-28)

[IES21] Umar Iqbal, Steven Englehardt, Zubair Shafiq. *Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors,* https://www.computer.org/csdl/proceedings-article/sp/2021/893400a283/1mbmHGY5Lpu, 2021 (tikrinta 2022-05-28)

[IETF14] Internet Engineering Task Force. *Hypertext Transfer Protocol (HTTP/1.1),* https://www.rfc-editor.org/rfc/rfc7230#section-6.3.2, 2014 (tikrinta 2022-05-28)

[Jaw17] Dina Jawad. *Detection of Web API Content Scraping*, https://www.diva-portal.org/smash/get/diva2:1117695/FULLTEXT01.pdf, 2017 (tikrinta 2022-05-28)

[Jon20] JonasCz. *A guide to preventing Webscraping*, https://github.com/JonasCz/How-To-Prevent-Scraping, 2020 (tikrinta 2022-05-28)

[Kas22] Kasada. *Product and Technology: Flip the Script on Bots*, https://www.kasada.io/product-and-technology/, 2022 (tikrinta 2022-05-28)

[KJS20] Vlad Krotov, Leigh Johnson, Leiser Silva. *Tutorial: Legality and Ethics of Web Scraping*, https://digitalcommons.murraystate.edu/cgi/viewcontent.cgi?article=1071&context=faculty, 2020 (tikrinta 2022-05-28)

[Kos20] Martijn Koster. *About /robots.txt* https://www.robotstxt.org/robotstxt.html, 2020 (tikrinta 2022-05-28)

[Mck16] Sean F. McKenna. *Detection and Classification of Web Robots with Honeypots*, https://calhoun.nps.edu/bitstream/handle/10945/48562/16Mar_McKenna_Sean.pdf, 2016 (tikrinta 2022-05-28)

[Mis18] Sneha Mishra. *Introduction to DOM,* https://medium.com/nybles/introduction-to-dom-bee3b2dd9911, 2018 (tikrinta 2022-05-28)

[Moz21a] Mozilla. *User-Agent*, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent, 2021 (tikrinta 2022-05-28)

[Moz21b] Mozilla. *Definitions of Web-related terms Parse,* https://developer.mozilla.org/en-US/docs/Glossary/Parse, 2021 (tikrinta 2022-05-28)

[Moz21c] Mozilla. *Cascading Style Sheets display,* https://developer.mozilla.org/en-US/docs/Web/CSS/display, 2021 (tikrinta 2022-05-28)

[Moz21d] Mozilla. *DOMParser,* https://developer.mozilla.org/en-US/docs/Web/API/DOMParser, 2021 (tikrinta 2022-05-28)

[Pet21] Jeff Petters. *What is a Proxy Server and How Does it Work?*, https://www.varonis.com/blog/what-is-a-proxy-server/, 2021 (tikrinta 2022-05-28)

[Rog19] Armando Roggio. *SEO: How Google Reads and Renders JavaScript*, https://www.practicalecommerce.com/seo-how-google-reads-and-renders-javascript, 2019 (tikrinta 2022-05-28)

[Sno21] Snowflake. *Ad-Hoc Queries*, https://www.snowflake.com/data-warehousing-glossary/what-are-ad-hoc-queries/, 2021 (tikrinta 2022-05-28)

[Tax17] Taxilian. *Exact difference between add-ons, plugins and extensions,* https://stackoverflow.com/questions/33462500/exact-difference-between-add-ons-plugins-and-extensions, 2017 (tikrinta 2022-05-28)

[TO04] Boon Thau, Loo Owen. *Distributed Web Crawling over DHTs*, https://repository.upenn.edu/cgi/viewcontent.cgi?article=1345&context=cis_papers, 2004 (tikrinta 2022-05-28)

[UKD14] Trupti V. Udapure, Ravindra D. Kale, R. C. Dharmik. *Study of Web Crawler and its Different Types* https://www.iosrjournals.org/iosr-jce/papers/Vol16-issue1/Version-6/A016160105.pdf, 2014 (tikrinta 2022-05-28)

[URR+17] Joni Uitto, Sampsa Rauti, Samuel Rauti, Samuel Laurén, Ville Leppänen. *A Survey on Anti-honeypot and Anti-introspection Methods,* https://link.springer.com/chapter/10.1007/978-3-319-56538-5_13, 2017 (tikrinta 2022-01-17)

[Whi19] Zack Whittaker. *Bots are cheap and effective. One startup trolls them into going away,* https://techcrunch.com/2019/02/05/kasada-bots/, 2019 (tikrinta 2022-05-28)

[WZX+16] Weihang Wang, Yunhui Zheng, Xinyu Xing, Yonghwi Kwon, Xiangyu Zhang, Patrick Eugster. *WebRanz: web page randomization for better advertisement delivery and web-bot prevention,* https://dl.acm.org/doi/10.1145/2950290.2950352, 2016 (tikrinta 2022-05-28)

SAVOKŲ APIBRĖŽIMAI

Autonominė sistema (angl. *autonomous system*) – interneto nukreipiamo IP priešdėlių rinkinys, priklausantis tinklui arba tinklų rinkiniui, kurį valdo, kontroliuoja ir prižiūri vienas subjektas arba organizacija [Cis20].

Dokumento objektų modelio analizavimas (angl. *DOM parsing*) – „JavaScript“ programavimo sąsaja, atsakinga už HTML bei XML šaltinio kodo simbolių eilutės pavertimą į DOM dokumentą (analogija su HTML dokumentu) [Moz21d].

HTML semantinis elementas – HTML5 standarto elementas, skirtas tematiniam turinio grupavimui, tačiau funkciškai nesiskiriantis nuo paprasto `<div>` elemento.

Interneto robotas – terminas, apibūdinantis tiek saityno peržiūros robotą (angl. *web crawler*), tiek saityno duomenų ištraukėją (angl. *web scraper*).

Leksema – simbolių seka, kuri atitinka specifinio tokeno taisyklių, požymių rinkinį, t. y. yra egzempliorius (angl. *instance*) identifikuoto tokeno [ALS+06].

Reguliariojo reiškinio teksto eilutės atitikmens ieškojimas (angl. *regex matching*) – šablono, padedančio suderinti, rasti ir tvarkyti tekstą, atitikmens ieškojimas remiantis reguliariojo reiškinio sintakse [Hop20b].

Robotų atskyrimo metodas – priemonė, kuria galima identifikuoti požymius, kad interneto svetainės lankytojas yra robotas, pavyzdžiui, interneto srauto, apsilankymo dažnumo, intervalų bei aplankytų svetainių analizės, remiantis statistiniais metodais [HS15].

Saitynas – pasaulinis interneto svetainių bei kitų duomenų šaltinių tinklas.

Saityno duomenų šaltinis (angl. web assets) – pakopinių stilių šablonai bei „JavaScript“, nuotraukų failai, įprastai kartu užkraunami su interneto svetaine, vykdant HTTP užklausas, kaip tam tikro dydžio paketai.

Slapukas (angl. *web cookie*) – interneto svetainės sugeneruota informacija apie klientą, kuri išsaugoma kliento kompiuteryje su tikslu pakartotinai perpanaudoti informaciją tarp skirtingų svetainės apsilankymų.

Svetainės turinio organizavimas (angl. *content organization*) – viena iš priemonių, kuria siekiama tikslingai išpildyti vartotojo patirties reikalavimus, pasirenkant kaip skirtingi interneto svetainės tinklalapiai yra tarpusavyje susiję bei kokios struktūros turi būti laikomasi pateikiant turinį.

Tiuringo testas – kompiuterinė programa, skirta atlikti testą su tikslu atskirti ar interneto svetainės lankytojo sesiją sugeneravo robotas ar žmogus [Mck16], pavyzdžiui, pateikiant AI pilnas (angl. *AI-complete*) problemas, kaip objektų atpažinimą iš pateikto konteksto su stebimais iškraipymais (CAPTCHA, reCAPTCHA).

Tokenas (angl. *DOM Token*) – prasminga simbolių eilutė, atskirta kampiniais skliaustais, kaip „`<html>`“ ir „`<body>`“, su savo taisyklių rinkiniu, išskiriama iš HTML dokumento gautos baitų sekos, pavyzdžiui, vykdant HTTP užklausą, paverstos į vieną simbolių eilutę [Mis18].

Vartotojo agentas (angl. *user agent*) – HTTP užklausos antraštė, kuri leidžia serveriams ir kitiems tinklo mazgams (interneto prieigą turintiems įrenginiams) identifikuoti programą, operacinę sistemą, tiekėją ir (arba) užklausos vartotojo agento versiją [Moz21a].