VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VYLEPŠENÍ ROZŠÍŘENÍ PRO OMEZENÍ VOLÁNÍ JAVASCRIPTU

JAVASCRIPT RESTRICTING WEB EXTENSION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MARTIN TIMKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. LIBOR POLČÁK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Timko Martin, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Vylepšení rozšíření pro omezení volání JavaScriptu**
JavaScript Restricting Web Extension
Kategorie: Bezpečnost

Zadání:

1. Nastudujte rozhraní WebExtensions používané webovými prohlížeči.
2. Seznamte se s diplomovou prací Zbyňka Červinky, projektem JavaScript Zero a jinými nástroji pro vylepšení soukromí uživatelů webu.
3. Analyzujte nedostatky doplňku vytvořeném Zbyňkem Červinkou. Navrhněte vylepšení projektu a návrh konzultujte s vedoucím práce.
4. Implementujte vylepšení rozšíření prohlížeče.
5. Implementaci otestujte a zveřejněte ke stažení na serverech s doplňky pro příslušné prohlížeče.
6. Práci vyhodnoťte a navrhněte možná zlepšení.

Literatura:

- Schwarzm Michael aj. *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks.* Network and Distributed Systems Security Symposium 2018. San Diego, CA, USA.
- FRANKEN, Gertjan aj. Who Left Open the Cookie Jar? A Comprehensive Evaluation of Third-Party Cookie Policies. 27th USENIX Security Symposium, 2018, 978-1-931971-46-1, str. 151-168.
- ČERVINKA, Zbyněk. *Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Polčák Libor, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Abstrakt

Cieľom tejto práce je rozšíriť a funkčne vylepšiť prototyp webového rozšírenia vytvoreného Ing. Zbyňkom Červinkom, zamerané na ochranu súkromia užívateľa pri prehliadaní webu. V riešení boli využité nadobudnuté poznatky o fungovaní existujúcich nástrojov pre bezpečnosť a ochranu súkromia, ako napríklad technológia JavaScript Zero. Vytvorené riešenie pomocou techniky zapuzdrenia vhodných JavaScriptových objektov a funkcií, poskytuje užívateľom väčšiu anonymitu a ochranu pri prehliadaní webu. Rozšírenie bolo otestované a zverejnené pod názvom JavaScript Restrictor. Hlavným prínosom práce je zvýšenie ochrany súkromia užívateľa, ako aj zvýšenie bezpečnosti pred útokmi spojenými so zberom dát o užívateľoch.

Abstract

The purpose of this thesis is to functionally expand the browser extension prototype created by Ing. Zbyněk Červinka, focused on the privacy protection of the user during his web browsing. Acquired facts about the function of existing tools for safety and privacy protection, such as technology JavaScript Zero, were used in the solution, which was created by employing a technique of encapsulating JavaScript objects and functions. This allows greater anonymity and safety for users during web browsing. The extension was tested and published under the title JavaScript Restrictor. The main benefit of this thesis is the increase of safety from attacks and increase of anonymity linked to the user data harvesting.

Kľúčové slová

Ochrana súkromia užívateľov internetu, bezpečnosť užívateľov internetu, JavaScript Restrictor, JavaScript Zero, rozšírenie webového prehliadača, JavaScript, WebExtensions.

Keywords

Protection of internet user's privacy, the safety of internet users, JavaScript Restrictor, JavaScript Zero, web extension, JavaScript, WebExtensions.

Citácia

TIMKO, Martin. *Vylepšení rozšíření pro omezení volání JavaScriptu*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D.

Vylepšení rozšíření pro omezení volání JavaScriptu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Libora Polčáka, Ph.D. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Martin Timko
13. mája 2019

Poďakovanie

Rád by som sa poďakoval Ing. Liborovi Polčákovi, Ph.D. za vedenie tejto diplomovej práce, odbornú pomoc a cenné rady pri spracovaní tejto práce.

Obsah

Kapitola 1

Úvod

Bezpečnosť a súkromie používateľov pri navštevovaní webových stránok sa v dnešnej dobe stále viac a viac narúša. Keďže sú dnešné webové stránky často dynamické a ich súčasťou je aj JavaScript, práve zneužívanie JavaScriptu voči užívateľom predstavuje možné riziko. JavaScript je súčasťou až 95% webov z top 10 miliónov najobľúbenejších webov na svete [27] a pomocou JavaScriptu je možné zberať o užívateľoch rôzne osobné a citlivé dáta aj bez súhlasu užívateľa, ako aj vykonávať záškodné činnosti voči používateľom. Je preto potrebné chrániť užívateľov pred akýmikoľvek útokmi cez JavaScript.

Cieľom tejto diplomovej práce bolo naštudovať problematiku webových rozšírení, ako aj možnosti vylepšenia bezpečnosti a ochrany osobných údajov používateľov pomocou webových rozšírení. Následne vylepšiť prototyp webového rozšírenia Ing. Zbyňka Červinky [36], ktoré bude užívateľov chrániť pred zberom osobných dát, ako aj zabrániť istým druhom útokov cez JavaScript, pričom roršírenie by nemalo pokaziť funkcionalitu navštívených webových stránok. Toto vytvorené rozšírenie pod názvom JavaScript Restrictor zverejniť pre stiahnutie na serveroch s rozšíreniami pre príslušné prehliadače. Rozšírenie využíva princíp obaľovania JavaScriptových funkcií a objektov, čím dochádza k redefinovaniu pôvodných implementácii funkcií a objektov a vytvorí sa lokálna platnosť premenných v obálke. Týmto spôsobom sa zamedzí prístup k obalenej časti kódu z iných časti kódu a je následne možné modifikovať alebo blokovať navrátené hodnoty volaní obalených funkcií a objektov. Tento spôsob tak podáva skreslené informácie o užívateľoch, čím sa zvyšuje anonymita prehliadania webu a zároveň znemožňuje niektoré útoky pomocou JavaScriptu na používateľoch.

Najprv je v tejto práci v kapitole 2 predstavený Javascript, webové rozšírenia a ich tvorba. V kapitole 3 analyzujem projekt JavaScript Zero [30], ktorý sa zaoberá útokmi pomocou JavaScriptu a navrhuje riešenia pre zvýšenie bezpečnosti používateľov na webe formou webového rozšírenia. Následne v kapitole 4 analyzujem prototyp webového rozšírenia vytvoreného Ing. Zbyňkom Červinkom [36], z ktorého vychádza táto diplomová práca. Kapitola 5 sa venuje ďalším existujúcim rozšíreniam pre ochranu súkromia. Nasleduje popis návrhu rozšírenia v kapitole 6 a popis implementácie rozšírenia v kapitole 7, ako aj proces zverejnenia rozšírenia na príslušných oficiálnych obchodoch rozšírení webových prehliadačov. Nakoniec sa v práci nachádza v kapitole 8 popis otestovania výsledného rozšírenia.

Kapitola 2

Predstavenie jazyka JavaScript a webových rozšírení

Táto kapitola sa venuje programovaciemu jazyku Javascript[1] a webovým rozšíreniam. V kapitole nájdeme popis oblasti JavaScriptu relevantnej pre túto prácu. Ďalej sú v kapitole uvedené základné informácie čo rozšírenia predstavujú a aké sú možnosti využitia týchto rozšírení. Kapitola sa rovnako zaoberá aj technológiou WebExtensions[2], ktorá slúži pre tvorbu webových rozšírení naprieč viacerými webovými prehliadačmi. Diskutované sú hlavné aspekty, ktoré musí rozšírenie vytvárané pomocou WebExtensions API spĺňať. Kapitola sa taktiež zaoberá aj bezpečnostnou problematikou a spomenuté sú aj klady či zápory tejto technológie.

2.1 JavaScript

JavaScript nám umožňuje pridávať dynamické prvky na statické webové stránky. Tento koncept je dnes využívaný na takmer každej webovej stránke na ktorú narazíme. Tak ako to popisuje aj kniha autora Marijna Haverbekea [11], JavaScript je možné využiť vo všetkých najpoužívanejších prehliadačoch, ktoré sa dnes využívajú, a tým JavaScript umožnil vytvorenie dnešných moderných webových aplikácií. V tejto práci je využitie oblasti pôsobnosti premenných, ako aj zapuzdrenie funkcií v JavaScripte kľúčové. Preto si tieto oblasti pre lepšie pochopenie problematiky bližšie popíšeme.

2.1.1 Platnosť premenných

Oblasť pôsobnosti premenných definuje viditeľnosť a životnosť premennej. Platnosť premenených sa rozlišuje na lokálne a globálne [11]. Premenné majú platnosť v istej úrovni kódu, v ktorej boli deklarované. Ak bola premenná deklarovaná na najvyššej úrovni dokumentu, ide o globálnu premennú. Keďže JavaScript povoľuje nedeklarované premenné, globálna premenná je automaticky vytvorená z takejto nedeklarovanej premennej, nech už je v akomkoľvek bloku kódu alebo vo funkcii. Ku globálnym premenným má prístup každá funkcia či blok kódu v JavaScripte.

Naopak, lokálne premenné sú platné a dostupné iba v danom bloku kódu alebo funkcii. Tieto premenné vznikajú až po volaní funkcii, v ktorej sa nachádzajú. Po vykonaní funkcie,

[1] Skriptovací jazyk JavaScript https://www.techopedia.com/definition/3929/javascript-js

[2] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

lokálne premenné zanikajú. K týmto premenným nie je možné pristupovať globálne z iných častí kódu. Lokálna premenná sa vo funkcii, v ktorej bola deklarovaná, správa ako globálna. Teda pri zanorených funkciách vo funkciách sa javí ako globálna. Rovnako to platí pri zanorených blokoch kódu.

V JavaScripte vieme deklarovať premenné pomocou kľúčového slova `var` ako aj `let`. Rozdielom je oblasť platnosti danej deklarovanej premennej. `let` a `var` sa pri deklarovaní vo funkciách a pri deklarovaní globálnej premennej správajú podobne. Rozdielom je, že premenná deklarovaná pomocou `let` nie je dostupná cez globálny objekt `window`. Ďalším rozdielom je, že kým premenná deklarovaná pomocou `var` vo vnútri funkcie je dostupná kdekoľvek v rámci funkcie, aj keď bola deklarovaná napríklad vo vnorenom bloku funkcie, tak premenná deklarovaná pomocou `let` ma vždy platnosť a je dostupná iba v rámci daného bloku kódu a zároveň sa správa ako globálna premenná pre viac zanorené bloky z daného bloku kódu [34].

2.1.2 Zapuzdrenie

Zapuzdrenie alebo enkapsulácia či obaľovanie v JavaScripte je technika, pri ktorej vzniká takzvaná obálka úseku JavaScriptového kódu, a tým sa zamedzuje prístup k danému úseku kódu z iných častí kódu. Toto zamedzenie je spôsobené tým, že sa vytvorí oblasť s lokálnou platnosťou premenných a všetky premenné, objekty a funkcie v tejto oblasti teda nie sú prístupné globálne. Ako je to popísané v článku Jasona Shapira [32], často sa to využíva ak nechceme, aby nejaká časť kódu alebo dáta boli priamo viditeľné a dostupné.

Následujúci príklad ukazuje obaľovanie ľubovoľnej časti kódu v JavaScripte. Všetko v zapuzdrenom kóde tak nie je prístupné globálne a naberá iba lokálnu platnosť.

```
(function() {
    \\ kod ktory chceme zapuzdrit
    };
})();
```

Kód 2.1: Príklad zapuzdrenia časti kódu v JavaScripte.

2.2 Webové rozšírenie

Už ako samotný názov napovedá, webové rozšírenia rozširujú funkcie webového prehliadača. Rozšírenia pridávajú nové funkcie do samotného prehliadača. Tým sa vylepšuje funkcionalita a zároveň často krát sa uľahčuje a zlepšuje pocit z prehliadania webu zo strany užívateľa. Nasledujúce informácie som získal z webovej stránky dokumentácie MDN Web Docs [22]. Tieto rozšírenia sa vytvárajú na základe najčastejšie vyskytujúcich sa technológií na webe, a teda vytvárajú sa pomocou štandardného značkovacieho jazyka HTML[3], mechanizmu na vizuálne formátovanie HTML CSS[3] a taktiež pomocou programovacieho jazyka JavaScript.

Ako je aj uvedené v knihe Prateeka Metha [15], je nutné poznamenať, že rozšírenia nie sú zásuvné moduly (známe tiež pod anglickým výrazom *plug-in*) do prehliadačov. Rozšírenia bežia v prehliadači v rámci bezpečnostného mechanizmu známom pod pojmom sandbox, ktorý slúži na oddeľovanie bežiacich procesov a virtualizáciu tak, aby škodlivý softvér nemohol poškodiť hostiteľský softvér. Naopak zásuvné moduly nie sú sandboxované a často

[3]Viac informácií o HTML a CSS na https://www.w3.org/standards/webdesign/htmlcss

je spúšťaný priamo natívny kód. Aj vďaka tomu, sú rozšírenia oproti zásuvným modulom považované za bezpečnejšie.

2.2.1 Možnosti využitia webových rozšírení

Dnešné spektrum webových rozšírení je obrovské. Po nainštalovaní do prehliadača, rozšírenia ponúkajú užívateľom mnohé funkcie. Jednotlivé prehliadače ponúkajú svoje vlastné obchody pre rozšírenia ako napríklad Firefox Add-ons alebo Chrome Web Store. Výber a inštalácia rozšírení je tak pre užívateľa veľmi dostupná a jednoduchá. Rozšírenia umožňujú meniť a upravovať obsah webových stránok, ako napríklad farbu pozadia[4] podľa potrieb užívateľa. Ďalej nám rozšírenia ponúkajú pridávať, ale aj odstrániť obsah stránky. Napríklad blokovať nechcenú reklamu[5] alebo naopak reformátovať obsah webu pre lepšie čítanie textov[6]. Rozšírenia umožňujú pridávať nové funkcionality do prehliadača pre bežného užívateľa, ako aj nástroje pre webových vývojárov[7]. Kategória rozšírení, ktorá nás zaujíma v tejto práci najviac je ale kategória rozšírení na ochranu súkromia užívateľa a zvýšenie bezpečnosti pri prehliadaní webu. V ďalších kapitolách sú ukázané a rozvinuté niektoré zaujímavé príklady takýchto rozšírení. Príklady známych rozšírení pre zvýšenie bezpečnosti pri prehliadaní webu nájdeme v kapitole 5. Prístupy na ich tvorbu v kapitolách 3 a 4.

2.3 WebExtensions API

WebExtensions je technológia určená pre vývoj rozšírení [17]. Ide o najmodernejšiu dostupnú technológiu vyvinutú spoločnosťou Google, ktorú si postupne osvojujú všetky moderné webové prehliadače. V tejto chvíli je WebExtensions podporovaná vo všetkých popredných prehliadačoch ako Google Chrome, Mozilla Firefox, Opera, Microsoft Edge. Výnimku tvorí prehliadač Safari vyvíjaný spoločnosťou Apple. Od novembra 2017 je WebExtensions oficiálne hlavnou a jedinou technológiou, ktorá je využívaná pre vývoj rozšírení pre Mozillu Firefox, ktorá dovtedy využívala iné nástroje či technológie ako XUL/XPCOM alebo Add-on SDK. Konkrétnejšie porovnanie starších technológií je dostupná na stránkach mozilly MDN[8]. Rozšírenia vyvíjané pomocou starších nástrojov je ale stále možné upraviť a preniesť (anglicky *porting*) na technológiu WebExtensions. Rozšírenia vytvorené pomocou WebExtensions bez zmien alebo s malými obmenami fungujú na všetkých zmienených prehliadačoch. Ide teda o multiplatformovú technológiu (anglicky termín *cross-platform*).

2.3.1 Súborová štruktúra

Informácie o súborovej štruktúre rozšírenia boli čerpané ako z dokumentácie MDN [18], tak aj z knihy Prateeka Mehta [15], ktorá sa zaoberá vytváraním rozšírení pre Google Chrome. Rozšírenie vytvorené pomocou WebExtensions pozostáva z kolekcie súborov. Ide o HTML, CSS, JavaScript súbory, ako aj obrázky a iné potrebné súbory, ktoré dané rozšírenie potrebuje pre svoju plnú funkčnosť. Aby tieto súbory tvorili rozšírenie, musia spĺňať isté zákonitosti.

[4] Príklad webového rozšírenia na https://addons.mozilla.org/en-US/firefox/addon/myweb-new-tab/

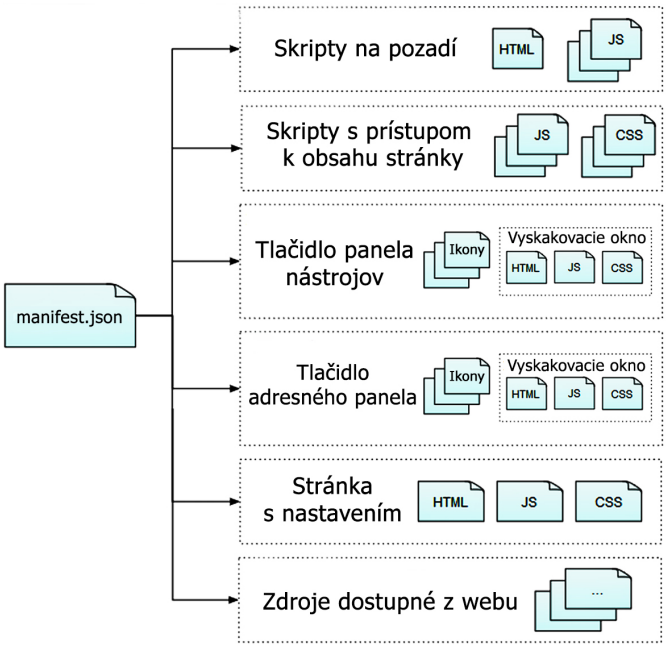
[5] Príklad webového rozšírenia na https://addons.opera.com/sk/extensions/details/adblock-plus/

[6] Príklad webového rozšírenia na https://addons.mozilla.org/en-US/firefox/addon/reader/

[7] Príklad webového rozšírenia na https://addons.mozilla.org/en-US/firefox/addon/firebug/

[8] Viac informácií na https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Comparison_with_XUL_XPCOM_extensions a https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Comparison_with_the_Add-on_SDK

Jediným povinným súborom v každom rozšírení je `manifest.json`. Tento súbor obsahuje základne údaje a metadáta o rozšírení. Konkrétnejšie v tomto súbore nájdeme informácie ako názov rozšírenia, jeho verziu, popis rozšírenia, ako aj oprávnenia, ktoré rozšírenie vyžaduje na správnu funkcionalitu v prehliadači. Ide o súbor vo formáte JSON[9]. `manifest.json` obsahuje odkazy na ďalšie súbory spojené s rozšírením. Tieto súbory, na ktoré sa môžeme pozerať ako na komponenty samotného rozšírenia sú znázornené na obrázku 2.1.



Obr. 2.1: Základné komponenty technológie WebExtensions.[10]

Ide o šesť hlavných komponent:

- **Skripty na pozadí** (anglicky *Background scripts*)**:** Jedná sa o dlhodobú logiku na pozadí rozšírenia. Skripty na pozadí sú JavaScriptové kódy, ktoré sa vykonávajú na pozadí načítanej webovej stránky. Skripty ostávajú načítané od chvíle nahrania rozšírenia a ostávajú načítané pokiaľ sa rozšírenie neodinštaluje alebo nezakáže. Tieto skripty nepristupujú k webovej stránke ako takej, ale majú prístup ku všetkým rozhraniam WebExstensions API.

- **Skripty s prístupom k obsahu stránky** (*Content scripts*)**:** Tieto skripty by sme mohli charakterizovať ako skripty, ktoré majú prístup ku konkrétnemu obsahu danej stránky. Takéto skripty bežia v kontexte konkrétnej webovej stránky, kde oproti skriptom na pozadí majú výrazne menší prístup k rozhraniam WebExtensions API, no pomocou správ vedia tieto skripty komunikovať so skriptami na pozadí.

[9] Viac informácií o JSON na https://www.json.org/

[10] Prevzaté z https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension a upravené v GIMP.

- **Tlačidlo panela nástrojov** (*Browser action*)**:** Takto sa označuje tlačidlo, ktoré sa zobrazuje užívateľovi v prehliadači v paneli nástrojov. Po kliknutí sa otvorí stavové okno, ktoré sa označuje ako vyskakovacie okno, tiež známe pod anglickým pojmom *pop-up*. Okno po zadefinovaní ponúka užívateľovi zobraziť dané informácie, ktoré rozšírenie ponúka alebo užívateľ môže cez okno interagovať s rozšírením. Príklad takýchto tlačidiel na obrázku 2.2.

Obr. 2.2: Ukážka tlačidiel Browser action.[11]

- **Tlačidlo adresného panela** (*Page action*)**:** Veľmi podobné tlačidlo ako tlačidlo panela nástrojov. Avšak na rozdiel od tlačidla panela nástrojov je toto tlačidlo adresného panela viazané na konkrétne webové stránky. Po nastavení rozšírenia je tak graficky znázornené, že na danej stránke je rozšírenie aktívne alebo naopak nie je pre daný web tlačidlo špecifikované. Toto grafické znázornenie sa líši podľa prehliadača. Google Chrome to rieši zmenou farby ikonky tlačidla. Zmenou farieb do odtieňov sivej indikuje, že rozšírenie sa na danej stránke nepoužíva.

- **Stránka s nastavením** (*Options page*)**:** Na užívateľské nastavenia rozšírenia slúži práve stránka s nastavením. Rozšírenie si užívateľ môže nastaviť aj cez vyskakovacie okno, no odporúčaný prístup je vytvorenie stránky s možnosťami, kde užívateľ má prístup k nastaveniam rozšírenia.

- **Zdroje dostupné z webu** (*Web accessible resources*)**:** HTML, CSS a JavaScriptové zdroje, ktoré sú dostupné z webu a sú zahrnuté v rozšírení, no nie sú pribalené vo forme súborov v samotnom rozšírení, sa nazývajú zdroje dostupné na webe. Takto dokáže rozšírenie napríklad vložiť obrázok na webovú stránku, pričom tento obrázok nie je súčasťou zdrojových súborov rozšírenia.

V skriptoch je možné využiť WebExtensions úložisko pre ukladanie ako aj načítanie dát. Čítanie a zápis sa deje asynchrónne a väčšinou sa možnosť úložiska využíva pre ukladanie nastavení rozšírenia. Uložené dáta sú vo formáte JSON.

WebExtensions nám umožňuje cez skripty rozšírenia zachytávať HTTP žiadosti a získať prístup k hlavičkám týchto žiadostí, ako aj k telu danej žiadosti pomocou WebRequest API [21]. Je tak možné upraviť hlavičky v HTTP žiadostiach alebo aj celkom zrušiť odoslanie žiadosti. Pre prácu so žiadosťami je nutné najprv špecifikovať udalosť, na ktorej sa bude daná akcia (napríklad modifikovanie hlavičiek) vykonávať. Tieto udalosti sa líšia podľa toho, kedy je nutné pozastaviť HTTP žiadosť. WebRequest API umožňuje upravovať žiadosť pred samotným odoslaním aj pri prijatí a taktiež umožňuje presmerovať odosielanú alebo prijatú žiadosť na inú URL.

2.3.2 Bezpečnostné aspekty WebExtensions API

WebExtensions API ponúka spôsoby, ako vytvárať bezpečné rozšírenia. Webové rozšírenia majú prístup k veľkej časti činností, ktoré prehliadač vykonáva. Rovnako na pozadí môže

[11] Prevzaté z https://developer.chrome.com/extensions/user_interface

rozšírenie vykonávať mnohé činnosti a odosielať požiadavky na servery tretích strán. Rozšírenia teda môžu slúžiť aj ako akási forma škodlivého softvéru teda malvéru[12]. Škodlivé rozšírenia sú prítomné aj na oficiálnych obchodoch, ktoré rozšírenia ponúkajú [24].

Prvá úroveň bezpečnosti rozšírení je ponúkaná automaticky, a to pomocou oprávnení rozšírenia. Pomocou oprávnení vieme limitovať prístup rozšírenia k špecifikovaným API, a teda ak napríklad rozšírenie pracuje iba so záložkami prehliadača a je kompromitované škodlivým softvérom, tak oprávnenia nedovolia rozšíreniu prístup k histórii alebo otvoreným kartám prehliadača.

Ak rozšírenie komunikuje s externými webmi a vymieňajú si dáta pomocou správ, je potrebné špecifikovať túto komunikáciu v súbore `manifest.json` pomocou atribútu `externally_connectable`. Atribút umožňuje detailne nastavenie špecifických URL čím sa zamedzí nevyžiadaným správam posielaným rozšíreniu a definuje zdroje, s ktorými má rozšírenie povolenie komunikovať.

Jednou z nevýhod WebExtensions API je, že ak rozšírenie ukladá, načíta a pracuje s dátami, tak tieto dáta nie sú šifrované. Rovnako si treba dávať pozor aj pri písaní skriptov s prístupom k obsahu stránky (*content scripts*). Je dôležité, aby rozšírenie, ktoré zasahuje do webovej stránky nezanieslo na danú stránku zraniteľné miesta. To znamená, že v prípade, že rozšírenie prijíma dáta zo servera, je potrebné ošetriť dáta pred útokmi typu cross-site scripting[13]. Ďalšie bezpečnostné riziko nastáva pri spustenom skripte, ktorý má prístup k obsahu stránky, v prípade, že rozšírenie chce upravovať škodlivú stránku. Pri nesprávnom použití metódy na parsovanie dát stránky, môže dôjsť k umožneniu vykonávania škodlivého kódu pomocou rozšírenia. Prateek Metha [15] uvádza ako príklad nesprávne použitie metódy `eval` pre parsovanie dát. Namiesto toho, je vhodnejšie použiť `JSON.parse` pre zachovanie bezpečnosti.

[12] Vysvetlenie pojmu malvér dostupné na https://www.eset.com/sk/malver/

[13] Viac informácii o Cross-site Scripting (XSS) útokoch na https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Kapitola 3

JavaScript Zero

Celá táto kapitola je venovaná práci *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks* [30].

JavaScript Zero obsahuje súbor opatrení pre zabránenie útočenia cez JavaScript, ktorý útočníci využívajú pre útoky pomocou škodlivých webových stránok. Demonštráciou tohto modelu je webové rozšírenie nazvané Chrome Zero, ktoré autori implementovali a má za úlohu znemožniť útoky útočníkov.

Práca JavaScript Zero [30] sa skladá z niekoľkých častí. Najprv sú analyzované aktuálne útoky, ktoré samotný JavaScript umožňuje. Ide o mikroarchitekturálne útoky (anglicky *microarchitectural attacks*) ako aj útoky postrannými kanálmi, taktiež známe pod anglickým pojmom *side-channel attacks*, ktoré sú popísané v sekcii 3.1. Následne sa pozrieme ako takýmto útokom predchádzať a aké protiopatrenia musia byť zavedené, aby sa predišlo konkrétnym útokom, a to v sekcii 3.2. V sekcii 3.3 sa pozrieme aj na to, ako sa autori rozhodli tieto protiopatrenia implementovať a prečo zvolili formu webového rozšírenia. Na konci tejto kapitoly v sekcii 3.4 sú zhrnuté výsledky celého konceptu JavaScript Zero, ako aj jeho výhody či nevýhody.

3.1 Analýza známych útokov

Táto sekcia je venovaná známym útokom postrannými kanálmi, ako aj mikroarchitekturálnymi útokmi. Dnešné moderné procesory sú vysoko optimalizované a efektívne vo výpočtoch. Mikroarchitekturálne útoky a útoky postranných kanálov vedia využiť tieto skutočnosti a pomocou mikroarchitekturálnych elementov ako napríklad zberníc, rýchlej vyrovnávacej pamäte (anglicky *cache*) alebo DRAM môžu odhaliť utajované informácie. Napríklad pri meraní času pri kryptografických operáciach potrebného pre nájdenie záznamu v pamäti cache, známym pod anglickým pojmom *cache hit*, alebo naopak v prípade, ak sa takýto záznam v pamäti nenájde (*cache miss*) je možné pomocou štatistickej analýzy vyčítať celý kľúč použitý pri šifrovaní informácie [5]. Autori práce [30] analyzovali jednotlivé požiadavky jedenástich známych útokov a rozdelili tieto požiadavky do piatich kategórií. Každý útok pritom vyžaduje aspoň jednu požiadavku z týchto kategórií.

Kategórie požiadavok na útoky:

- pamäťové adresy,
- presné načasovanie,

- viacvláknové procesy,
- zdieľané dáta,
- senzorové API.

3.1.1 Pamäťové adresy

JavaScript ako programovací jazyk neponúka programátorovi koncept ukazovateľov do pamäte ani žiadny iný spôsob odkrytia virtuálnych adries pamäte. Útočník preto musí voliť iný spôsob ako sa dostať k virtuálnym adresám pamäti. Najbližšie k virtuálnej pamäti majú polia nazvané `ArrayBuffers` [30]. Sú to bloky virtuálnej pamäti používané podobne ako bežné polia, no sú efektívnejšie. Ak sa podarí identifikovať jednu adresu virtuálnej pamäte, je možné tým identifikovať zbytok pamäte, pretože `ArrayBuffers` sú lineárne polia a prehliadače alokujú `ArrayBuffers` vždy zarovnane vzhľadom na začiatok stránky v pamäti [9].

Pre útoky zamerané na DRAM, vie útočník využiť fakt, že prehliadače používajú na alokáciu pamäti `mmap`, ktorý alokuje 2 MB namiesto 4 KB veľkosti stránok v pamäti [10]. Následne pri iterácii cez tieto dáta dochádza k chybám pri indexácii stránok (*page faults*) na začiatku novej stránky. Čas potrebný pre vyriešenie tejto chyby je dlhší, ako keď sa iba pristupuje do pamäti. Týmto spôsobom vie útočník zistiť index novej stránky [10].

3.1.2 Presné načasovanie

Takmer každý mikroarchitekturálny útok aj útoky postranných kanálov využívajú presné načasovanie (anglicky *accurate timing*). Ide o spôsob merania časových rozdielov pri istých úlohách ako napríklad pri zásahoch do pamäte DRAM alebo časových rozdieloch prístupu do rýchlej vyrovnávacej pamäte [10]. Tieto merania časových rozdielov môžu byť podľa potreby útočníka v rámci nanosekúnd až milisekúnd.

JavaScript ponúka pre meranie času objekt `Date`, ktorý meria čas v milisekundách a objekt `Performance`, ktorý pomocou metódy `performance.now()` vracia hodnoty v mikrosekundách. Pre útočníkov, ktorí často potrebujú väčšiu jemnosť, je výhodnejšie implementovať vlastný časový merač s ešte väčšou jemnosťou. Väčšinou vytvorí útočník istý čítač, ktorý využíva monotónnu inkrementáciu počítadla, pomocou ktorého je možné merať časové rozdiely až v nanosekundách [31].

3.1.3 Viacvláknové procesy

Pojem *multithreading* alebo vykonávanie procesu viacerými vláknami procesora súčasne, nebolo v JavaScripte možné, kým sa neobjavila posledná verzia HTML. HTML5 zaviedol viaceré vlákna nazývané ako *web workers*. Týmto sa do JavaScriptu zaviedli nové možnosti útokov cez postranné kanály.

Ako Vila a kol. [35] ukázali, pomocou merania časových závislostí vo fronte udalostí (*event queue*) vo vlákne, je možné identifikovať používateľov vstup, ako aj identifikovať stránky načítané v iných oknách prehliadača.

3.1.4 Zdieľané dáta

Zavedením *web workers* do JavaSciptu bolo potrebné zaviesť komunikáciu medzi vláknami. Zavedením poľa s názvom `SharedArrayBuffer` však vznikajú aj nové bezpečnostné hrozby

ako využiť zdieľanú komunikáciu. Útočník si vie pomocou nekonečného cyklu s jednoduchým pripočítavaním hodnôt nepriamo vypočítať časové rozdiely s jemnosťou až 2 nanosekundy [31]. Tento spôsob tak umožňuje vykonávať útoky postranných kanálov na rýchlu vyrovnávaciu pamäť alebo DRAM.

3.1.5 Senzorové API, batéria

Keďže JavaScript je dostupný aj na mobilných zariadeniach, od HTML5 je možné pracovať aj so senzormi zariadenia. Senzory sa ale tiež dokážu využiť pre útoky, pretože pre niektoré senzory (napríklad senzor pohybu, senzor pre snímanie úrovne svetla) nie je potrebné explicitné potvrdenie pre užívanie senzora. Článok autrov Mehrnezhad a kol. [14] ukázal, že pomocou senzorov pre zaznamenávanie pohybov a orientácie bolo možné odvodiť PIN kód obete útoku alebo bolo možné zistiť jednotivé dotykové gestá používateľa, ako napríklad priblíženie obrazu prstami.

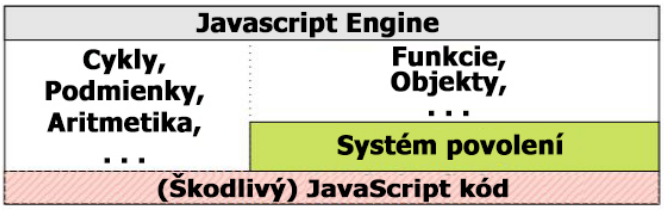
Nakoniec sa autori pozreli aj na stav batérie zariadenia, z ktorého je možné získať informácie o stave dobíjania či vybíjania batérie. Ďalej je možné získať predpoklad, kedy bude batéria plne nabitá, ale aj samotnú úroveň batérie, teda na koľko percent je batéria aktuálne nabitá.

3.2 Protiopatrenia

Navrhované protiopatrenia autormi JavaScript Zero [30] majú pokrývať všetky spomínané kategórie požiadavok na útoky. Ďalej by nemali byť viditeľné pre programátorov webových aplikácií a mali by byť implementované na čo najnižšej úrovni, teda na mikroarchitekturálnej úrovni. Zamedzenie všetkým útokom vo všetkých kategóriách je aj dobrým predpokladom pre zamedzenie budúcim útokom.

Autori práce [30] predpokladajú, že útočník je schopný kontrolovať bežiaci kód JavaScriptu v prehliadači používateľa. Naopak práca sa nezaoberá možnosťami využívať chyby v interpreteri jazyka JavaScript alebo chyby spojené so samotným sandboxom.

JavaScript Zero predstavuje abstraktnú vrstvu medzi samotným JavaScript engine, ktorý vykonáva JavaScriptový kód a rozhraním poskytnutým potencionálnemu útočníkovi alebo vývojárovi. Hlavnou myšlienkou je zabezpečiť funkcie, rozhrania ako aj vlastnosti objektov. Táto abstraktná vrstva vie jednoducho zachytiť a presmerovať všetku interakciu s kódom samotného enginu alebo ju modifikovať, či úplne zablokovať. Vrstva je transparentná pre webové aplikácie, a teda nie je potrebné modifikovať žiaden kód aplikácií, aby JavaScript Zero fungoval. JavaScript Zero ale nezasahuje do samotných riadiacich štruktúr jazyka, ako napríklad cykly, podmienky alebo do primitívnych dátových typov.



Obr. 3.1: Znázornená abstraktná vrstva predstavujúca JavaScript Zero a jeho systém povolení.[2]

3.2.1 Systém povolení

Systém povolení predstavuje možnosť nastavenia jednej zo štyroch možností pre každú JavaScript funkciu.

Pre každú funkciu je možné zvoliť:

- **Povoliť:** Funkcia je explicitne povolená alebo nešpecifikovaná v systéme povolení. Funkcia sa vykonáva bez akéhokoľvek zásahu.
- **Blokovať:** Funkcia je blokovaná. JavaScript Zero teda nahrádza vykonávanie funkcie a vracia iba dopredu predvolené hodnoty.
- **Modifikovať:** Funkcia nie je blokovaná úplne, ale na základe užívateľom zadaných parametrov sa jej vykonávanie upravuje a funkcia vracia upravené hodnoty.
- **Používateľove povolenie:** JavaScript Zero pozastaví všetkú činnosť funkcie a zobrazí notifikáciu užívateľovi, ktorý rozhodne o povolení vykonávania funkcie.

Pôvodne autori zamýšľali implementovať aj možnosť užívateľovi nastaviť pre jednotlivé funkcie vlastné povolenia. Napríklad nastaviť, že funkcia `history.back` bude úplne blokovaná alebo aké bude použité zaokrúhľovanie pri použití metódy `performance.now()`. Nakoniec ale ostal systém povolení iba v jednoduchšej verzii. Užívateľ si môže zvoliť jednú z pripravených úrovní zabezpečenia, a to na úrovne: **žiadna, malá, stredná, vysoká, paranoidná.** Každá úroveň má nastavené predvolené nastavenia pre jednotlivé funkcie na spomínané hodnoty ako povoliť, blokovať, modifikovať či zobraziť povolenie pre užívateľa. Umožňuje to rýchle nastavenie užívateľovi, ktorý nepotrebuje detailné nastavovania každej funkcie.

3.2.2 Konkrétne opatrenia

Z pohľadu mikroarchitektúry je pre funkčnosť tohto systému povolení potrebné zaviesť konkrétne opatrenia.

Proti zneužitiu pamäťových adries spomínaných v sekcii 3.1.1 je potrebné zabezpečiť, aby sa pamäť nejavila ako lineárna. Spolu s pridávaním umelo vytvorenými náhodnými prístupmi do pamäti, ako aj rýchlej vyrovnávajúcej pamäti je možné predísť mnohým útokom [30].

Znížením presnosti merania času funkcie `performance.now()` sa dosiahne zamedzenie mnohým útokom zameraných na presné načasovanie popísaných v sekcii 3.1.2 [30]. Rovnako sa dá zamedziť útokom tejto kategórie pridaním náhodných meškaní do vykonávania funkcií, ktoré útočníkovi znemožní získavanie potrebných presných časových meraní [30].

Najlepším opatrením pre zamedzenie útokov spojené s viacvláknovými procesmi spomenutých v sekcii 3.1.3 je úplne zakázať takzvaný *multithreading*. Keďže by šlo o zásah pri vykonávaní niektorých aplikácii, autorom [30] neostáva iná možnosť ako povoliť viacvláknové procesy. Niektoré útoky sa dajú zamedziť pridávaním náhodných meškaní pri posielaní správ medzi vláknami.

Rovnako najlepším opatrením pri bezpečnostných rizikách zdieľaných dát uvedených v sekcii 3.1.4 by bolo podľa [30] ich úplne blokovanie. Útoky využívajúce rýchly opakovaný prístup do `SharedArrayBuffer` je možné zamedziť spomalením prístupu do pamäte, čím sa zníži schopnosť útočníka presne merať časové závislosti v hodnotách mikrosekúnd.

[2]Prevzaté z [30] a upravené v GIMP.

Nakoniec autori [30] navrhujú protiopatrenia pre senzorové API zmienené v sekcii 3.1.5. Napríklad znížiť frekvenciu a presnosť senzorov a vždy vyžadovať používateľove povolenie pred spustením všetkých senzorov, pretože na útoky je možné využívať mnohé senzory, ktoré sa nemusia javiť ako nebezpečné.

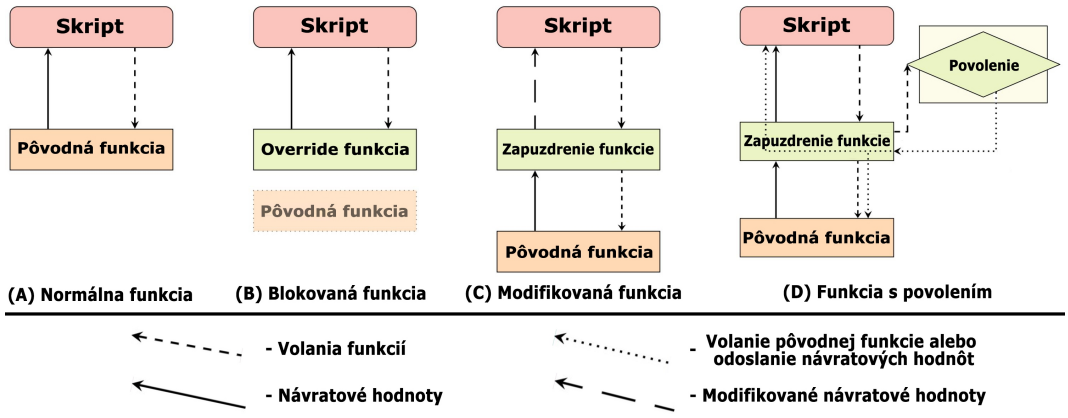
3.3 Chrome Zero

Najlepší spôsob, ako tieto protiopatrenia implementovať, je priama implementácia do samotného jadra prehliadača. Tento spôsob by bol ale veľmi náročný a bola by nutná spolupráca vývojárov jednotlivých prehliadačov. Autori [30] chceli generickejšie riešenie pre viaceré prehliadače, a preto autori prišli s riešením v podobe webového rozšírenia.

Webové rozšírenie Chrome Zero[3] bolo vyvíjané pre webový prehliadač Google Chrome verzie 49. Tvorcovia chceli vytvoriť rozšírenie, ktoré by pri aplikovaní protiopatrení vzdorovalo a nedalo sa obísť pri záškodnej činnosti. Ďalej aby zabraňovalo útočníkom útočiť na rozšírenie samotné a zároveň aby toto rozšírenie nemalo markantný vplyv pre užívateľskú spokojnosť (známa pod anglickým pojmom *user experience*) pri prezeraní webu.

3.3.1 Implementácia

Hlavnou myšlienkou je pre implementáciu použiť techniku známu pod pojmom *virtual machine layering*, čo sa dá preložiť ako vrstvenie virtuálnych strojov. Táto technika zaručuje, že sa po zavolaní funkcie zavolá modifikovaná funkcia namiesto pôvodnej bez modifikovania samotného prehliadača. Keďže funkcie sú nahradzované priamo v kóde, nie je možné ich obísť a zavolať tak pôvodnú funkciu. Pre túto techniku sa využívajú možnosti JavaScriptu pre redefinovanie a zapuzdrenie pôvodnej implementácie (anglicky *wrapper*). Skrátene môžeme hovoriť o zapuzdrení alebo obaľovaní. Tento princíp sa využíva pre obaľovanie daných funkcií v rozšírení. Pôvodná funkcia sa tak vyjme z globálneho priestoru platnosti a je dostupná iba v obalenej funkcii. Pri volaní funkcií skriptom webu sa dostáva k slovu systém povolení. Obrázok 3.2 ukazuje princíp systému povolení. Podľa zvolenej úrovne zabezpečenia sa zvolí daná akcia (povoliť, blokovať, modifikovať, zobraziť povolenie) pre konkrétnu volanú funkciu v skripte (napríklad `performance.now`). Tieto povolenia sú uložené vo formáte JSON.



Obr. 3.2: Nasledujúci obrázok ilustruje systém povolení pre funkcie s použitím techniky zapuzdrenia funkcií.[4]

[3] Dostupné na https://github.com/IAIK/ChromeZero

Aby bolo možné aplikovať jednotlivé povolenia na objekty a ich vlastnosti, Chrome Zero na to využíva takzvaný proxy objekt. Proxy objekt obsahuje pôvodný objekt a predáva mu všetky volania smerujúce na pôvodný objekt. Stačí redefinovať v proxy objekte iba tie funkcie, na ktoré sa vzťahuje systém povolení, a tým sa aplikuje systém povolení na objekty.

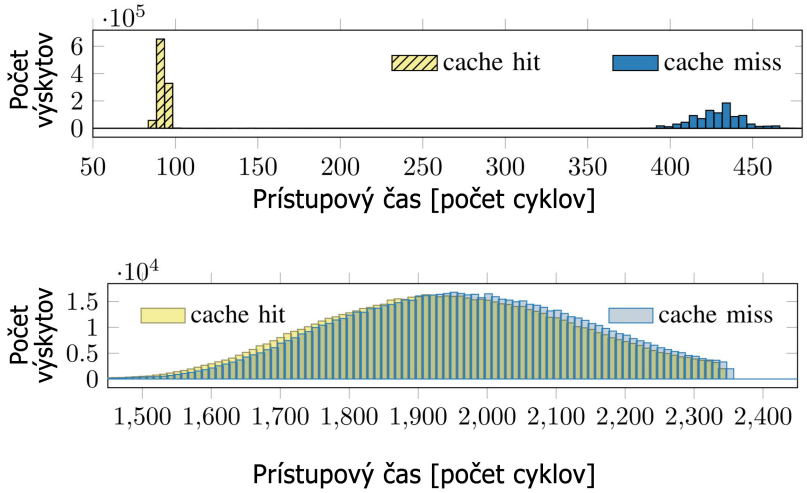
Útočník nie je schopný pristúpiť k pôvodnej funkcii, ak je pri implementácii vhodne využitá technika zapuzdrenia. Pri zapuzdrení funkcií, ktoré nepatria žiadnemu objektu, nie je možné zvonku pristúpiť ku kódu. Rovnako sa všetky premenné pri obaľovaní stávajú lokálnymi a nie je možné k nim pristúpiť ani pomocou globálneho objektu `window`, cez ktorý je možné pristupovať ku globálnym premenným.

Nasledujúci kód predstavuje ukážku zapuzdrenia funkcie `performance.now` so znížením presnosti merania času. Meno pôvodnej funkcie odkazuje na novú funkciu. Pôvodná funkcia sa volá pomocou referencie a prístup k nej je iba v rámci tohto priestoru. Táto referencia ale nie je viditeľná mimo tohto lokálneho priestoru, rámca.

```
(function() {
    var original = window.performance.now;
    window.performance.now = function() {
        return Math.floor((original.call(window.performance)) / 1000.0) * 1000.0;
    };
})();
```

Kód 3.1: Príklad zapuzdrenia funkcie `performance.now`.

Autori implementovali všetky popísané opatrenia v sekcii 3.2.2. Keďže sa jedná o mnoho opatrení, uvediem iba príklad. Pre útoky spojené so zneužitím pamäťových adries autori vytvorili proxy objekty pre všetky typy polí v JavaScripte (napríklad `Unit8Array`, `Unit16Array`) aby sa reguloval prístup k týmto objektom. Ďalej napríklad pri zvedení náhodnej prístupovej doby do rýchlej vyrovnávajúcej pamäti je pre útočníka nemožné vyčítať, kedy nastal cache hit alebo cache miss (obrázok 3.3). Pre kompletný popis opatrení viď. [30].



Obr. 3.3: Porovnanie prístupov do rýchlej vyrovnávajúcej pamäti bez pridania náhodnéj prístupovej doby (hore) a s pridaním náhodnej prístupovej doby (dole).[5]

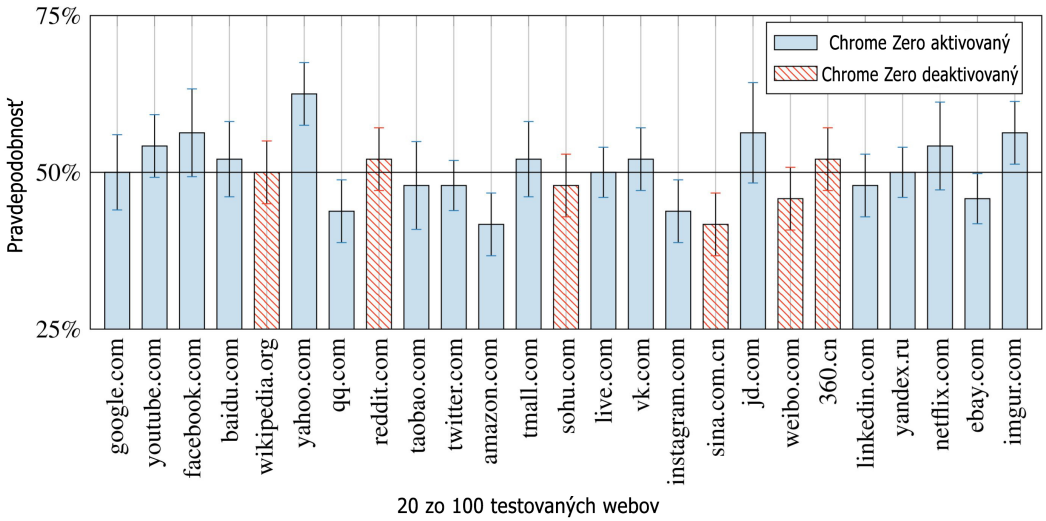
[4]Prevzaté z [30] a upravené v GIMP.
[5]Prevzaté z [30] a upravené v GIMP.

3.4 Zhodnotenie JavaScript Zero

Autori JavaScript Zero implementovali jednotlivé opatrenia a vytvorili webové rozšírenie nazvané Chrome Zero. Toto rozšírenie chráni užívateľov od rôznych útokov postrannými kanálmi, ako aj pred mikroarchitekturálnymi útokmi.

Nameraná latencia načítania webových stránok sa pohybovala medzi 10, 64 milisekúnd pri vypnutých všetkých povoleniach až po 89, 08 milisekúnd pri zapnutých všetkých bezpečnostných prvkoch, ktoré Chrome Zero ponúka.

Autori rozšírenie testovali s pomocou dobrovoľníkov, kde mali účastníci za úlohu prezerať najpoužívanejšie weby podľa rebríčka *Alexa Top 100 websites*. Chrome Zero bol pre jednotlivé weby náhodne spúšťaný alebo naopak nebol aktívny. Následne účastníci odpovedali, či pozorovali nejaké výkonnostné rozdiely pri prezeraní webu. Testovanie ukázalo, že pravdepodobnosť, kedy účastník správne identifikoval web so spusteným Chrome Zero, sa pohybovala okolo 50%. To znamená, že až na pár výnimiek (napríklad web *amazon.com* obrázok 3.4), účastník nevedel rozlíšiť, či je Chrome Zero spustený alebo nie. Teda dopad spusteného rozšírenia Chrome Zero je pre bežného užívateľa nepostrehnuteľný.



Obr. 3.4: Pravdepodobnosť používateľov správne identifikovanej prítomnosti Chrome Zero pri prezeraní daného webu.[7]

3.4.1 Výhody a nevýhody JavaScript Zero

JavaScript Zero ponúka riešenie pre zamedzenie útočenia cez JavaScript na úmyselne škodlivých alebo napadnutých webových stránkach. Jeho výhodou je veľká flexibilita pre zamedzenie všetkých útokov, ktoré boli v dobe vzniku JavaScript Zero známe. Výhodou je aj systém povolení, ktorý autori zaviedli a implementácia, ktorá nevyžaduje žiadne zásahy do prehliadača, aby rozšírenie Chrome Zero fungovalo správne. Ponuka piatich úrovní zabezpečenia je vhodná pre ľahké a rýchle nastavenie používateľov rozšírenia Chrome Zero.

V rozšírení chýba nastavenie povolení osobitne pre jednotlivé weby. Vytvorenie vlastných stupňov zabezpečenia užívateľom s možnosťou vlastného nastavenia pre každú funkciu osobitne taktiež nie je možné. Dôvodom je absencia rozšírených nastavení okrem spomínaných

[7] Prevzaté z [30] a upravené v GIMP.

piatich úrovní zabezpečenia. Ďalej Chrome Zero je dostupný iba na prehliadačoch Google Chrome a keďže toto rozšírenie nie je ponúkané cez oficiálny obchod rozšírení, inštalácia je o to viac sťažená. Dôvodom je, že Google Chrome sa snaží všetky rozšírenia, ktoré nepochádzajú z oficiálneho obchodu z bezpečnostných dôvodov blokovať. Pri testovaní rozšírenia sa naviac zistilo, že napríklad API pre blokovanie alebo modifikovanie stavu batérie nefunguje, nakoľko je stále možné cez Battery Status API pomocou JavaScriptu zistiť úroveň a stav batérie zariadenia. Taktiež pri nastavení najvyššej úrovne ochrany pri niektorých weboch[8] dochádzalo k spomaleniu celého prehliadača, až k úplnému zamrznutiu.

[8]Napríklad https://jsfiddle.net/

Kapitola 4

JavaScript Restricter - rozšírenie zamerané na ochranu súkromia

Cieľom tejto práce je vylepšenie webového rozšírenia JavaScript Restricter[1], ktoré vytvoril Ing. Zbyněk Červinka vo svojej diplomovej práci [36] a bolo vyvíjané pre prehliadač Mozilla Firefox. Táto kapitola sa zaoberá práve rozšírením JavaScript Restricter. V tejto kapitole sa v sekcii 4.1 pozrieme na motiváciu, prečo toto rozšírenie vzniklo. V ďalšej sekcii 4.2 je uvedený návrh a implementácia rozšírenia. Nezabudneme ani na návrh užívateľského rozhrania v sekcii 4.3. Nakoniec sa v sekcii 4.4 pozrieme na testovanie rozšírenia.

Rozšírenie JavaScript Restricter si kladie za úlohu zvýšiť súkromie, ako aj anonymitu používateľov pri prezeraní webu. Jeho jedinečnosť v porovnaní s inými dostupnými rozšíreniami (popísané v kapitole 5 tejto práce) je tá, že neblokuje úplne JavaScript, ale technikou zapuzdrenia JavaScriptového kódu využívané funkcie iba modifikuje. Pri kompletnom zablokovaní JavaScriptu často dochádza k nefunkčnosti webových stránok, keďže sa dnes JavaScript často využíva ako kľúčová komponenta webov pre ich správne fungovanie.

4.1 Monitorovanie osobných a citlivých informácií

Zvýšiť súkromie používateľov je možné pomocou zabránenia alebo skreslenia údajov, ktoré o používateľovi daný web zbiera. Často sa jedná o osobné údaje. Osobné a citlivé informácie používateľov sú informácie, ktoré môžu byť použité k identifikácii osôb [36]. Teda okrem samotného mena alebo adresy aj napríklad e-mail, telefóny, zdravotné údaje, politické alebo náboženské presvedčenia či genetické a biometrické údaje [7]. V našom prípade v IT sfére ide vlastne o dáta, pomocou ktorých je možné identifikovať danú osobu alebo osoby. Môžeme tak k citlivým dátam doplniť aj napríklad dáta popisujúce polohu užívateľa, alebo informácie o pohybe myši či klikoch na reklamy a podobne.

Taktiež je dobré si uvedomiť, že zber osobných dát o používateľovi neznamená automaticky, že sa jedná o záškodnú činnosť pre daného používateľa. Ak sa osobné a citlivé informácie snaží získať útočník pomocou škodlivého webu s cieľom poškodiť osobu, vtedy sa dá hovoriť o záškodnej činnosti. Táto činnosť môže napríklad spočívať v zhromažďovaní osobných údajov a následnom zverejnení citlivých údajov danej osoby. Účelom môže byť celková diskreditácia osoby.

Naopak, dnes sa často osobné údaje využívajú v mnohých službách na internete. O možnostiach využitia hovorí aj článok od autora Alexei Kounine [12]. Personalizované reklamy

[1] Rozšírenie pre Mozillu Firefox dostupné na https://github.com/cervinka-zbynek/masters-thesis

na webe nie sú vlastne nič iné ako pozbierané osobné údaje o používateľoch, na základe ktorých sú užívateľom ponúkané reklamy na webových stránkach.

Existuje viacero možností, ako takéto dáta získať. Diplomová práca [36] popisuje dve základné možnosti pre získavanie informácií o užívateľoch. Pomocou cookies alebo pomocou takzvaných session replay skriptov.

4.1.1 Cookies

Krátke textové súbory, ktoré sa ukladajú lokálne na strane používateľa pri navštívení stránky sa nazývajú cookies [13]. Cookies slúžia často pre zjednodušenie prístupu k webom. Pomocou cookies sa často nastavujú používateľské predvoľby.

Často sú dáta na strane klienta, teda používateľa, ukladané alebo modifikované bez jeho vedomia. Tým, že je protokol HTTP nestavový, je preto tento nástroj dôležitý pre webové stránky, tak aj pre užívateľov, pretože zjednodušuje prehliadanie webu. Na druhej strane cookies môžu slúžiť aj k monitorovaniu používateľov [36]. Pomocou cookies sa dajú získať informácie o navštívených stránkach alebo kliknutiach na reklamy. Možnosti cookies sú ale obmedzené, a preto nie je možné získavať dáta o užívateľovi v reálnom čase alebo detailnejšie podrobnosti o používaní webu. To ale nič nemení na fakte, že sa cookies dajú použiť na identifikáciu užívateľa na internete [23].

4.1.2 Session replay skripty

Druhou možnosťou, ako sledovať správanie užívateľa na webe, a tým zbierať o ňom údaje, je využiť k tomu takzvané session replay skripty. Pod týmto názvom si máme predstaviť skripty alebo nástroje, ktoré vedia komplexne zaznamenať akcie užívateľa na webe alebo webovej aplikácii a tieto informácie uchovávajú [8]. Tieto nástroje často využívajú napríklad poskytovatelia e-shopov, aby vedeli, ako sa správa zákazník na ich webe a podľa toho mohli upravovať model svojho e-shopu.

Pomocou DOM objektov webovej stránky sú zaznamenávané pohyby myšou, kliknutia na dané objekty stránky, ako napríklad reklamy. Zbierajú aj dáta o úderoch do klávesnice a podobne. Všetky tieto udalosti je možné zbierať pomocou JavaScriptu, ktorý na to ponúka nástroje a knižnice. Príkladom služby pre analýzu webov a aplikácií, kde sa využíva monitorovanie používateľa v reálnom čase je Smartlook[2] [36]. Smartlook využíva JavaScript pre sledovanie krokov používateľa, na základe ktorých analyzuje chovanie užívateľov.

4.2 Implementácia rozšírenia

Hlavný princíp implementácie rozšírenia JavaScript Restricter je rovnaký ako pri Chrome Zero (kapitola 3). Ide o techniku zapuzdrenia popísanú v sekcii 2.1.2. Vytvorením zapuzdrenia obalený kód naberá lokálnu platnosť. Teda nie je dostupný pre kód mimo obálky, čiže pre priamu prácu s funkciami alebo objektmi. Manipulovať s týmto kódom je možné iba vo vnútri obálky. Keďže k invokácii tohto zapuzdreného JavaScriptového kódu dochádza v dobe, kedy ešte prehliadač nezačal spracovávať zdrojový kód načítavajúcej sa webovej stránky, žiaden kód okrem nášho nebude mať prístup k pôvodnej implementácii [36].

Pre každú obaľovanú funkciu alebo objekt si pomocou užívateľského rozhrania rozšírenia, popísaného v nasledujúcej sekcii 4.3, je možné zvoliť do akej miery sa daný objekt alebo funkcia bude modifikovať.

[2]Kvalitatívna analytika pre webové stránky dostupná na https://www.smartlook.com/

V diplomovej práci [36] sú obaľované nasledovné objekty a funkcie jazyka JavaScript:

- **Objekt `window.Date`:** tento objekt v sebe nesie časové údaje. Jedná sa o dátum a čas s presnosťou na milisekundy. Zapuzdrením objektu bude možné znížiť presnosť časových dát. Používateľ si môže v nastaveniach rozšírenia zvoliť, do akej miery budú tieto dáta skreslené. Užívateľ môže zvoliť zaokrúhlenie s presnosťou na desiatky, stovky milisekúnd alebo na celé sekundy.

- **Funkcia `window.performance.now()`:** funkcia `window.performance.now()` sa nachádza v objekte `window.performance`. Funkcia `window.performance.now()` vracia časové hodnoty až v mikrosekundách. Podobne ako v prípade `window.Date` bude úlohou rozšírenia modifikovať a zaokrúhliť vracajúce hodnoty. Taktiež bude možné pomocou stránky nastavenia nastaviť presnosť zaokrúhľovania na desiatky, stovky alebo tisíce milisekúnd. Ako bolo spomínané v sekcii 3.1.2, tieto objekty `Date` a `Performance` v JavaScripte je možné využiť pre útoky spojené s presným načasovaním. Taktiež práca [26] ukázala, že je možné identifikovať zariadenie pomocou *clock-skew* časových posunov. Obalenie objektu `Date` a funkcie `performance.now()` tak dokáže eliminovať túto metódu identifikácie.

- **Funkcia `window.HTMLCanvasElement.prototype.getContext()`:** HTML elementy, ktoré sú typu Canvas môžu byť používané ako grafické elementy pre kreslenie grafiky alebo animácií a podobne. Ako uvádza článok [16], pomocou týchto elementov je možné získať napríklad popisy (anglicky *fingerprint*) grafických kariet a identifikovať tak užívateľa. Obalením funkcie `getContext()` sa zaistí kontrola nad vykresľovaním do HTML elementov. Užívateľ má na stránke nastavenia možnosť úplne zakázať tento zápis do HTML elementov Canvas, čím vie obmedziť zbieranie informácií o užívateľovom systéme alebo možnosť opýtať sa užívateľa pri každom pokuse o zápis do týchto HTML elementov či povoľuje tento zápis.

- **Funkcia `navigator.geolocation.getCurrentPosition()`:** v JavaScriptovom objekte `navigator.geolocation` nájdeme funkciu `getCurrentPosition()`, ktorá vie vrátiť aktuálnu polohu zariadenia. Vhodným zapuzdrením funkcie je možné znížiť presnosť dát pre lokalizáciu používateľa. Pomocou stránky nastavenia si vie užívateľ nastaviť, s akou presnosťou sa dané GPS atribúty zaokrúhľujú. Ide o zemepisnú šírku, zemepisnú dĺžku vrátane presnosti, s akou sú tieto súradnice zbierané. Ďalej sa dá nastaviť presnosť merania nadmorskej výšky, ako aj jej zaokrúhľovanie. K tomu je možné nastaviť aj zaokrúhľovanie informácií ohľadom smerovania zariadenia a rýchlosti pohybu zariadenia v danom smere, kam smeruje. Keďže meranie týchto atribútov obsahuje aj časové razítka, podľa nastavenia objektu `window.Date` sa automaticky nastaví aj presnosť časového razítka pri týchto GPS meraniach. Užívateľ má tiež možnosť zvoliť úplné zablokovanie, kedy budú dané atribúty vracať nulové hodnoty.

- **Objekt `window.XMLHttpRequest`:** Keďže `XMLHttpRequest` žiadosti môžu obsahovať niektoré osobné dáta, ktoré sa zbierajú a odosielajú bez užívateľovho vedomia, obalením tohto objektu je možné odchytiť tieto žiadosti. Je preto možné zobraziť, povoliť či blokovať dotazy. Používateľ si tak vie nastaviť, že buď sa úplne blokujú všetky `XMLHttpRequest` žiadosti alebo sa pre každú žiadosť zobrazia používateľovi podrobnosti o žiadosti. Následne môže užívateľ povoliť alebo zablokovať danú žiadosť.

Zapuzdrením spomínaných objektov tak vieme zamedziť identifikácii pomocou presného načasovania (`window.Date, window.performance.now()`), pomocou geolokácie (funkcia

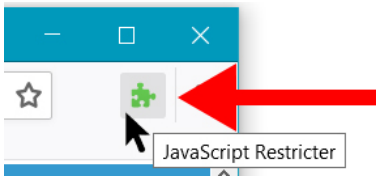
`getCurrentPosition()`), ako aj zamedziť hardvérové stopy grafických kariet užívateľov (blokovania HTML elementov Canvas). Taktiež blokovaním `window.XMLHttpRequest` vieme zamedziť nežiadúcemu odosielaniu žiadostí `XMLHttpRequest` [36]. Nasledujúci príklad kódu prevzatý z [36] ukazuje príklad zapuzdrenia funkcie `window.Date`:

```
(function() {
       var originalDateObject = window.Date;
       window.Date = function() {
           var myDate = new originalDateObject();
           var roundedValue = Math.floor(myDate.getMilliseconds()/100) * 100;
           myDate.setMilliseconds(roundedValue);
       return myDate;
       };
   }
)();
```

Kód 4.1: Príklad zapuzdrenia funkcie `window.Date`.

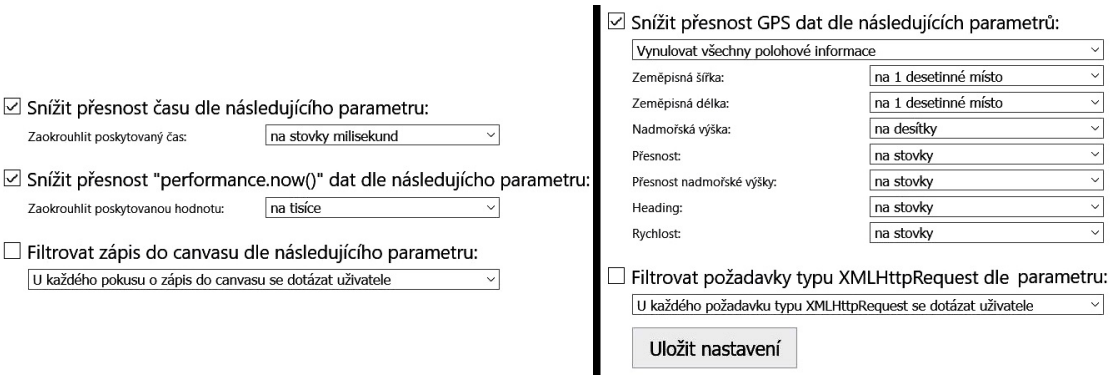
4.3 Užívateľské rozhranie rozšírenia

Užívateľské rozhranie obsahuje tlačidlo panela nástrojov (anglicky *Browser action*) na obrázku, cez ktoré sa po stlačení dostaneme na konfiguračnú stránku, teda stránku s nastaveniami rozšírenia.



Obr. 4.1: Tlačidlo panela nástrojov.

Stránka s nastavením obsahuje nastavenia pre zapuzdrené konštrukcie jazyka. Pre každý objekt alebo funkciu je možnosť nastavenia daných hodnôt, poprípade je možné úplne vynechať konkrétne nastavenie v prípade, že sa neodškrtne checkbox daného nastavenia. Tieto nastavenia sú zachytené na nasledujúcom obrázku 4.2.



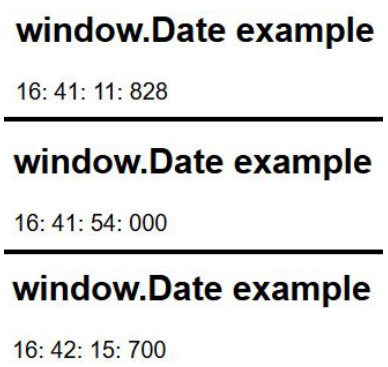
Obr. 4.2: Všetky nastavenia dosptuné v rozšírení JavaScript Restricter. (Naľavo nájdeme vrchnú časť nastavení a napravo spodnú časť nastavení na stránke nastavenia.)

4.4 Testovanie rozšírenia

Sekcia sa venuje testovaniu rozšírenia JavaScript Restricter, ktoré bolo navrhnuté a implementované v rámci diplomovej práce Ing. Zbyňka Červinky [36].

Rozšírenie JavaScript Restricter bolo testované tromi spôsobmi:

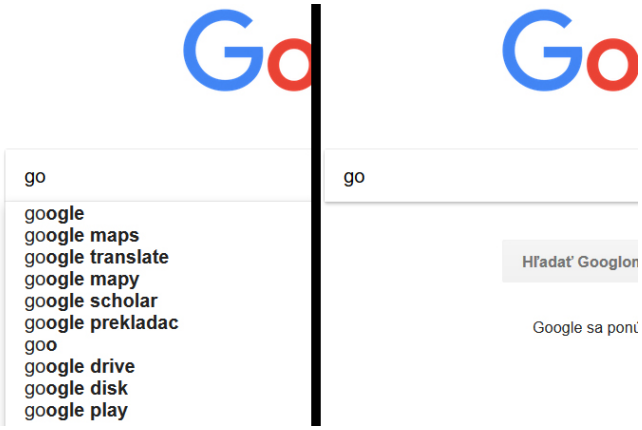
- **Testovanie na ukážkovom webe**[3]**:** k rozšíreniu bola vytvorená testovacia stránka, ktorá slúži na demonštráciu funkčnosti implementácie. Web demonštruje prácu so zapuzdrenými objektmi, na ktoré sa práca [36] zamerala. Ide teda o `window.Date()`, `window.performance.now()`, `getCurrentPosition()`, `window.XMLHttpRequest` a samozrejme aj HTML elementy typu Canvas. Pri testovaní boli postupne vypínané a zapínané dané nastavenia pre funkcie a objekty prostredníctvom stránky s nastavením. Taktiež sa menili nastavenia pre testovanie každého nastavenia. Pri testovaní sa ukázalo, že každý prvok rozšírenia funguje správne. Teda napríklad pri zaokrúhľovaní hodnôt `window.Date()` sa naozaj zobrazovaný čas na webovej stránke zaokrúhľoval podľa toho, ako boli nastavené dané nastavenia. Teda na jednotky sekúnd alebo stovky milisekúnd a podobne (obrázok 4.3). Ukážkový web teda dokázal funkčnosť rozšírenia v prehliadači Mozilla Firefox.



Obr. 4.3: Hore je objekt `window.Date` pri vypnutých nastaveniach rozšírenia. V strede so sníženou presnosťou a teda zaokrúhľovaním na celé jednotky sekúnd. Dole je zaokrúhľovanie na stovky milisekúnd. Obrázok je kompiláciou troch rôznych výstrižkov obrazovky pri mojom testovaní rozšírenia JavaScript Restricter.

- **Testovanie na reálnych weboch:** pre testovanie na reálnych weboch boli vybrané niektoré stránky, na ktorých bola preukázaná funkčnosť rozšírenia. Napríklad pre web *seznam.cz* bolo ukázané, že pred vykresľovaním reklám cez HTML elementy typu Canvas, bolo pri nastavení vyžadovať povolenie, vyžadované každé povolenie od používateľa pre vykresľovanie daných elementov. Rovnako po nastavení úplného blokovania žiadostí `window.XMLHttpRequest` pri vyhľadávaní pomocou vyhľadávača Google nebola ponúkaná žiadna nápoveda pri zadávaní hľadanej položky (obrázok 4.4). Týmto spôsobom bola ukázaná funkčnosť rozšírenia na celej rade webových stránok, ktoré bežný užívateľ môže denne navštevovať.

[3] Ukážková webová stránka dostupná na https://github.com/cervinka-zbynek/masters-thesis/tree/master/demo_example



Obr. 4.4: Situácia pred zapnutím blokovania žiadostí `XMLHttpRequest` (vľavo) a po zapnutí blokovania žiadostí (vpravo). Kompilácia výstrižkov obrazovky pri mojom testovaní rozšírenia JavaScript Restricter.

- **Testovanie v konzole prehliadača:** pri testovaní cez konzolu prehliadača sa potvrdila funkčnosť rozšírenia. Opakovane boli spúšťané dané úseky JavaScriptového kódu v prehliadači, ktoré vypisovali do konzole prehliadača výsledky. Tieto výsledky potvrdili správnu funkčnosť všetkých implementovaných súčastí rozšírenia JavaScript Restricter pri danom nastavení rozšírenia.

```
var performanceNowValue = window.performance.now();
console.log(performanceNowValue);
```

Kód 4.2: Príklad testovaného JavaScriptového kódu. Do konzole vypisuje aktuálne hodnoty funkcie.

Ing. Zbyněk Červinka tak vytvoril v svojej diplomovej práci prototyp rozšírenia, ktoré pomáha chrániť užívateľa pred škodlivými webovými stránkami a zvyšuje ochranu súkromia používateľov pri prehliadaní webu. Konkrétnejšie zhodnotenie rozšírenia JavaScript Restricter a analyzovanie jeho silných stránok, ako aj nedostatkov sa nachádza v kapitole 6 mojej práce.

Kapitola 5

Ďalšie existujúce rozšírenia pre ochranu súkromia

V tejto kapitole sa pozrieme na niektoré existujúce rozšírenia, ktoré prispievajú k ochrane súkromia užívateľa pri prezeraní webu. Tieto rozšírenia sú dostupné v oficiálnych obchodoch pre rozšírenia prehliadačov. Zhodnotíme prínos pre používateľa, ako aj nevýhody spojené s daným rozšírením.

5.1 Privacy Badger

Rozšírenie Privacy Badger dostupné pre Google Chrome, Mozillu Firefox, Operu. Rozšírenie automaticky deteguje a blokuje externe načítané prvky stránky, ktoré potencionálne môžu odosielať informácie o užívateľovi. Na základe heuristiky, ktorá sa stále vylepšuje, Privacy Badger rozhoduje, ktoré prvky budú blokované, a teda ide o prvky, ktoré narúšajú súkromie užívateľa. Ďalej rozšírenie rozhodne, ktoré prvky sú pravdepodobne bezpečné a budú povolené. Užívateľ si pomocou užívateľského rozhrania vie detailne pozrieť a skontrolovať, ktoré prvky boli a neboli blokované, poprípade vie zmeniť ich stav a ručne zapnúť alebo vypnúť ich blokovanie. Rovnako je možné nastaviť nie úplné blokovanie ale iba blokovať cookies sledovacieho prvku [4].

Pri testovaní tohto rozšírenia sa ukázalo, že sa heuristika daného rozšírenia niekedy mýli a zle rozhoduje, ktoré prvky blokovať. Narazil som na situácie, kedy boli povolené niektoré prvky, ktoré po preskúmaní odkazovali na analytické nástroje pre sledovanie a zachytávanie krokov užívateľa na danej webovej stránke. Napríklad išlo o nástroje ako Hotjar[1] alebo Clicky[2], ktoré zhromažďovali informácie o klikoch na dané miesta webovej stránky.

Naopak, ak Privacy Badger blokoval dôležitý prvok pre chod webovej stránky a web by nepracoval správne, môže nastať situácia, kedy bude nutné manuálne vypnúť rozšírenie na danej stránke alebo nájsť daný prvok a povoliť ho, čo sťažuje používanie rozšírenia pre užívateľa.

5.2 Adblock Plus

Adblock Plus je dostupný na prehliadače Google Chrome, Microsoft Edge, Opera, Safari, Yandex Browser a funguje aj priamo integrovaný v prehliadači Adblock Browser pre zaria-

[1] Nástroj pre analýzu https://hotjar.com
[2] Nástroj pre analýzu https://clicky.com/

denia s operačným systémom Android. Toto rozšírenie deteguje a blokuje reklamy na webových stránkach. Okrem toho vie ešte blokovať sledovacie prvky na prehliadaných stránkach. Adblock Plus používa zoznamy pravidiel na odfiltrovanie nežiadúcej reklamy [6]. Taktiež podľa pravidiel, podľa ktorých sa riadi rozšírenie vie blokovať nežiadúce sledovacie prvky. Pre blokovanie a filtrovanie nežiadúcich prvkov využíva regulárne výrazy, na základe ktorých sa dané prvky blokujú. Adblock môže teda blokovať JavaScript, no len na úrovni celého súboru, čo môže viesť k nefunkčnosti častí webových stránok.

Pre ochranu súkromia je možné zapnúť blokovanie sledovania ikon sociálnych médií, teda zamedzenie ikon sociálnych médií na prehliadaných stránkach. Pomocou týchto ikon vedia sociálne siete sledovať zvyky užívateľa danej sociálnej siete. Ďalej je možné si nastaviť blokovanie sledovacích prvkov na weboch. Teda niečo ako Privacy Badger s rozdielom, že nie je možné zobraziť, ktoré sledovacie prvky sú blokované. Tým pádom blokovanie týchto prvkov ostáva len na rozšírení a v prípade, že sa chybne blokujú aj prvky pre správny beh webu, neostáva užívateľovi nič iné ako úplne vypnúť rozšírenie na danom webe. Tým sa ale vypne ochrana užívateľa a začnú sa zobrazovať nechcené reklamy.

5.3 Quick Javascript Switcher

Rozšírenie Quick Javascript Switcher slúži pre rýchle vypnutie JavaScriptu na webovej stránke. Jedným klikom na ikonku rozšírenia v paneli nástrojov prehliadača sa vypne alebo zapne vypnutý JavaScript na danej stránke. Jednoduchosť rozšírenia si získala desaťtisíce užívateľov prehliadača Google Chrome, na ktorý je toto rozšírenie dostupné. Všetky sledovacie prvky na základe JavaScriptu sa tak dajú jednoducho blokovať.

Nevýhodou ale je, že ide o úplne blokovanie JavaScriptu. Keďže dnes je správny chod webových stránok často krát založený práve na JavaScripte, dochádza často k znefunkčneniu webov. Pri testovaní sa to hneď prejavilo a napríklad nebolo možné načítať modul komentárov pod článkom na informačnom webe alebo nebolo možné vybrať položku z rozbaľovacieho menu na webe.

Existuje mnoho veľmi podobných rozšírení ako Quick Javascript Switcher. Napríklad aj NoScript pre Mozillu Firefox, JavaScript Switcher pre Operu alebo Disable JavaScript pre Google Chrome.

5.4 Zhodnotenie existujúcich rozšírení

Z jednotlivých testovaní vyplýva, že rozšírenia, ktoré automaticky rozhodujú o tom, ktoré prvky sa považujú za nechcené, nie vždy rozhodujú správne a blokované prvky môžu znefunkčniť stránku. Časom sa síce očakáva zlepšenie analýzy nechcených sledovacích prvkov, no v tejto chvíli môže byť užívateľ miestami nútený ručne vypínať alebo zapínať ochranné prvky rozšírení. Tým sa narušuje plynulosť prehliadania webu používateľmi rozšírení.

Úplným zablokovaním JavaScriptu sa síce zamedzí funkcia sledovacích prvkov cez JavaScript, ale rovnako sa aj často znefunkčnia webové stránky a niekedy je to až neprijateľné z použiteľného hľadiska pri prezeraní webu.

Keďže podobných rozšírení je mnoho, tieto boli vybrané ako príklady k danej problematike. Popis niektorých ďalších je možné napríklad nájsť aj v tretej kapitole práce [36].

Kapitola 6

Návrh implementácie rozšírenia JavaScript Restrictor

Celá táto kapitola sa venuje návrhu implementácie rozšírenia pomenovaného JavaScript Restrictor alebo skrátene aj JSR, ktoré bolo vytvorené v rámci tejto diplomovej práce. Keďže moje implementované rozšírenie vychádza z riešenia JavaScript Restricter, najprv sa pozrieme na analýzu rozšírenia JavaScript Restricter v sekcii 6.1. Je kľúčové zhodnotiť klady a zápory prototypu, aby bolo možné vylepšiť funkcionalitu rozšírenia. Ďalej si popíšeme zmeny JSR oproti prototypu.

Na prvý pohľad by nás mohlo napadnúť, že by sa dali spojiť rozšírenia JSR a Chrome Zero a vytvoriť akúsi fúziu týchto dvoch rozšírení. Rozšírenia sa ale nebudú spájať z dôvodu nie celkom jasnej funkčnosti Chrome Zero, ako to bolo spomenuté v sekcii 3.4.1. Ďalším dôvodom je závislosť rozšírenia Chrome Zero na prehliadači Google Chrome, kedy by bolo pri fúzii rozšírení potrebné značne prepracovávať niektoré časti zdrojového kódu Chrome Zero. Vývoj JSR tak pôjde nateraz svojou cestou, kde sa oproti prototypu rozšíri počet obaľovaných konštrukcií jazyka a zavedú sa úrovne ochrany užívateľa vysvetlené v sekcii 6.2. K tomu sa vytvorí patričné užívateľské rozhranie, ktorého popis nájdeme v sekcii 6.3.

6.1 Výhody a nevýhody rozšírenia JavaScript Restricter

Rozšírenie JavaScript Restricter, ktoré vytvoril Ing. Zbyněk Červinka vo svojej diplomovej práci [36], ponúka riešenie pre zamedzenie zberu osobných dát používateľa cez JavaScript. Jeho veľkou výhodou je implementácia rozšírenia. Spôsob zapuzdrenia a obalenia pôvodných objektov a funkcií jazyka JavaScript umožňuje skryť konštrukcie a zabrániť tak možnému zneužitiu a prístupu k funkciám. JavaScript Restricter tak vie napríklad automaticky zabrániť aj útokom spojeným s presným načasovaním, popísaných v sekcii 3.1.2. Zaokrúhľovanie hodnôt času či hodnôt spojeným s detekciou lokácie, blokovanie žiadostí `XMLHttpRequest` a vykresľovania elementov typu Canvas pomáhajú znížiť prúd a presnosť osobných údajov, ktoré webové stránky o používateľoch zbierajú. Stránka s nastavením ponúka detailné nastavenia a je teda možné si rozšírenie dostatočne prispôsobiť, ak je používateľ dostatočne oboznámený s problematikou.

Na druhej strane tu nájdeme aj nevýhody. Prototyp ponúka zapuzdrenie iba piatich objektov či funkcií jazyka. Rozhodne ako nevýhodu môžeme považovať obaľovanie funkcie `window.HTMLCanvasElement.prototype.getContext()`. Úplným blokovaním tohto elementu môže často dochádzať ku čiastočnému znefunkčneniu webových stránok, ktoré

užívateľovi zobrazujú informácie pomocou elementov Canvas. Pri testovaní prototypu som viac krát narazil na weby, ktoré boli negatívne ovplyvnené. Napríklad zobrazenie máp[1], pri zobrazení grafov a rebríčkov niektoré webové stránky nevedeli zobraziť dané grafy. Namiesto nich zobrazovali prázdne oblasti[2] alebo vypísali upozornenie, že pravdepodobne prehliadač nepodporuje grafické prvky Canvas či WebGL. Pri stránkach Google často dochádzalo k presmerovaniu na stránku pomoci alebo výzve na aktualizáciu prehliadača. Týmto rozšírenie síce chráni užívateľa pred získaním popisov grafických kariet (fingerprintu), no na druhej strane znefunkčňuje weby.

Ďalej taktiež môže byť stránka s nastavením vnímaná ako negatívum. Keďže je potrebné pre každú funkciu ručne prejsť nastavenie a vybrať si jednu z možností, to je pre bežných užívateľov, ktorí nie sú detailne oboznámení s touto problematikou, dosť problémové. Chýba teda nejaké jednoduchšie nastavenie pre bežného používateľa. Nevýhodou je aj, že nie je stránka s nastavením v anglickom jazyku. Keďže je iba v češtine, rozšírenie nie je použiteľné mimo česky alebo slovensky hovoriacu komunitu. Taktiež rozšírenie je vo forme prototypu, a preto nie je dostupné na oficiálnych stránkach rozšírení a bolo vyvíjané iba pre Mozillu Firefox.

6.2 Návrh funkcionality rozšírenia

Táto sekcia sa zaoberá návrhom funkcionality rozšírenia JavaScript Restrictor. Jedným z cieľov tejto práce je, že rozšírenie bude oficiálne dostupné na viac ako jednom webovom prehliadači. Rozšírenie tak bude dostupné na oficiálnych obchodoch pre webové prehliadače Mozilla Firefox, Google Chrome a Opera. Tieto prehliadače boli zvolené, pretože Google Chrome a Mozilla Firefox sú momentálne najpoužívanejšie prehliadače [28]. Opera je založená na Chromium[3] technológii ako aj Google Chrome. Keďže bude vyvíjaná verzia pre Google Chrome, bez zmien ju bude možné využiť aj v Opere. Týmto bude možné rozšíriť publikum pre naše rozšírenie ešte o niečo viac. Oficiálnym názvom, pod ktorým je rozšírenie publikované, je JavaScript Restrictor.

Názov rozšírenia sa oproti prototypu Ing. Zbyňka Červinky [36] mení z JavaScript Restricter na JavaScript Restrictor. Je to z dôvodu, že slovo restrictor sa v technickej sfére využíva viac ako restricter, aj keď významovo obe slová vyjadrujú to isté. Koncovka *-or* sa totiž častejšie používa pri pomenovaní neživotných podstatných mien.

Keďže prototyp bol vyvíjaný na prehliadač Mozilla Firefox, najprv bude moje rozšírenie vyvíjané pre Mozillu Firefox a následne sa prevedú nutné úpravy, aby rozšírenie fungovalo aj na prehliadačoch Google Chrome a Opera.

Princíp fungovania rozšírenia bude zachovaný z prototypu. Zapuzdrené a obalené budú ako funkcie a objekty z prototypu, tak aj nové objekty či vlastnosti objektov, ktoré považujem za vhodné pre zvýšenie anonymity užívateľov. Kritérium, kedy považujem dané API za vhodným je, že API poskytuje informácie alebo nepriamo vie poskytnúť informácie o prehliadači užívateľa alebo jeho hardvéri. Úlohou je teda zamedziť zber informácii alebo upraviť dané informácie, aby tvorili čo najväčšiu anonymitnú množinu [25], teda aby bola anonymita užívateľa silnejšia. Zároveň rozšírenie by nemalo narušiť funkcionalitu webových stránok, aby si nemusel užívateľ voliť medzi ochranou a správnou funkcionalitou stránky.

[1] https://www.google.com/maps
[2] Napríklad https://theskylive.com/jupiter-info alebo https://mega.nz/
[3] https://www.chromium.org/Home

6.2.1 Obaľované API

JavaScript Restrictor ponechá implementáciu troch z pôvodných piatich obaľovaných konštrukcií z prototypu bez zmeny. Konkrétne ide o tieto API:

- **Funkcia `window.performance.now()`**,
- **funkcia `navigator.geolocation.getCurrentPosition()`**,
- **objekt `window.XMLHttpRequest`**.

Ďalej implementácia týchto konštrukcií bude upravená alebo vymenená z dôvodu, že sa pri testovaní prototypu rozšírenia prišlo na ich nedostatky:

- **Objekt `window.Date`:** Implementácia objektu bude musieť byť oproti prototypu upravená, presnejšie povedané mierne rozšírená. Dôvodom je, že prototyp neobaľoval celý objekt aj s jeho vlastnosťami. Modifikovaný objekt tak neobsahoval niektoré vlastnosti, ktorých volanie na strane klienta mohlo viesť k znefunkčneniu webových stránok, pretože dané vlastnosti objektu neboli dostupné. Sprístupnené budú teda aj všetky vlastnosti objektu `window.Date`.
- **Funkcia `window.HTMLCanvasElement.prototype.toDataURL()` bude náhradou za funkciu `window.HTMLCanvasElement.prototype.getContext()`:** JSR naďalej nebude obaľovať Canvas funkciu `getContext()`. Dôvodom je, ako bolo spomínané v sekcii 6.1, zapuzdrenie tejto funkcie často viedlo k čiastočnému znefunkčeniu prehliadanej webovej stránky. Namiesto tejto funkcie bude obaľovaná funkcia `toDataURL()`. Táto funkcia Canvas API sa podľa článku [1] najčastejšie podieľa na získavaní fingerprintu zariadenia užívateľa. Výsledkom obalenia tejto funkcie bude, že stránka, ktorá zavolá túto funkciu dostane upravený obrázok namiesto pôvodného. Teda funkcia vráti rovnako veľký obrázok, ale bude mať všetky pixely bielej farby. Každý pixel bude mať hodnoty *RGB(255,255,255)*, čím nebude možné získať presný fingerprint zariadenia. Zároveň zápis do Canvas elementov nebude ovplyvnený. Týmto nebude dochádzať k znefunkčneniu stránok využívajúcich Canvas elementy, ako tomu bolo v prototype rozšírenia.

Všetky nasledujúce API boli vybrané a pridané do rozšírenia. Tieto API tak bude rozšírenie zapuzdrovať a podľa potreby modifikovať, aby sa zvýšila anonymita užívateľa. Rovnako dochádza k modifikovanou niekoľkých položiek v HTTP hlavičkách, ktoré webové stránky využívajú. Aj tieto hlavičky budú v prípade potreby modifikované.

- **Vlastnosť `window.navigator.userAgent`:** Táto vlastnosť, ako všetky vlastnosti objektu `window.navigator` slúži iba na čítanie (anglicky *read-only* vlastnosť) a po zavolaní vracia textový reťazec obsahujúci meno používaného prehliadača, ako aj aktuálnu verziu prehliadača. Ďalej obsahuje informácie o operačnom systéme užívateľa v podobe druhu operačného systému (napríklad Linux, Windows) a bitovej verzii, teda či sa jedná o 32 alebo 64-bitovú verziu operačného systému. Príkladom je reťazec *Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729 Safari/537.36*, ktorý označuje prehliadač Google Chrome bežiaci na 64-bitovom Windows 10. Hodnota *Mozilla/5.0* sa uvádza pri každom prehliadači, a to z historických dôvodov. Modifikovaním tejto hodnoty na najpoužívanejší a najpopulárnejší prehliadač, verziu prehliadača, ako aj operačný systém, tak vieme dosiahnuť vyššiu anonymitu užívateľa v prípade, že používa inú, napríklad staršiu verziu prehliadača.

- **HTTP hlavička `User-Agent`:** Pri úprave `window.navigator.userAgent` sa musí upravovať aj HTTP hlavička `User-Agent`, pretože niektoré weby nezisťujú používaný prehliadač a operačný systém cez JavaScript, ale pomocou HTTP WebRequest volaní. Teda hlavička sa musí nastaviť na ekvivalentný textový reťazec ako v prípade JavaScriptu.

- **Vlastnosť `window.navigator.vendor`:** Vlastnosť, ktorá úzko súvisí s predchádzajúcou vlastnosťou `window.navigator.userAgent`. Jedná sa o informáciu o výrobcovi prehliadača. V prípade Firefoxu sa jedná o prázdny reťazec, v prípade Chrome alebo Opery ide o označenie *Google Inc.* Ak chceme podvrhnúť používaný prehliadač, je potrebné podvrhnúť aj túto hodnotu.

- **Vlastnosť `window.navigator.platform`:** Ide o textový reťazec, ktorý vracia jednoznačné označenie platformy, na ktorej beží daný prehliadač. Každá platforma ma svoje hodnoty. Napríklad všetky operačné systémy Windows od Windows 95 a novšie vracajú reťazec *Win32*. Táto informácia sa podobne ako pri `navigator.UserAgent` dá využiť pre identifikovanie platformy užívateľa.

- **Vlastnosť `window.navigator.appVersion`:** Jedná sa o reťazec, ktorý udáva aktuálnu hodnotu verzie prehliadača. Rôzne prehliadače vracajú rôzne hodnoty, no štandardne sa jedná o hodnotu *5.0* z historických dôvodov, nasledovanou platformou, na ktorej beží prehliadač.

- **Vlastnosť `window.navigator.oscpu`:** Po zavolaní tejto vlastnosti dostávame názov operačného systému, na ktorom beží prehliadač. Avšak niektoré prehliadače ako napríklad Google Chrome nemajú definovanú túto vlastnosť a teda nie je vhodné sa na túto vlastnosť z pohľadu vývojára spoliehať.

- **Vlastnosť `window.document.referrer`:** Táto vlastnosť vracia informáciu o predchádzajúcej navštívenej stránke užívateľa. Konkrétnejšie ide o URL stránky, na ktorej bol odkaz, ktorý priviedol užívateľa na aktuálnu stránku. Táto vlastnosť prezrádza odkiaľ daný užívateľ na stránku prišiel. Samozrejme je to často samotný navštívený web, na ktorom sa užívateľ nachádza a kliká na rôzne hyperlinky a presúva sa na podstránky danej stránky. Môžu to byť ale aj sociálne siete alebo reklamné bannery na iných weboch, ktoré priviedli užívateľa na aktuálnu stránku. Upravenou hodnotou tejto vlastnosti na prázdny textový reťazec vytvárame obraz, že užívateľ vždy pristupoval k danej webovej stránke priamo a nikdy cez odkazy na iných stránkach. Týmto spôsobom sa nedá o užívateľovi povedať, z akej URL prišiel a zvyšuje sa tým jeho anonymita.

- **HTTP hlavička `Referer`:** URL stránky, ktorá viedla na aktuálnu stránku môžeme získať aj cez HTTP hlavičku, kde je dané URL uvedené pri položke `Referer`. Niektoré weby tak môžu získavať túto hodnotu cez HTTP hlavičku, niektoré cez vlastnosť `window.document.referrer`. Preto je potrebné obe cesty modifikovať v prípade, keď budeme chcieť tieto hodnoty upraviť.

- **Vlastnosť `window.navigator.language` a `window.navigator.languages`:** Tieto dve vlastnosti vracajú preferované jazyky, ktoré ma užívateľ v prehliadači prednastavené na zobrazovanie webov. V prípade, že užívateľ používa verziu prehliadača kde má ako preferovaný jazyk nastavenú napríklad slovenčinu či češtinu, vlastnosť `window.navigator.language` vracia ako predvolený jazyk reťazec *sk-SK* respektíve

cs-CZ. `window.navigator.languages` vracia celé pole hodnôt, ktoré sú preferované od najviac po najmenej preferovaný jazyk. V prípade, že daný web nepodporuje najviac preferovaný jazyk, pokúsi sa vyhovieť menej a menej preferovanému jazyku až po základný jazyk, ktorý je často angličtina. Podvrhnutím týchto hodnôt na najviac používané hodnoty v rámci sveta, vieme zabrániť tomu, že užívateľ bude ľahšie identifikovateľný. Pretože ak užívateľ používa napríklad slovenčinu ako preferovaný jazyk, táto množina užívateľov je podstatne menšia ako užívatelia s preferovaným napríklad anglickým jazykom.

- **HTTP hlavička `Accept-Language`:** Keďže druhou možnosťou ako zistiť preferovaný jazyk a všetky podporované jazyky je pomocou HTTP požiadavky, je potrebné upraviť aj túto položku hlavičky. Rozdielom oproti JavaScriptovým API je to, že `Accept-Language` môže udávať váhu daného jazyka priamo pri jazyku. Teda niečo ako poradie, v ktorom sa preferujú dané jazyky. Tieto jazykové hodnoty, ako aj vyššie spomenuté hodnoty získané cez API JavaScriptu, sa využívajú pre viacjazyčné weby, aby vedeli užívateľovi zobraziť obsah v jeho preferovanom jazyku.

- **Vlastnosť `window.navigator.doNotTrack`:** Nastavením tejto vlastnosti na hodnotu *1* alebo *yes* sa dáva najavo webovým stránkam, že užívateľ si nežiada, aby bol sledovaný. Ide o vlastnosť, ktorá je stále vo vývoji a ktorú je odporúčané sledovať pri vývoji webov a webových aplikácii, no nie je zaručené, že vývojári sa týmto aj naozaj riadia. Nastavením tejto hodnoty, aby webové stránky nesledovali užívateľa, tak môžeme pomôcť pri stránkach, ktoré kontrolu pre `window.navigator.doNotTrack` naozaj impelentovali. Týmto spôsobom tak zvýšiť anonymitu užívateľa.

- **Vlastnosť `window.navigator.deviceMemory`:** Cez JavaScript je možné zistiť aj veľkosť pamäte RAM daného zariadenia. Čísla, hodnoty, ktoré vybočujú z najviac používaných hodnôt tak vedia prezradiť informáciu naviac o užívateľovi so špecifickejším hardvérom. Nastavením tejto hodnoty na najčastejšie vyskytujúcu sa hodnotu vieme týmto užívateľom zmenšiť mieru identifikovateľnosti.

- **Vlastnosť `window.navigator.hardwareConcurrency`:** Druhou hardvérovou vlastnosťou, ktorú objekt `navigator` poskytuje, je hodnota udávajúca počet logických jadier procesora daného zariadenia dostupných pre využitie aplikácie alebo webu na svoju prácu. Nastavením tejto hodnoty na najčastejšie sa vyskytujúcu hodnotu môžeme zvýšiť anonymitu užívateľov, ktorých procesor je unikátnejší ako pri väčšine užívateľov.

- **Vlastnosť `window.navigator.cookieEnabled`:** Táto vlastnosť vracia boolean hodnotu *true* alebo *false*, ktorou udáva nastavenie, či sú cookies povolené alebo nie. Ak teda prehliadač nepovoľuje cookies, tak toto API vracia hodnotu *false*. Je otázne do akej miery vývojári webových stránok a aplikácií naozaj kontrolujú túto položku, no nastavením hodnoty na *false* je teoreticky možné dopomôcť k zvýšeniu ochrany a anonymity užívateľa. Dôvodom je, ako bolo spomenuté v sekcii 4.1.1, cookies sa dajú použiť pre identifikáciu užívateľa na internete. Na druhej strane užívateľ chce cookies často využívať pre väčšie pohodlie prehliadania webu.

6.2.2 Funkcionalita

Rozdielom oproti prototypu mimo pridania nových obaľovaných API je, že rozšírenie nebude ponúkať iba manuálne nastavenie úrovne ochrany používateľa pred útokmi či zberom osobných dát. Používateľ bude mať na výber jednu z troch prednastavených úrovní. Ďalej si bude môcť zvoliť aj vlastné nastavenie, ktoré si bude vedieť manuálne nastaviť, presne tak ako v prototype rozšírenia. Nebude chýbať možnosť vypnúť funkcionalitu rozšírenia. Spolu teda bude päť úrovní ochrany užívateľa. Vypnutá funkcionalita alebo, inak povedané, úroveň 0, potom úrovne 1 až 3 a vlastná úroveň, teda manuálne nastavená úroveň užívateľom. Úroveň 1 predstavuje minimálnu ochranu a úroveň 3 maximálnu ochranu. Predvolená úroveň ochrany pri nainštalovaní rozšírenia bude úroveň 2, čo predstavuje akýsi kompromis prvej a tretej úrovne. Konkrétnejší popis jednotlivých úrovní sa nachádza v sekcii 6.2.3.

Keďže používateľ by mohol chcieť rozličné nastavenia ochrany pre rozličné webové stránky, bude možné si nastaviť explicitne úroveň ochrany pre každú doménu osobitne. Rovnako bude možné si nastaviť rozličnú úroveň pre subdomény danej domény. Napríklad na *fit.vutbr.cz* bude nastavená úroveň 3 ale na *vutbr.cz* to bude úroveň 1. Ak nebude nastavená úroveň pre konkrétnu webovú stránku, bude použitá globálna úroveň ochrany, ktorú si používateľ zvolil.

Tieto nastavenia pre jednotlivé domény budú uložené v databáze. Do databázy bude možné vložiť konkrétnu doménu s úrovňou nastavenia. Samozrejme bude možné meniť úrovne pre domény, ako aj vymazať doménu z databázy.

6.2.3 Popis úrovní ochrany

Konkrétne hodnoty či miera zaokrúhľovania obalených API pri jednotlivých úrovniach boli diskutované s vedúcim práce.

Úroveň 0 (anglicky *Level 0*) predstavuje vypnutú funkcionalitu rozšírenia. Je to stav ako keby rozšírenie nebolo ani v prehliadači nainštalované. Nemá žiadny vplyv na webovú stránku.

Úroveň 1 (*Level 1***)** predstavuje minimálnu úroveň ochrany. Správanie API je nasledujúce:

- **Hodnota `window.navigator.doNotTrack`:** *yes*.
- **Zaokrúhľovanie hodnôt `window.Date`:** *na stotiny sekundy*.
- **Zaokrúhľovanie hodnôt `window.performance.now()`:** *na desiatky* (ekvivalentné zaokrúhľovaniu na stotiny sekundy objektu `window.Date`).
- **Zaokrúhľovanie hodnôt `navigator.geolocation.getCurrentPosition()`:** zemepisná šírka a výška *na stotiny*, nadmorská výška a jednotlivé presnosti merania či rýchlosť a orientácia zariadenia *na desiatky*.
- **Hodnota `navigator.deviceMemory`:** *4* - najčastejšia hodnota v tejto chvíli [20].
- **Hodnota `navigator.hardwareConcurrency`:** *2* - najčastejšia hodnota v tejto chvíli [20].

Úroveň 2 (*Level 2*) ponecháva nastavenia z prvej úrovne. Naviac pridáva aj ochranu proti Canvas fingerptintingu a podvrhujú sa aj informácie o prehliadači a operačnom systéme. Zaokrúhľovanie hodnôt je ešte väčšie (uvádzané sú iba rozdiely oproti úrovni 1):

- **Zaokrúhľovanie hodnôt `window.Date`:** *na desatiny sekundy*.

- **Zaokrúhľovanie hodnôt `window.performance.now()`:** *na stovky* (ekvivalentné zaokrúhľovaniu na desatiny sekundy objektu `window.Date`).
- **Zaokrúhľovanie hodnôt `navigator.geolocation.getCurrentPosition()`:** zemepisná šírka a výška *na desatiny*, nadmorská výška a jednotlivé presnosti merania či rýchlosť a orientácia zariadenia *na stovky*.
- **`window.HTMLCanvasElement.prototype.toDataURL()`:** *zapnutá modifikácia* - funkcia vracia úplne biely obrázok.
- **`window.navigator.userAgent`:** na základe reálneho prehliadača užívateľa, ktorý používa, sa v prípade Firefox zvolí *Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:67.0) Gecko/20100101 Firefox/67.0*. Firefox verzia 67 je najnovšia verzia, ktorá sa postupne aktualizáciami rozšíri medzi užívateľov. V prípade Google Chrome a Opery to bude hodnota *Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729 Safari/537.36*, čo predstavuje najnovšiu verziu prehliadača Chrome. Keďže Windows 10 je najpoužívanejší operačný systém sveta [29], preto sa podvrhuje operačný systém práve na Windows 10. Od tohto sa odvíjajú ďalšie nastavenia API.
- **Hodnota `window.navigator.vendor`:** na základe reálneho prehliadača užívateľa sa v prípade Firefox zvolí prázdny reťazec a v prípade Chrome, Opery to bude reťazec *Google Inc.*
- **Hodnota `window.navigator.platform`:** *Win32* - odvodené od Windows 10.
- **Hodnota `window.navigator.appVersion`:** *5.0 (Windows)* - odvodené od Windows 10.
- **Hodnota `window.navigator.oscpu`:** *undefined* - dôvodom je, že väčšina moderných prehliadačov toto API nepodporuje vôbec. Naviac sa využitie tohto API už v najnovších prehliadačoch neodporúča.
- **Hodnoty `window.document.referrer` a HTTP WebReguest `Referer`:** *prázdny reťazec* - aby nebolo možné zistiť, z akého odkazu užívateľ prišiel na danú webovú stránku.

Úroveň 3 (*Level 3*) naviac podvrhuje ako jediný jazyk prehliadača angličtinu. Mení niektoré hodnoty alebo ponecháva nastavenia z druhej úrovne (uvádzané sú iba rozdiely oproti úrovni 2):

- **Zaokrúhľovanie hodnôt `window.Date`:** *na celé sekundy.*
- **Zaokrúhľovanie hodnôt `window.performance.now()`:** *na tisíce* (ekvivalentné zaokrúhľovaniu na celé sekundy objektu `window.Date`).
- **Nulovanie hodnôt `navigator.geolocation.getCurrentPosition()`:** všetky hodnoty sú nulované. Každá hodnota vracia číslo *0*.
- **`window.navigator.userAgent`:** keďže je v tejto chvíli najpoužívanejší prehliadač Google Chrome [28] a operačný systém Windows 10 [29], hodnota sa bude nastavovať na *Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729 Safari/537.36*. Bez rozdielu reálneho prehliadača, ktorý

používateľ využíva, hodnota bude udávať Chrome, aby maximálna úroveň ochrany odrážala aktuálne najpoužívanejší prehliadač a operačný systém.

- **Hodnota `window.navigator.vendor`:** *Google Inc.*, odvodené od Chrome.
- **Hodnoty `window.navigator.language`, `window.navigator.languages` a HTTP WebReguest hlavička `Accept-Language`:** hodnota *en-US*, pole hodnôt *en-US, en* respektíve pre HTTP WebRequest hodnota hlavičky *en-US,en;q=0.5* - nastavenie angličtiny ako jediný jazyk prehliadača.

Vlastná úroveň (*Custom level*) je úroveň, ktorú si bude môcť definovať užívateľ. Užívateľ si môže vybrať jednotlivé API, na ktoré chce aplikovať zapuzdrenie a ich hodnoty či mieru zaokrúhľovania. Naviac táto úroveň povoľuje aj možnosť využitia nastavenia pre XHR žiadostí a `window.navigator.cookieEnabled`.

Objekt `window.XMLHttpRequest` ako aj `window.navigator.cookieEnabled` neboli zaradené medzi úrovne 1 až 3, pretože ich použitie je skôr pre experimentálnu činnosť, ako reálne využitie. Pri práci s XHR žiadosťami je potrebné byť opatrný, pretože ich úplné blokovanie môže značne znefunkčniť webové stránky a pýtanie sa užívateľa pri každej XHR žiadosti či ju povoľuje alebo blokuje môže nadmerne zaťažovať užívateľa. V tomto prípade sa skôr vyžaduje vytvorenie nejakej heuristiky na zisťovanie, ktoré žiadosti blokovať, a ktoré nie. Vytvorenie takejto heuristiky je ale nad rámec tejto diplomovej práce. Ďalej `window.navigator.cookieEnabled` je zaradené tiež ako experimentálne API, pretože cookies sú pre správny beh stránok často veľmi potrebné a webové stránky často ukladajú cookies priamo bez kontroly `window.navigator.cookieEnabled`. Nepovolenie cookies týmto spôsobom preto často nemá reálny vplyv na užívateľa.

6.2.4 Technológie

Pre vytvorenie JSR rozšírenia bude rovnako ako pri prototype použitá technológia WebExtensions (popísané v sekcii 2.3), ktorá využíva HTML, CSS, JavaScript a rôzne iné potrebné súbory pre rozšírenie, napríklad obrázky.

Základná štruktúra súborov rozšírenia ako súbory `manifest.json`, `options.html`, ale aj súbory `document_start.js`, `background.js` budú prevzaté z prototypu a následne sa budú dané súbory upravovať.

Nastavenia pre konkrétne domény budú ukladané pomocou WebExtensions úložiska. Pôjde o dvojice doména a úroveň ochrany. Nastavenia vlastnej úrovne ochrany vo formáte JSON sa budú rovnako ukladať na WebExtensions úložisko a v prípade potreby sa načítajú dané nastavenia.

6.3 Návrh užívateľského rozhrania

Keďže rozšírenie má byť verejne dostupné, je veľmi dôležité, aby bolo užívateľské rozhranie jednoduché, intuitívne a pochopiteľné pre užívateľa, ktorý sa rozhodne používať alebo aspoň vyskúšať rozšírenie. Všetky texty rozšírenia budú v anglickom jazyku, keďže chceme, aby rozšírenie mohlo osloviť viac ako len česky a slovensky hovoriacich používateľov. V užívateľskom rozhraní pri webových rozšíreniach sa väčšinou môžeme stretnúť s dvomi časťami. Po stlačení tlačidla panela nástrojov sa zobrazí vyskakovacie okno, kde sú často uvedené základné informácie alebo je takto možné zmeniť nastavenie rozšírenia. Druhou časťou užívateľského rozhrania je stránka s nastavením, kde sa nachádzajú zväčša rozšírené nastavenia a možnosti, ktoré pri vyskakovacom okne neboli uvedené.

V paneli nástrojov prehliadača sa bude nachádzať ikonka rozšírenia JSR. Na ikonke bude textový údaj ohľadom práve používanej úrovne ochrany na danej webovej stránke. Text bude vo forme WebExtensions odznaku (anglicky *badge*). Na obrázku 6.1 je zobrazený návrh tejto ikonky spolu s odznakom ukazujúci úroveň ochrany.

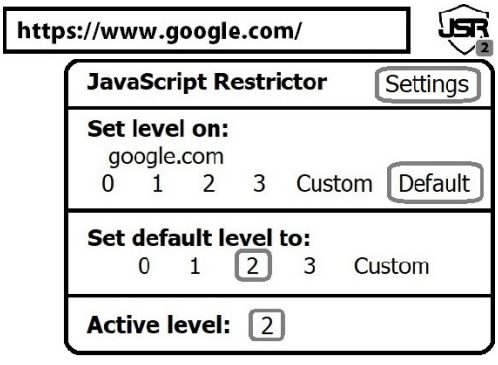
Obr. 6.1: Návrh loga a ikonky rozšírenia JSR s odznakom, ktorý práve ukazuje úroveň ochrany dva.

6.3.1 Vyskakovacie okno

Vyskakovacie okno rozšírenia (návrh na obrázku 6.2) sa zobrazí po kliknutí na ikonku v paneli nástrojov prehliadača. Okno bude obsahovať tri časti. Prvá časť bude venovaná aktuálnej doméne. Teda zobrazí sa možnosť pre nastavenie úrovne ochrany pre konkrétnu doménu. Možnosti úrovní 0, 1, 2, 3, vlastná úroveň alebo nastaviť úroveň ochrany na predvolenú úroveň (anglicky *default*). Teda po kliknutí myšou na danú úroveň sa táto úroveň nastaví pre danú doménu a zobrazí sa tlačidlo pre obnovenie stránky, aby sa vybraná úroveň aplikovala. Ak sa používa úroveň pre danú doménu, bude zvýraznené tlačidlo danej úrovne.

Cez druhú časť okna bude možné kliknutím myšou nastaviť predvolenú, globálnu úroveň ochrany. Zvýraznené tlačidlo bude slúžiť ako indikátor aktuálnej predvolenej úrovne ochrany. V prípade, že užívateľ nastaví špecifickú úroveň pre doménu pomocou prvej časti okna, tak druhá časť okna vybledne, aby to indikovalo, že sa neaplikuje predvolená úroveň ale špecifická úroveň pre aktuálnu doménu.

Nakoniec tretia časť vyskakovacieho okna bude zobrazovať, aká je aktuálne aktívna úroveň ochrany. Pretože, ak si bude užívateľ meniť nastavenie, môže sa stať, že si neuvedomí, akú má vlastne aktívnu úroveň. Rovnako pri subdoméne môže táto časť okna indikovať, že aktuálna úroveň bola zvolená na základe koreňovej domény. Napríklad ak užívateľ nastavil špecifickú úroveň pre doménu *vutbr.cz* na úroveň 1, tak sa pri *fit.vutbr.cz* aplikuje práve úroveň 1. Užívateľ si ale stále môže nastaviť špecifickú úroveň aj pre subdoménu. Tým pádom sa nebude brať do úvahy koreňová doména ale priamo subdoména.



Obr. 6.2: Návrh vyskakovacieho okna rozšírenia. Horná polovica obsahuje názov rozšírenia s tlačidlom pre prechod do nastavení a explicitné nastavenie úrovne pre doménu. Dolná polovica obsahuje nastavenie predvolenej úrovne ochrany a zobrazenie aktívnej úrovne.

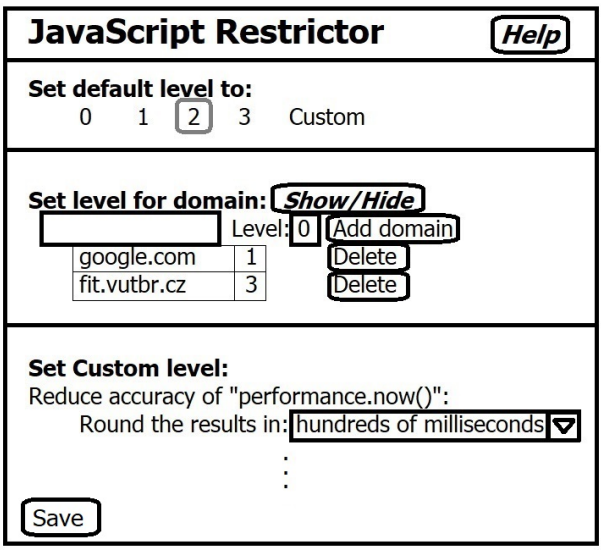
6.3.2 Stránka s nastavením

Stránka s nastavením bude rovnako ako vyskakovacie okno rozdelená na tri hlavné časti. V nastaveniach bude umiestnený odkaz na stránku pomoci a informácií o rozšírení. Na tejto stránke sa bude nachádzať popis na čo slúži rozšírenie a ako sa používa, čo znamenajú jednotlivé úrovne ochrany a odkaz na testovaciu stránku. Rovnako tam nájdeme aj odkaz pre zaslanie spätnej väzby, či nahlásenie chýb.

V prvej časti nájdeme nastavenie predvolenej úrovne ochrany. Teda bude možné si nastaviť predvolenú (globálnu) úroveň na úrovne 0, 1, 2, 3 alebo vlastne definovanú úroveň. Aktuálna úroveň bude zvýraznená rovnakým štýlom, ako pri vyskakovacom okne.

V druhej časti nájdeme nastavovanie špecifických úrovní pre domény. V tejto časti nájdeme zoznam domén a ich nastavené úrovne ochrany v podobe tabuľky. Používateľ bude mať možnosť pridať novú doménu a priradiť k nej stupeň ochrany. Rovnako bude môcť vymazať doménu zo zoznamu, čím bude následne pri jej návšteve použitá predvolená úroveň. Pri pridaní rovnakej domény, ktorá už bola v databáze prítomná sa nastavenie prepíše na novú hodnotu. Keďže zoznam môže byť v závislosti na počte pridaných domén pomerne dlhý, tento zoznam sa bude môcť skryť po stlačení tlačidla na skrytie alebo zobrazenie celej tabuľky.

Tretia časť bude slúžiť pre nastavenie vlastnej úrovne ochrany. Táto časť bude prevzatá z prototypu a budu v nej pridané nové API, ktoré bude JSR obaľovať. Teda bude možné nastavovať jednotlivým obaľovaným JavaScriptovým API mieru zaokrúhlenia vracajúcich hodnôt alebo vybrať hodnoty pre dané API. Následne používateľ potvrdí svoje preferované nastavenia tlačidlom pre uloženie nastavení. Nasledujúci obrázok 6.3 znázorňuje nákres stránky s nastavením.



Obr. 6.3: Návrh užívateľského rozhrania stránky s natavením. Zhora smerom dole sa nachádza názov rozšírenia a vedľa názvu odkaz na pomocníka. Ďalej tam nájdeme nastavenie predvolenej úrovne ochrany. Nižšie nájdeme nastavenie úrovní ochrany pre dané domény a nastavenie vlastnej úrovne ochrany.

Kapitola 7

Implementácia a publikovanie

Popis implementácie rozšírenia JavaScript Restrictor pomocou technológií WebExtensions a postup pri jeho publikovaní nájdeme v tejto kapitole. V kapitole sa v sekcii 7.1 budeme venovať implementácii a popisu jednotlivých skriptov rozšírenia, popisu vytvorenia stránky s nastavením či vyskakovacieho okna. Uvedieme aj niektoré rozdiely naprieč prehliadačmi pri tvorbe rozšírenia a na záver tejto kapitoly v sekcii 7.2 si povieme, čo všetko bolo potrebné dodržať pri publikovaní rozšírenia a aké postupy sa pri zverejňovaní musia dodržiavať.

7.1 Implementačné detaily

Táto sekcia sa zaoberá popisom implementácii jednotlivých súborov. Rozšírenie sa skladá z nasledujúcich súborov:

- **`manifest.json`**: manifest, v ktorom sú definované všetky potrebné metadáta JSR,
- **`background.js`**: skript na pozadí, ktorý beží neustále v pozadí rozšírenia,
- **`document_start.js`**: skript s prístupom k obsahu stránky, ktorý má na starosti zapuzdrenie jednotlivých API,
- **`popup.js`**: skript, ktorý sa stará o interakciu s vyskakovacím oknom (stavovým oknom rozšírenia) po kliknutí na ikonku rozšírenia,
- **`popup.html, popup.css`**: zobrazované vyskakovacie okno spoločne s jeho vizuálnou štylizáciou,
- **`options.js`**: skript, ktorý sa stará o interakciu stránky s nastavením rozšírenia,
- **`options.html, options.css`**: zobrazovaná stránka s nastavením a k nej odpovedajúca vizuálna štylizácia,
- **priečinok `img`**: priečinok so všetkými potrebnými ikonami a obrázkami, ktoré rozšírenie využíva.

Rozšírenia vytvárané pomocou WebExtensions API by mali byť kompatibilné naprieč prehliadačmi s minimálnymi zmenami. Od začiatku vývoja rozšírenia JavaScript Restrictor bolo rozšírenie vyvíjané s cieľom minimalizovať rozdiely v rámci zdrojového kódu pre odlišné prehliadače. Preto boli vždy pri implementácii pomocou zoznamu kompatibilných API [19] volené iba také API, ktoré sú dostupné ako vo Firefoxe, tak aj v Chrome a Opere.

Ďalej pre zjednotenie zdrojového kódu vrámci prehliadačov bolo potrebné zjednotiť objekty, nad ktorými sa volajú WebExtensions API. Firefox využíva objekt `browser` a Chrome s Operou objekt `chrome`. Výhodou ale je, že Firefox podporuje do istej miery aj objekt `chrome`. Rozšírenie bolo vyvíjané najprv priamo na Firefox pomocou objektu `browser`. Následne sa jednoduchým priradením `var browser = chrome` dosiahlo zjednotenie volaní API nezávisle od prehliadača pomocou objektu `browser`.

7.1.1 Manifest rozšírenia

Prvým súborom, ktorý si popíšeme je `manifest.json`. V tomto súbore sa nachádzajú všetky metadáta o rozšírení, ako aj informácie o stavbe rozšírenia alebo povolenia rozšírenia. V manifeste je definovaný názov rozšírenia, meno autora, verzia rozšírenia, skript na pozadí `background.js` a skript s prístupom k obsahu stránky `document_start.js`. Pre načítanie skriptu už pri načítaní samotnej webovej stránky je potrebné definovať kľúč `run_at: document_start`. Tento kľúč zaručuje, že JSR bude vykonávať obaľovania API už pri zostavovaní a načítaní DOM objektu stránky.

Ďalej v manifeste nájdeme definovanie hlavnej ikony rozšírenia, ktorá sa zobrazuje v paneli nástrojov prehliadača a definovanie akcie po kliknutí na túto ikonku. Ide o zobrazenie vyskakovacieho okna `popup.html`. Taktiež bola definovaná stránky s nastavením, ktorú predstavuje súbor `options.html`.

Pre správny chod rozšírenia bolo potrebné nastaviť aj oprávnenia pod kľúčom nazvaným `permissions`. Išlo o oprávnenia spojené s využitím WebExtensions úložiska využitého pri ukladaní nastavení rozšírenia a pre prístup k informáciám o kartách prehliadača potrebných pri aktualizácii ikonky rozšírenia v paneli nástrojov prehliadača. V neposlednom rade bolo potrebné aj oprávnenie pre možnosť využitia blokovania a následného upravovania HTTP WebRequest žiadostí.

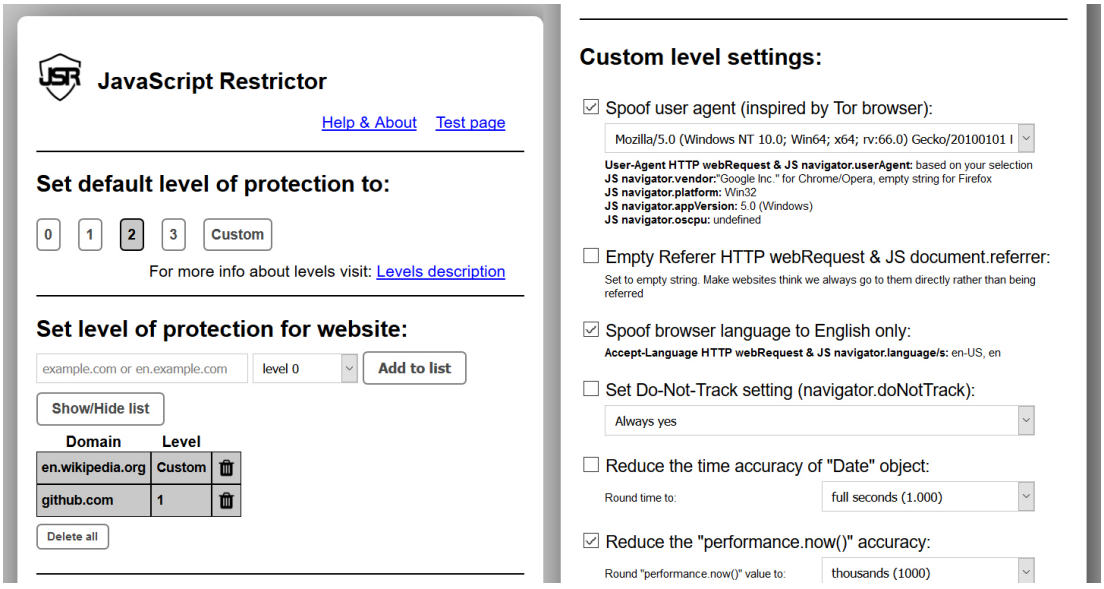
Rozličnosti v manifeste pre Firefox a Chrome, Operu sú iba formálneho charakteru. Napríklad Firefox naviac vyžaduje definovanie kľúča `browser_specific_settings`, kde je nutné uviesť vlastný identifikátor rozšírenia.

7.1.2 Stránka s nastavením

Pre lepšie pochopenie skriptu `document_start.js` a `background.js`, začnem najprv so stránkou nastavenia rozšírenia a jej skriptom. Najprv bol vytvorený front-end stránky, súbormi `options.html, options.css`. Obrázok 7.1 ukazuje výrez stránky s nastavením nášho rozšírenia.

O načítanie a upravovanie aktuálneho stavu nastavení rozšírenia a interakciu užívateľa so stránkou sa stará súbor `options.js`. Všetky nastavenia sa ukladajú do WebExtensions úložiska. Preto bolo potrebné špecifikovať oprávnenie úložiska v manifeste. Bolo zvolené synchrónne úložisko, ktorého výhoda spočíva v tom, že sa nastavenia rozšírenia prenášajú naprieč viacerými zariadeniami v rámci jedného užívateľského účtu pre daný prehliadač.

Hneď po načítaní stránky skript zavolá metódu `browser.storage.sync.get()` pre obdržanie všetkých nastavení uložených v úložisku (anglicky *storage*), ktoré sú uložené vo formáte JSON. JSON objekt obsahuje dvojicu `__default__` s hodnotou, čo predstavuje predvolenú, globálnu úroveň ochrany. Ďalej obsahuje dvojice položiek vo formáte *doména: úroveň* pre všetky špecifické domény, ktoré majú nastavené explicitne svoje úrovne. Objekt nakoniec obsahuje aj aktuálny stav nastavenia vlastnej úrovne ochrany definovanej užívateľom.



Obr. 7.1: Ukážka stránky s nastavením. Pre svoju dĺžku rozdelená na 2 časti vedľa seba. Vľavo obrázok zachtýva nastavenie rozšírenia, kedy ako predvolená úroveň je nastavená úroveň 2. V databázi domén so špecifickou úrovňou sú dve domény s ich nastavením úrovne. Vpravo potom vidíme definovanie vlastnej úrovne rozšírenia. Kedy niektoré API sa berú do úvahy a niektoré nie.

Pomocou cyklu sa prejdú všetky položky JSON a na základe logických podmienok sa nájde predvolená úroveň, ktorá sa následne vyznačí užívateľovi pridaním triedy `active` do HTML stránky, ako je zobrazené na obrázku 7.1. Na základe domén v úložisku sa vygeneruje tabuľka, ktorá sa vloží do stránky pomocou JavaScriptu. Ostáva už len načítanie vlastnej úrovne užívateľa. Keďže je vlastná úroveň implementovaná pomocou HTML formulára, načítané hodnoty objektu JSON upravia stav formulára. Načítané hodnoty tak predstavujú aktuálny stav formulára.

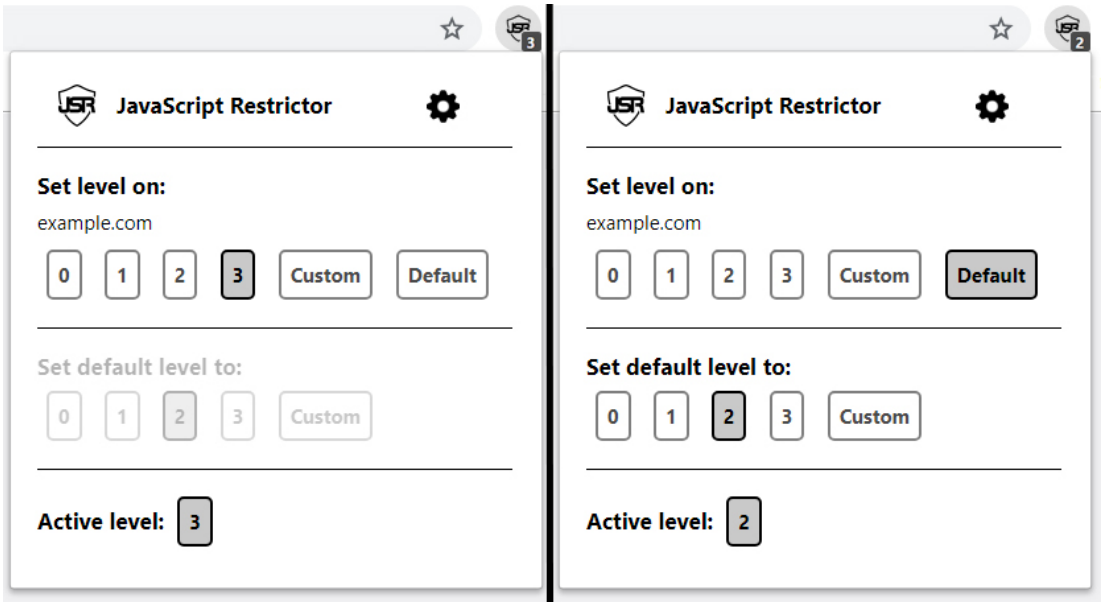
Všetky úpravy nastavení pracujú s úložiskom. V prípade, že užívateľ nastaví globálnu úroveň, okamžite sa zavolá funkcia pre vloženie hodnôt do úložiska. Prepíše sa daná hodnota predvolenej úrovne, a týmto sa rozšírenie udržuje stále aktuálne z pohľadu užívateľa aj rozšírenia. To isté platí pri vložení domény do zoznamu domén. Ak už doména existuje, prepíše sa hodnota úrovne. Ak nie, vloží sa nový záznam do úložiska. Pri mazaní domén z tabuľky sa naopak vymaže daná dvojica z úložiska.

Z dôvodu, že vlastná úroveň je implementovaná formulárom, prepis hodnôt vlastnej úrovne sa vykoná nárazovo, až po stlačení tlačidla pre uloženie nastavenia. Pre každú možnosť sa ukladajú boolean hodnoty *true* a *false*, indikujúce zaškrtnutie nastavenia. Podľa druhu možnosti sa ďalej ukladá o akú možnosť ide, ak je možné špecifikovať konkrétnu možnosť cez výber (element `select`) formulára.

Ďalej boli implementované niektoré maličkosti, aby užívateľ lepšie chápal ovládaniu. Napríklad pri uložení nastavenia vlastnej úrovne sa zmení text tlačidla indikujúci úspešné uloženie (*Saved*). Vzápätí sa text zmení na pôvodný (*Save custom level*), ak užívateľ pokračuje v nastavovaní jednotlivých možností. Taktiež dôjde k vyblednutiu možností miery zaokrúhľovania geolokačných dát, ak užívateľ nastaví ich nulovanie, aby to signalizovalo, že hodnoty zaokrúhľovania sú pri nulovaní irelevantné.

7.1.3 Vyskakovacie okno

Vyskakovacie popup okno môžeme chápať ako zjednodušenú verziu stránky s nastavením s rozdielom, že je chápaná pre aktuálnu stránku, ktorú užívateľ prezerá. Ukážka popup okna na obrázku 7.2.



Obr. 7.2: Vľavo ukážka explicitného zvolenia úrovne 3 pre doménu *exmaple.com*. Predvolená (*default*) úroveň vybledne, čo signalizuje užívateľovi, že sa používa špecifický level pre doménu. Vpravo ukážka aktívnej predvolenej úrovne 2.

Rovnako ako pri stránke s nastavením, bol najprv vytvorený statický front-end okna z návrhu okna pomocou súborov `popup.html, popup.css`. Okno obsahuje ikonu rozšírenia a tlačidlo pre rýchly prechod do stránky s nastavením.

Dynamickosť okna zaručuje súbor `popup.js`. Po kliknutí na ikonku JSR v paneli nástrojov prehliadača, rozšírenie zobrazí popup okno. Skript v pozadí okna podobne ako pri stránke s nastaveniami zavolá funkciu pre prístup do úložiska a obdržané dáta začne spracovávať. Zistí sa aktuálne URL stránky, ktorú užívateľ prezerá, pri ktorej popup okno vyskočilo. URL sa spracuje do podoby samotnej domény, s tým že sa odstráni aj subdoména *www*. Napríklad z `http://www.fit.vutbr.cz/study/DP/` sa URL zredukuje na `fit.vutbr.cz`. Tento zredukovaný reťazec aktuálne navštívenej stránky sa začne hľadať na základe logických podmienok v úložisku cyklom, ktorý prechádza všetky položky v JSON objekte. V prípade, že sa nájde daná doména s odpovedajúcou úrovňou, zvýrazní sa v okne daná úroveň domény a vybledne časť ukazujúca globálnu úroveň (obrázok 7.2 vľavo). Ak sa nenájde doména, úroveň domény sa zvýrazní ako predvolená (obrázok 7.2 vpravo).

Zároveň pri hľadaní domény v JSON objekte úložiska, sa pri nájdení globálnej predvolenej úrovne v okne zvýrazní aktuálne nastavenie pre globálnu úroveň.

V poslednej časti okna sa zobrazuje text s aktuálne aplikovanou úrovňou na stránke. Pri hľadaní aktuálnej domény webovej stránky sa vždy zisťuje aj koreňová doména, v prípade, že užívateľ má nastavenú úroveň pre koreňovú doménu ale pre jej subdoménu nie. Podmienky pri prechádzaní JSON objektu úložiska preto najprv zisťujú, či sa nenašla priama zhoda subdomény, kedy sa cyklus v danom momente ukončí. V prípade, že sa našla zhoda iba

koreňovej domény cyklus prehľadáva ďalej objekt kým nedôjde na koniec. V tom prípade sa zobrazí text, ktorý informuje užívateľa, že na tejto subdoméne sa aplikovala explicitne nastavená úroveň koreňovej domény. Tento stav zachycuje obrázok 7.3.

Obr. 7.3: Ukážka aktívnej úrovne subdomény *fit.vutbr.cz* založenej na explicitne nastavenej úrovni pre koreňovú doménu *vutbr.cz*.

7.1.4 Skript na pozadí

Doteraz sme si popisovali skôr funkčnosť a princíp užívateľského rozhrania ako princíp obaľovania a modifikovania jednotlivých API jazyka JavaScript. Nasledujúci popis spoločne so sekciou 7.1.5 nám hlbšie predvedie postupy pri modifikovaní a podvrhovaní jednotlivých hodnôt popísaných v sekcii 6.2.1.

Skript `background.js` ako už samotný názov naznačuje, je skript na pozadí rozšírenia JSR. Tento skript nemá prístup k upravovanej stránke. Jeho funkcionalita sa využíva pri udalostiach spojených s inštaláciou rozšírenia, udalostiach spojenými so samotným prehliadačom, ako napríklad aktuálne otvorené karty prehliadača a aktualizáciou odznaku pri ikonke rozšírenia v paneli nástrojov. Ďalej sa musí využiť pri odosielaní upravených HTTP WebRequest hlavičiek, pretože WebRequest žiadosti vznikajú nezávisle na načítanom stave stránky.

Pri nainštalovaní alebo aktualizácii rozšírenia sa spustí funkcia `installUpdate()`, ktorá sa postará o prvotné načítanie nastavení rozšírenia. Najprv sa pokúsi skript získať JSON objekt nastavení rozšírenia z úložiska, ak ho získa, rozšírenie už je prítomné v prehliadači a ide len o aktualizáciu. Teda ponechajú sa dané nastavenia z predchádzajúcej verzie. V opačnom prípade sa jedná o novú inštaláciu. Do úložiska sa vloží pomocou volania `browser.storage.sync.set()` globálna úroveň ochrany (ako predvolená úroveň bola zvolená úroveň 2). Ďalej sa vloží počiatočné nastavenie vlastnej úrovne rozšírenia. Toto nastavenie predstavuje JSON objekt so všetkými hodnotami nastavenými na *false*, čo znamená, že sa žiadne API neobaľuje v počiatočnom stave vlastnej úrovne. Tento objekt sa nachádza priamo v kóde skriptu, a teda volá sa iba daná premenná, ktorá tento objekt predstavuje a vloží sa priamo do úložiska. Týmto je počiatočné nastavenie rozšírenia hotové.

Skript sa stará o zmenu odznaku, teda textu pri ikonke JSR v paneli nástrojov. Text odznaku tvorí pre stručnosť iba jeden znak. Na základe aplikovanej úrovne, je to buď číslo úrovne alebo znak *C* ako skratka z anglického *Custom level*, ktorá jasne hovorí, že je aplikovaná vlastná úroveň definovaná užívateľom. Keďže skript na pozadí nemá vedomosť o tom, aká bola aplikovaná úroveň na danú stránku, potrebuje si to pri každom prepnutí medzi otvorenými kartami prehliadača kontrolovať a na základe toho následne nastaviť text odznaku. Kontroluje to rovnakým spôsobom, ako skript `popup.js` zisťuje, ktorá úroveň je aktívna. Pomocou cyklu prehľadávaním JSON objektu úložiska sa pre konkrétne URL aktívnej karty prehliadača hľadá odpovedajúca aktivovaná úroveň. Aby bol schopný skript pristupovať ku kartám prehliadača, bolo potrebné definovať v manifeste rozšírenia oprávnenie `tabs`. Vždy sa na každej udalosti predstavujúcu zmenu karty spustí funkcia pre overenie a nastavenie odznaku, aby bol odznak stále aktuálny. V prípadoch kedy užívateľ nenačítal

webovú stránku, napríklad užívateľ otvoril novú kartu prehliadača, alebo je na nastaveniach rozšírenia, tak odznak nie je potrebný a skryje sa.

Dostávam sa k poslednej oblasti o ktorú sa stará `background.js`. Je to odchytenie a úprava WebRequest hlavičiek pred odoslaním. Poslednými oprávneniami, ktoré bolo potrebné v manifeste definovať boli práve oprávnenia `webRequest, webRequestBlocking`, ktoré oprávňujú rozšírenie pozdržať odosielanie WebRequest hlavičiek a modifikovať ich.

Aby bolo možné modifikovať WebRequest žiadosti, bolo nutné vytvoriť poslucháča udalostí (anglicky *event listener*) s príslušnými parametrami pre blokovanie hlavičiek. Následne sa na základe aktuálnej úrovne, ktorá sa získala pri aktualizovaní odznaku ikonky, rozhoduje, ako sa hlavičky upravia. Upravovanie hlavičky `User-Agent` je ale pri úrovni 2 závislé na reálnom prehliadači užívateľa. Keďže JavaScript Rrestrictor môže upraviť položku `window.navigator.userAgent`, nemôžeme sa spoliehať na zisťovanie prehliadača takto, ale bolo potrebné hneď v úvode skriptu pomocou dostupnosti objektu `browser` alebo `chrome` identifikovať, o aký prehliadač sa jedná (kód 7.1). Toto zisťovanie muselo nastať ešte pred samotným zjednotením objektov `browser` a `chrome`. Pomocou nasledujúceho kódu, je možné zistiť, či sa jedná o Firefox alebo Chrome, Operu.

```
var isFirefox;
if ((typeof browser) !== "undefined") {
  isFirefox = true;
} else {
  isFirefox = false;
}
```

Kód 7.1: Zisťovanie prehliadača užívateľa nezávisle na `window.navigator.userAgent`.

Následne je možné upraviť hlavičku `User-Agent` a podvrhnúť prehliadač podľa úrovne popísaný v sekcii 6.2.3. Pri zisťovaní aktuálnej úrovne sa vždy nastavujú aj príznaky, pomocou ktorých sa neskôr v prípade potreby upravujú aj hlavičky `Accept-Language` a `Referer`, pretože nie vždy pri každej úrovni ochrany je potrebné tieto hlavičky upravovať ako bolo popísané v návrhu (sekcia 6.2.3).

7.1.5 Skript s prístupom k obsahu stránky

Ako posledný si popíšeme skript `document_start.js`. Tento skript je vlastne jadro celého rozšírenia. Zaručuje obaľovanie vybraných API. Najprv sa zistí, aký prehliadač používa užívateľ spôsobom kódu 7.1, aby bolo možné podvrhnúť API `userAgent`, `vendor` aj na základe skutočného prehliadača.

Rovnakým spôsobom, ako tomu bolo pri `popup.js`, sa zisťuje, aká úroveň má byť aplikovaná na danú načítavanú stránku. Keďže tento skript sa spúšťa pri zostavovaní DOM objektu stránky, URL stránky je už možné zo skriptu zistiť. Tým sa zistí z úložiska úroveň, ktorú je potrebné aplikovať na stránku. Potom sa načíta daná úroveň vo formáte JSON objektu pomocou premennej, ktorá sa nachádza priamo v skripte pre každý level osobitne. Formát objektu je rovnaký ako je formát vlastnej úrovne, ktorú je možné nastaviť cez formulár na stránke s nastavením.

Obaľovanie API prebieha postupne prechádzaním JSON objektu a zisťovaním boolean hodnôt pre dané API. Ak sa dané API bude obaľovať, vytvorí sa nový element typu `skript`, do ktorého sa vloží obaľovací kód. Tento element s kódom sa potom vloží za `html` element načítavanej stránky. Nasledujúci úsek kódu 7.3 nám ukazuje príklady zapuzdrenia niektorých API.

```
(function() {
    Object.defineProperty(navigator,"deviceMemory", {
        get: function () { return 4; },
        set: function (a) {},
        configurable: false
    });
}) ();
// alebo
(function() {
    Object.defineProperty(navigator,"platform", {
        get: function () { return "Win32"; },
        set: function (a) {},
        configurable: false
    });
}) ();
```

Kód 7.2: Príklad zapuzdrenia vlastností objektu `window.navigator`.

Tento spôsob zapuzdrenia vlastností objektu zaručuje, že sa bude vždy vracať určená hodnota vlastnosti a zároveň nie je možné vlastnosť znova konfigurovať.

Pri obaľovaní objektu `window.Date` je potrebné myslieť aj na to, že vývojári webových stránok si môžu v prípade potreby rozšíriť prototyp objektu a pridať doň nové vlastnosti. Aby bolo možné po obalení pôvodného objektu volať aj tieto vlastnosti, využili sme funkciu `Object.setPrototypeOf()`. Funkcia nám zaručí, že pri pristupovaní k neznámym vlastnostiam pre obalený objekt sa tieto volania delegujú na pôvodný objekt.

Ďalší kód ukazuje zapuzdrenie `HTMLCanvasElement.prototype.toDataUrl()`. Môžeme si všimnúť, že sa pôvodná funkcia volá nad upraveným kontextom Canvas elementu, ktorého každý pixel pôvodného obrázka sa upraví na úplne biely pixel, pričom zachováva rovnakú veľkosť obrázka. Týmto sa tak dosiahne ochrana Canvas fingerprintigu, pred stránkami, ktoré na to využívajú funkciu `toDataURL()`.

```
(function(){
    var origToDataURL = HTMLCanvasElement.prototype.toDataURL;
    HTMLCanvasElement.prototype.toDataURL = function(type, encoderOptions) {
        var ctx = this.getContext("2d");
        ctx.fillStyle = "white";
        ctx.fillRect(0, 0, this.width, this.height);
        return origToDataURL.call(this, type, encoderOptions);
    };
})();
```

Kód 7.3: Zapuzdrenie funkcie `HTMLCanvasElement.prototype.toDataUrl()`.

7.2 Publikovanie rozšírenia

Pred zverejnením rozšírenia, boli vytvorené domovské stránky[1], kde užívatelia nájdu popis rozšírenia a môžu vyhľadať pomoc, či zanechať spätnú väzbu. Zdrojové kódy rozšírenia sú dostupné na GitHube[2]. Taktiež bola vytvorená aj testovacia stránka[3], ktorá názorne

[1] https://polcak.github.io/jsrestrictor/
[2] https://github.com/polcak/jsrestrictor
[3] https://polcak.github.io/jsrestrictor/test/test.html

ukazuje hodnoty všetkých obaľovaných API a pri zmenách úrovní môže užívateľ sledovať, ako sa menia dané hodnoty API.

Aby sme sa vyhli a predišli nečakaným problémom zo strany oficiálnych obchodov rozšírení prehliadačov Mozilla Firefox, Google Chrome a Opera, navrhol som zverejnenie rozšírenia ešte pred finálnou podobou JSR, a to z toho dôvodu, aby sa zistili všetky potrebné postupy a hlavne časové závislosti pri zverejňovaní. Prvá verzia bola publikovaná 27. februára. Po dokončení implementácie bola publikovaná druhá verzia rozšírenia 9. apríla. Následne sa po konzultáciách s vedúcim práce ešte upravilo obaľovanie objektu `window.Date` a finálna verzia rozšírenia sa publikovala 5. mája.

Rôzne obchody pristupujú k zverejneniu rôzne. Firefox Add-ons vykonáva všetky kroky automatizovane a zverejnenie rozšírenia nastáva takmer okamžite. Na rozdiel od toho, Chrome Web Store, ako aj Opera addons vyžaduje manuálnu kontrolu kódu, čo môže trvať niekoľko dní až týždňov. V našom prípade to bolo pre Chrome Web Store 9 dní, Opera addons 4 dni. Každý obchod naviac vyžadoval presné a špecifické vloženie popisov, ukážok rozšírenia a poprípade aj propagačného loga pre rozšírenie.

Rozšírenie je tak dostupné na stiahnutie a nainštalovanie cez Firefox Add-ons[4], Chrome Web Store[5] a Opera addons[6].

[4] https://addons.mozilla.org/en-US/firefox/addon/javascript-restrictor/

[5] https://chrome.google.com/webstore/detail/javascript-restrictor/ammoloihpcbognfddfjcljgembpibcmb

[6] https://addons.opera.com/en/extensions/details/javascript-restrictor/

Kapitola 8

Testovanie

Táto kapitola sa venuje testovaniu a následnému zhodnoteniu výsledkov testovania implementovaného rozšírenia JavaScript Restrictor. Samotnému testovaniu princípu zapuzdrovania a dokázaniu funkčnosti obaľovania jednotlivých funkcií sa venovala práca [36] pri vytváraní prototypu. My sa viac zameriame v prvej časti testovania na otestovanie funkčnosti webových stránok pri používaní JSR a v druhej časti si overíme, či naozaj rozšírenie zvyšuje mieru anonymity užívateľa na internete.

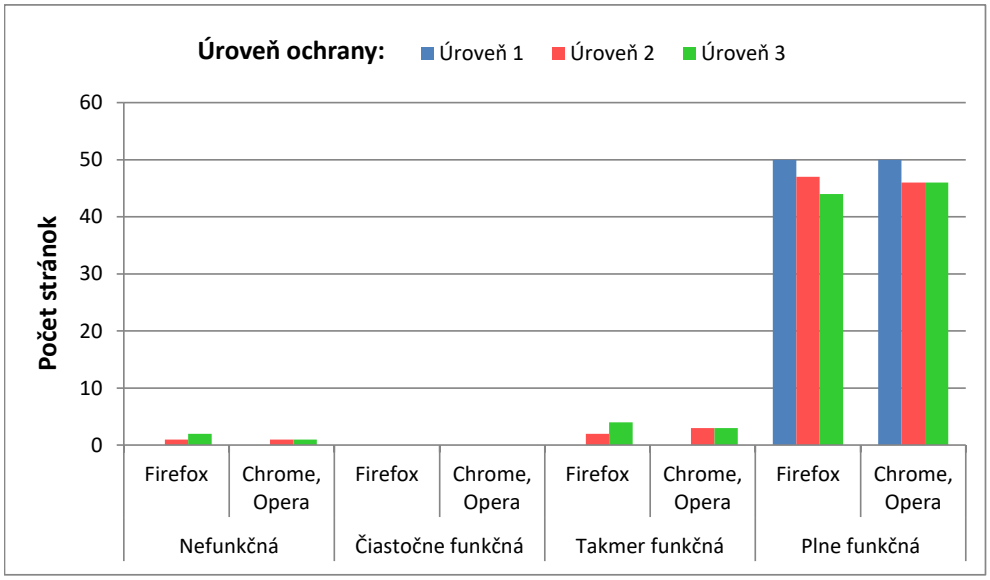
8.1 Testovanie funkčnosti webových stránok

V tejto časti sa pozrieme na funkčnosť webových stránok pri zapnutom rozšírení. Keďže jedným z našich cieľov je pri chránení užívateľa nezlomiť funkčnosť prehliadaných stránok, vytvoril som zoznam s piatimi desiatkami stránok a testoval ich funkčnosť. Testovanie spočívalo v simulovaní užívateľa mnou. Teda klikal som na odkazy na stránke, skúšal prípadné multimédia na webe, v prípade špecifických funkcií som testoval využitie týchto funkcií, v neposlednom rade som na základe daného testovaného webu skúšal aj prihlasovanie či odhlasovanie do a z užívateľských účtov.

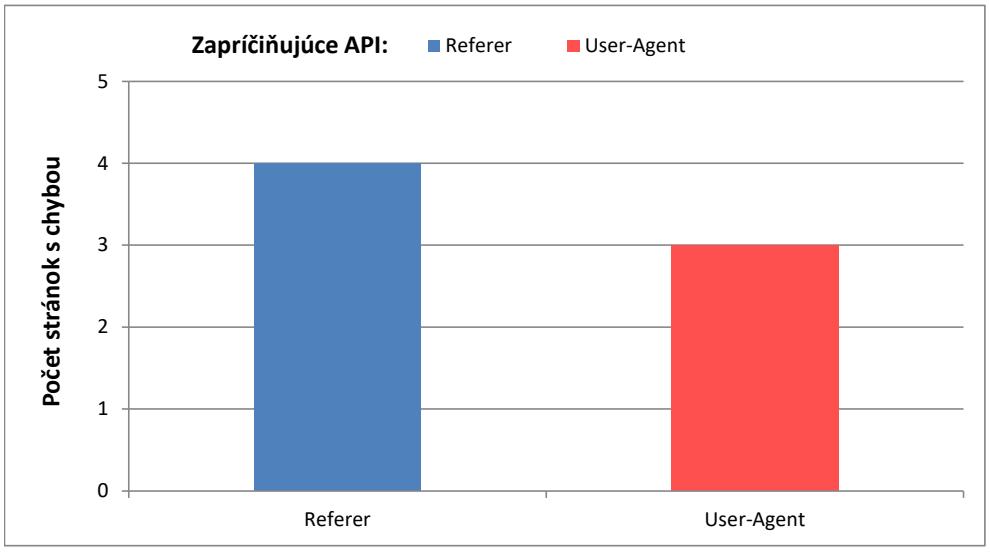
Testovanie prebehlo na osobnom počítači Lenovo Y50 na operačnom systéme Ubuntu 18.04 a Windows 10. Každý web bol testovaný na Mozille Firefox verzie 66, Google Chrome 73 a Opere 58. Aby bolo možné spozorovať abnormálne správanie, najprv bol každý web prehliadaný na úrovni 0, teda pri vypnutom JSR. Až následne na úrovniach 1 až 3. V prípade, že sa zistila nejaká vada alebo chyba a stránka bola narušená, pomocou konzoly prehliadača a vlastnej úrovne nastavenia JSR sa zisťovalo, čo je problémom a kde vzniká. Hodnotenie funkčnosti webov je nasledovné:

- **Stránka nefunkčná**: ak sa testovaná stránka nedá vôbec používať. Jej funkčnosť bola nadmerne narušená do takej miery, že nie je možné na danom webe vykonávať akúkoľvek činnosť.
- **Stránka čiastočne funkčná**: ak bola značne narušená funkčnosť stránky, no zároveň to neviedlo k úplnému znefunkčneniu webu.
- **Stránka takmer funkčná**: v prípade, že sa našla nejaká vada na stránke, ale nezabraňuje to plnému alebo takmer plnému využitiu webu.
- **Stránka plne funkčná**: tento stav je označením, že je stránka plne funkčná a použiteľná. Užívateľ nepozoruje žiadnu zmenu oproti vypnutému rozšíreniu a tento stav je zároveň stavom, ktorý chceme dosiahnuť.

Prvých 30 stránok bolo vybraných na základe rebríčkov najnavštevovanejších a najlepšie hodnotených stránok sveta [2], [33], pre reprezentáciu celosvetovo známych webov. Ďalších 20 bolo zvolených pre overenie funkčnosti viac regionálnych stránok, teda pre otestovanie geograficky lokálnejšej sféry. Boli vyberané známe stránky v českom a slovenskom regióne. Celý zoznam testovaných stránok a výsledky pre každú stránku osobitne nájdeme vo forme tabuľky v prílohe A. Graf 8.1 zobrazuje počty testovaných webov pre jednotlivé ohodnotenia funkčnosti osobitne pre Firefox a Chrome, Operu. Graf 8.2 ukazuje, ktoré API a koľko krát spôsobovali nefunkčnosti stránok.



Graf 8.1: Počty funkčnosti testovaných stránok pre Firefox a Chrome, Operu.



Graf 8.2: API spôsobujúce chyby a ich početnosť v rámci testovaných stránok.

Z 50 testovaných stránok je na úrovni 1 každá stránka plne funkčná. Netflix bol nefunkčný na úrovniach ochrany 2 a 3. Vo Firefoxe je nefukčný YouTube na úrovni 3. Zbytok

je buď plne funkčný alebo takmer plne funkčný. Tieto nefunkčnosti vznikli z dôvodu podvrhnutia `User-Agent` hodnôt cez JavaScript či modifikovaním WebRequest žiadostí a z dôvodu prázdnej `Referer` hodnoty. Ostatné obaľované API problémy na stránkach nespôsobovali.

Google stránky vrátane Google Máp fungovali správne, až na stránky *docs.google.com* a *drive.google.com*, kedy pri úrovni 2 a 3 pre Chrome a Operu neprebehlo načítanie posledných otvorených dokumentov alebo súborov na disku. Dôvodom bola prázdna WebRequest hlavička `Referer`. Na Firefoxe sa tento problém pri úrovni 2 nevyskytoval, čo môže ukazovať na rozličnú implementáciu *docs.google.com* a *drive.google.com* pre prehliadače bežiace na Chromium (Chrome, Opera) a ostatných prehliadačoch ako Firefox. Úroveň 3 pri podvrhnutí `User-Agent` na Chrome znefunkčnila stránku aj pre Firefox, keďže web očakával skutočný prehliadač Chrome, čím Firefox nieje.

YouTube bol plne použiteľný okrem tretej úrovne vo Firefoxe. Dôvodom je spomínaná rozdielna implementácia Google stránok. Keďže sa v úrovni 3 podvrhuje prehliadač Chrome, YouTube očakáva Chrome, no v skutočnosti je to Firefox. Konkrétne tento problém spôsobuje, že YouTube v prípade detekcie prehliadača založeného na Chromium využíva funkciu `window.document.registerElement`. Táto funkcia ale nie je vo Firefoxe podporovaná, a teda táto skutočnosť vedie k chybe na tejto stránke.

Problém s upravovanými WebRequest hlavičkami `Referer` som zaznamenal pri Twittri, kde sa nezobrazovali niektoré odkazy na externé stránky mimo sociálnej siete. Na sociálnej sieti Pinterest pre upravovanie hlavičky nefungovalo prihlásenie a odhlásenie zo sociálnej siete. Okrem tejto vady bola stránka plne funkčná. Pokiaľ je užívateľ prihlásený trvalo bez nutnosti zadávania prihlasovacích údajov pri návšte stránky, stránka je plne funkčná. Netflix nebol funkčný pri upravovaní hlavičiek `Referer`, pretože priamo vyžaduje hodnoty týchto hlavičiek pri spúšťaní videí.

Pri Bing mapách nefunguje na tretej úrovni vo Firefoxe zoom máp pomocou kolieska myši. Dôvodom je podvrhnutie prehliadača na Chrome. Pomocou tlačidiel na stránke je ale stále možné približovať mapu. Iné vady neboli zistené.

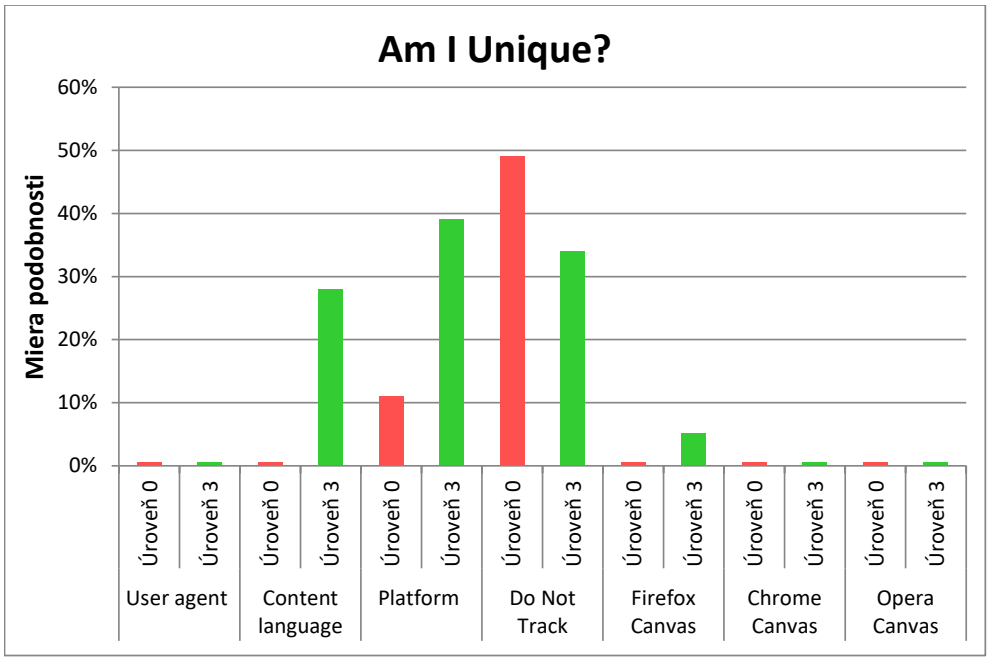
8.2 Overenie zvýšenia anonymity

Jedným z cieľov tejto práce bolo zvýšiť anonymitu užívateľa na internete. Pre zisťovanie miery anonymity využijeme nástroje, ktoré testujú jedinečnosť užívateľa a porovnávajú ju s inými testovanými užívateľmi.

Nástroj Am I Unique?[1] [3] vracia mieru podobnosti v percentách získaného otlačku prehliadača v porovnaní s inými otestovanými užívateľmi. To znamená, že väčšie percento predstavuje väčšiu mieru podobnosti a menšiu mieru jedinečnosti medzi užívateľmi. Graf 8.3 nám zhrnie pozorované výsledky pri testovaní.

Pri úrovniach 2 a 3, kedy dochádza k blokovaniu Canvas fingerprintingu [1] a podvrhnutia operačného systému spolu s prehliadačom vykazoval najväčšie zmeny. Kým podobnosť získaného fingerprintu na základe obrázka bola pri vypnutej ochrane iba 0,23%, pri zapnutej ochrane na Firefoxe to činilo až 5,20%. Je to rapídne zlepšenie. Rovnakú mieru podobnosti dostaneme aj pri používaní prehliadača Tor. Na druhej strane pri prehliadačoch Chrome a Opera bol síce získaný obrázok biely, no napriek tomu bola miera podobnosti iba menej ako 0,1%. Dôvodom je, že funkcia `toDataURL` vracia rozdielne dátové URI reťazce v prehliadačoch založených na Chromium a v prehliadačoch ako je Firefox alebo Tor, ktorý je založený na technológiách Firefox.

[1] https://amiunique.org/



Graf 8.3: Graf zobrazuje možnú mieru podobnosti v percentách podľa nástroja Am I Unique?. Ukázané sú API, ktoré JSR ovplyvňuje. Testované na Ubuntu 18.04 s preferovaným slovenským jazykom vo všetkých prehliadačoch.

Prehliadač Chrome pri položke WebGL Renderer pri testovaní uvádzal hodnotu *Google SwiftShader*[2], čo je Renderer využívaný prehliadačom Chrome. No Firefox a Opera zobrazovala dokonca presné označenie grafickej karty (napríklad označenie integrovanej grafickej karty *Intel(R) HD Graphics 4600* na testovacom stroji). Pri zapnutej ochrane proti Canvas fingerprintingu tieto hodnoty boli neutralizované a bola vrátená hodnota *Not supported* pri všetkých prehliadačoch (weby využívajúce WebGL[3] ale naďalej fungujú), čo taktiež prispieva z zvýšeniu anonymity, kedy stránka nie je schopná zistiť označenie grafickej karty (presné čísla podobnosti naprieč užívateľmi Am I Unique? neposkytuje).

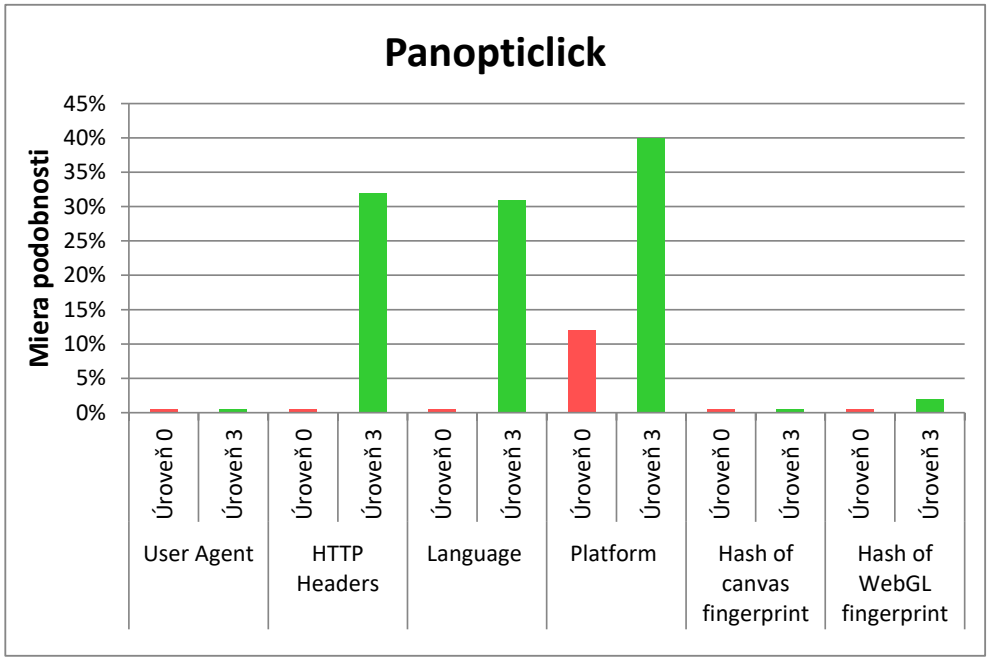
V prípade, že užívateľ využíva Linuxový operačný systém a má ako predvolené jazyky prehliadača aj iné ako angličtinu, užívateľ pri podvrhnutí operačného systému môže jasne sledovať nárast anonymity z 11% na 39%. Pri podvrhnutí predvolených jazykov z menej ako 0,1% pri slovenčine a češtine na 28% pri angličtine ako jedinom predvolenom jazyku.

V dobe testovania sú verzie podvrhnutých prehliadačov dostatočne nové a tieto hodnoty ešte nie je možné jednoznačne vyhodnotiť, pretože počet užívateľov s najnovšou verziou ešte len vzrastá. Do Not Track vykazuje pokles percent podobnosti. Je to spôsobené tým, že zatiaľ neprevláda trend, aby väčšina prehliadačov udávala túto hodnotu (nastavenie tejto hodnoty však aj napriek zníženej miere podobnosti môže mať pozitívny vplyv pre užívateľa, spomenutý v sekcii 6.2.1).

Druhý testovaný nástroj Panopticlick[4] vykazoval podobné zvýšenie hodnôt ako Am I Unique?, ktoré sú uvedené v grafe 8.4. Napríklad pri podvrhnutí predvolených jazykov sa z hodnoty menej ako 0,1% dostaneme na približne 32%. Pri podvrhnutí operačného systému môžeme jasne sledovať nárast anonymity z 12% na 40%. Hodnota Canvas fingerprintingu

[2] https://opensource.google.com/projects/swiftshader
[3] Príklad webov využívajúcich WebGL: https://www.awwwards.com/websites/webgl/
[4] https://panopticlick.eff.org/

bola iba menej ako 0,1% pri zapnutej ochrane aj pri Firefoxe aj Chrome a Opere. Svetlou stránkou je, že získaná hash hodnota Cavasu sa ale líšila pri vypnutej a zapnutej ochrane, čo znamená, že Panopticlick nebol schopný získať pôvodný hash, a teda bol Panopticlick zavádzaný naším rozšírením. Panopticlick vracia iba absolútne hodnoty v počte rovnakých otlačkov užívateľov z celkového počtu testovaných, ktorý bol v čase testovania okolo 350000 získaných otlačkov, a preto boli dané hodnoty prepočítané na percentá pre lepšiu čitateľnosť.



Graf 8.4: Graf zobrazuje možnú mieru podobnosti v percentách podľa nástroja Panopticlick. Ukázané sú API, ktoré JSR ovplyvňuje. Testované na Ubuntu 18.04 s preferovaným slovenským jazykom vo všetkých prehliadačoch.

Jednotlivé percentá sa môžu pri opätovnom testovaní líšiť, pretože množina porovnávaných otlačkov sa aktualizuje vzhľadom na návštevnosť nástrojov Am I Unique? a Panopticlick, ktoré som využil pri testovaní. Iné stránky poskytujúce získavanie otlačkov prehliadača vykazovali podobné hodnoty ako vyššie spomenuté nástroje ale neponúkali porovnanie s inými užívateľmi[5] alebo množina získaných otlačkov bola menšia[6] ako napríklad pri nástroji Panopticlick.

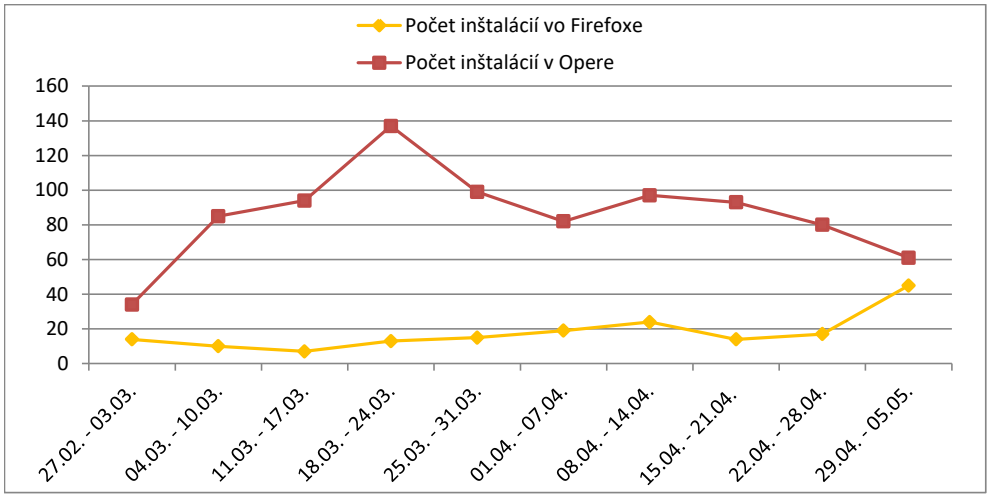
Pri testovaní môžeme sledovať nárast anonymity pri obaľovaných API, čo ukazuje, že obaľovanie skutočne funguje a zväčšuje sa veľkosť anonymitnej množiny [25] používateľa pri využití JSR. Pre skutočné otestovanie úrovní ochrany bolo potrebné vymazať zakaždým testovaním cookies daného webu. Keďže niektoré hodnoty si mohli nástroje ukladať, výsledky by mohli byť skreslené. Firefox pri testovaní niekedy nepredvídateľne neaktualizoval hodnoty pre danú úroveň a bolo potrebné opätovné načítanie stránky. Chrome a Opera pri zmenách reagovali ihneď bez problémov.

[5] https://valve.github.io/fingerprintjs2/
[6] https://browserprint.info/

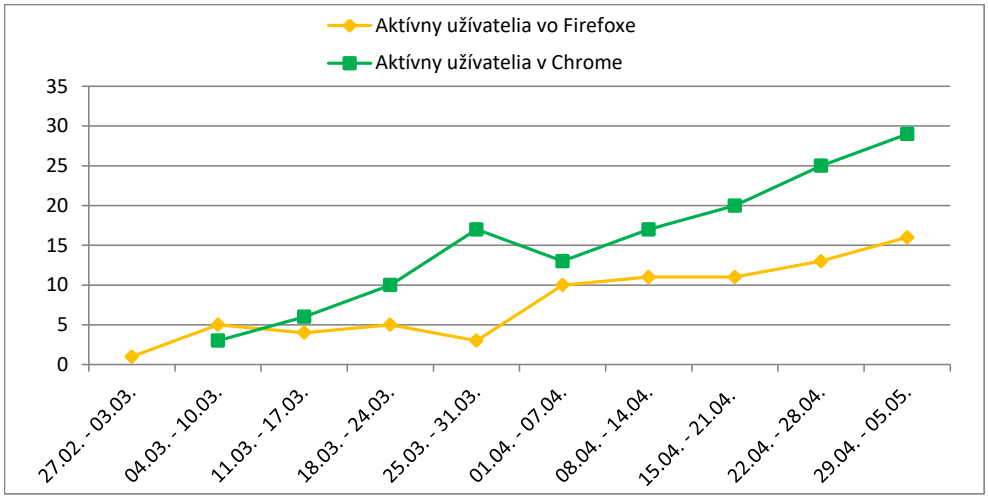
8.3 Populárnosť rozšírenia

V tejto sekcii sa pozrieme, ako sa v rámci prehliadačov JSR rozšírenie presadilo. Prvá verzia rozšírenia bola zverejnená 27. februára. Druhá 9. apríla a tretia 5. mája. Uzávierka zberu štatistík bola 5. mája. Nasledujúce grafy ukazujú sťahovanie a používanie rozšírenia v rámci prehlaidačov. Môžeme si všimnúť že na Opere bolo najviac stiahnutí – inštalácií. Môže to byť z dôvodu, že dostupných rozšírení na tento prehliadač je podstatne menej ako na Firefoxe alebo Chrome a užívatelia tak natrafia na JSR častejšie.

Počet aktívnych užívateľov je počítaný priemerne v rámci týždňa pre Firefox a Chrome, Opera túto štatistiku neposkytuje. Počet stiahnutí je súčet denných stiahnutí za daný týždeň pre Firefox a Operu. Chrome sa potýka s problémami pri tejto štatistike. Už aj rýchlym nahliadnutím bolo zjavné, že počet inštalácií neodpovedá počtu aktívnych užívateľov.



Graf 8.5: Štatistiky inštalácií rozšírenia JavaScript Restrictor v rámci prehliadačov. Oficiálne štatistiky prevzaté z Firefox Add-ons, Opera addons.



Graf 8.6: Štatistiky aktívnych užívateľov rozšírenia JavaScript Restrictor v rámci prehliadačov. Oficiálne štatistiky prevzaté z Firefox Add-ons, Chrome Web Store.

Kapitola 9

Záver

V rámci tejto diplomovej práce som sa zoznámil s problematikou tvorby webových rozšírení popísanou v kapitole 2 a možnosťami ako pomocou rozšírení zvýšiť anonymitu a ochranu užívateľa na internete. Cieľom práce bolo vylepšiť prototyp rozšírenia pre zvýšenie ochrany súkromia užívateľa popísaný v kapitole 4 vytvorený Ing. Zbyňkom Červinkom [36]. V mojej práci bola analyzovaná v kapitole 3 práca JavaScript Zero [30], ktorá podobne ako prototyp rozšírenia [36] využíva princíp zapuzdrenia objektov a premenných jazyka JavaScript, čím sa zamedzi prístup k obalenej časti kódu. Tieto objekty a premenné jazyka tak naberajú iba lokálnu viditeľnosť. Rozšírenie vie následne modifikovať návratové hodnoty obalených objektov či funkcií jazyka. Vhodným modifikovaním návratových hodnôt je možné zvýšiť ochranu a súkromie užívateľa na internete. Popis iných existujúcich rozšírení pre zvýšenie súkromia je možné nájsť v kapitole 5. Rozšírenie JavaScript Restrictor bolo navrhnuté v kapitole 6. Popis implementácie a publikovania môjho rozšírenia je popísaný v kapitole 7. Nakoniec bolo v kapitole 8 opísané, ako sa rozšírenie testovalo a či spĺňa kritériá zvýšiť anonymitu ale zároveň neporušiť funkčnosť webových stránok.

Vytvorené rozšírenie je pod názvom JavaScript Restrictor dostupné na stiahnutie a nainštalovanie cez Firefox Add-ons[1], Chrome Web Store[2] a Opera addons[3]. Zdrojové kódy je možné nájsť na GitHube[4] a na pribalenom CD k tejto diplomovej práci.

Rozšírenie bolo prezentované na konferencii Excel@FIT 2019[5] 26. apríla vo forme plagátu, kedy sa mohli návštevníci konferencie dozvedieť viac o rozšírení JavaScript Restrictor. Aj účasť na tejto konferencii dopomohla k propagácii rozšírenia. Celkovo bolo ku dňu 5. mája (deň uzávierky zberu štatistík) rozšírenie pre všetky dostupné prehliadače nainštalované viac ako 1000-krát. Počet aktívnych užívateľov je vrámci všetkých dostupných prehliadačov spolu viac ako 50.

Možnosti ďalšej práce na rozšírení sú naozaj široké. Mohlo by nasledovať vyladenie ako užívateľského rozhrania, tak aj zaviesť niektoré vyžadované zmeny od získanej spätnej väzby užívateľov. Na stránke podpory sa objavil od jedného užívateľa návrh[6], ako lepšie prispôsobiť rozšírenie na Android prehliadače, ktoré podporujú Chrome rozšírenia, čo je

[1] https://addons.mozilla.org/en-US/firefox/addon/javascript-restrictor/

[2] https://chrome.google.com/webstore/detail/javascript-restrictor/ammoloihpcbognfddfjcljgembpibcmb

[3] https://addons.opera.com/en/extensions/details/javascript-restrictor/

[4] https://github.com/polcak/jsrestrictor

[5] http://excel.fit.vutbr.cz/

[6] https://github.com/polcak/jsrestrictor/issues/22#issue-433955292

dobrý námet na vylepšenie rozšírenia a zameranie sa aj na mobilné zariadenia, nie len na desktop počítače, ako to bolo v tejto práci.

Ďalšie možnosti vylepšenia rozšírenia spočívajú v hlbšom skúmaní stránok, ktoré nefungovali správne, ktoré boli popísané v sekcii 8.1 a následné opravenie týchto vad. Ďalšou možnosťou je spojenie JavaScript Restrictor s Chrome Zero (popísaný v sekcii 3.3). Nepochybne je určite možné nájsť ďalšie vylepšenia v rámci pridania nových obaľovaných objektov či funkcií jazyka JavaScript alebo vylepšenie ochrany voči pokročilejším technikám získavaniu otlačkov zariadenia. Pri zaslaní HTTP požiadavky s hlavičkou Content-Security-Policy serverom je pri aktuálnej verzii rozšírenia možné v prehliadači Firefox deaktivovať celé rozšírenie. Ďalším krokom by teda mohlo byť skúmanie možných chýb v terajšej implementácii a ich opravenie.

Naskytuje sa aj možnosť optimalizácie rozšírenia. Napríklad pomocou vytvorenia vlastných zoznamov aktuálne otvorených kariet prehliadača a k nim priradených úrovní ochrany by pri prepínaní kariet prehliadača bolo možné meniť odznak v ikonke rozšírenia indikujúci aktívnu úroveň ochrany aj bez nutnosti opätovného čítania dát z úložiska rozšírenia. V neposlednom rade by určite vylepšenie rozšírenia spočívalo v automatickom zisťovaní novo dostupných verzií populárnych prehliadačov a aktualizácii podvrhovaných verzií, namiesto nutnosti manuálneho zásahu do kódu rozšírenia.

Literatúra

[1] Acar, G.; Eubank, C.; Englehardt, S.; aj.: *The Web Never Forgets: Persistent Tracking Mechanisms in the Wild*. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS 14, 2014.
URL https://securehomes.esat.kuleuven.be/~gacar/persistent/the_web_never_forgets.pdf

[2] Alexa Internet, Inc.: *The top 500 sites on the web*. [Online; navštíveno 3.4.2019].
URL https://www.alexa.com/topsites

[3] Amiunique.org: *About Am I Unique?* [Online; navštíveno 20.4.2019].
URL https://amiunique.org/about

[4] Arrieta, A.: *A New Welcome to Privacy Badger and How We Got Here*. [Online; navštíveno 21.12.2018].
URL https://www.eff.org/deeplinks/2018/04/new-welcome-privacy-badger-and-how-we-got-here

[5] Bernstein, D. J.: *Cache-timing attacks on AES*. 2005.

[6] Cassidy, R.: *A How To Guide to Ad Blocking*. [Online; navštíveno 21.12.2018].
URL https://blog.getadblock.com/a-how-to-guide-to-ad-blocking-be452ed5ed6f

[7] Europa.eu: *Ktoré osobné údaje sa považujú za citlivé?* [Online; navštíveno 21.11.2018].
URL https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/legal-grounds-processing-data/sensitive-data/what-personal-data-considered-sensitive_sk

[8] FullStory: *The Definitive Guide to Session Replay*. [Online; navštíveno 21.11.2018].
URL https://www.fullstory.com/resources/the-definitive-guide-to-session-replay/

[9] Gruss, D.; Bidner, D.; Mangard, S.: *Practical Memory Deduplication Attacks in Sandboxed JavaScript*. ESORICS'15, 2015.

[10] Gruss, D.; Maurice, C.; Mangard, S.: *Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript*. DIMVA'16, 2016.

[11] Haverbeke, M.: *Eloquent javascript: a modern introduction to programming*. No Starch Press, 2018, ISBN 978-1593279509.

[12] Kounine, A.: *How your personal data is used in personalization* . [Online; navštíveno 21.11.2018].
URL https://www.tastehit.com/blog/personal-data-in-personalization-and-advertising

[13] Lemonweb s.r.o.: *Dodržujete "cookie" zákon? Je na čase...* [Online; navštíveno 21.11.2018].
URL https://www.lemonweb.sk/blog/dodrzujete-cookie-zakon-je-na-case

[14] Mehrnezhad, M.; Toreini, E.; Shahandashti, S. F.; aj.: *TouchSignatures: Identification of User Touch Actions and PINs Based on Mobile Sensor Data via JavaScript.* Journal of Information Security and Applications, 2016.

[15] Mehta, P.: *Creating Google Chrome extensions.* Apress, Berkeley, CA, 2016, ISBN 978-1-4842-1775-7.

[16] Mowery, K.; Shacham, H.: *Pixel Perfect: Fingerprinting Canvas in HTML5.* Proceedings of W2SP 2012. IEEE Computer Society, 2012.

[17] Mozilla.org: *Add-ons.* [Online; navštíveno 10.10.2018].
URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons

[18] Mozilla.org: *Anatomy of an extension.* [Online; navštíveno 10.10.2018].
URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension

[19] Mozilla.org: *Browser support for JavaScript APIs.* [Online; navštíveno 3.4.2019].
URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Browser_support_for_JavaScript_APIs

[20] Mozilla.org: *Hardware Across the Web.* [Online; navštíveno 3.4.2019].
URL https://data.firefox.com/dashboard/hardware

[21] Mozilla.org: *WebRequest.* [Online; navštíveno 20.4.2019].
URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest

[22] Mozilla.org: *What are extensions?* [Online; navštíveno 10.10.2018].
URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/What_are_WebExtensions

[23] Nikiforakis, N.; Kapravelos, A.; Joosen, W.; aj.: *Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting.* 2013 IEEE Symposium on Security and Privacy, pages 541–555, 2013.

[24] Perekalin, A.: *Why you should be careful with browser extensions.* [Online; navštíveno 16.10.2018].
URL https://www.kaspersky.com/blog/browser-extensions-security/20886/

[25] Pfitzmann, A.; Hansen, M.: *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management.* 2010.
URL https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[26] Polčák, L.: *Zákonné odposlechy: detekce identity*. Dizertačná práca, FIT VUT v Brne 2017.

[27] Q-Success: *Usage of JavaScript for websites*. [Online; navštíveno 10.1.2019]. URL https://w3techs.com/technologies/details/cp-javascript/all/all

[28] Refsnes Data: *Browser Statistics*. [Online; navštíveno 3.4.2019]. URL https://www.w3schools.com/browsers/default.asp

[29] Refsnes Data: *OS Platform Statistics*. [Online; navštíveno 3.4.2019]. URL https://www.w3schools.com/browsers/browsers_os.asp

[30] Schwarz, M.; Lipp, M.; Gruss, D.: *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks*. NDSS Symposium 2018, 2018.

[31] Schwarz, M.; Maurice, C.; Gruss, D.; aj.: *Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript*. FC'17, 2017.

[32] Shapiro, J.: *Encapsulation in JavaScript*. [Online; navštíveno 20.11.2018]. URL https://www.intertech.com/Blog/encapsulation-in-javascript/

[33] SimilarWeb Ltd.: *Top Websites Ranking*. [Online; navštíveno 3.4.2019]. URL https://www.similarweb.com/top-websites

[34] Simpson, K.: *You Don't Know JS: Scope & Closures*. O'Reilly Media, 2014, ISBN 978-1449335588.

[35] Vila, P.; Kopf, B.: *Loophole: Timing Attacks on Shared Event Loops in Chrome*. USENIX Security Symposium, 2017.

[36] Červinka, Z.: *Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí*. Diplomová práca, FIT VUT v Brne 2018.

Príloha A

Zoznam testovaných stránok

Nasledujúca tabuľka A.1 obsahuje názvy webov a ich funkčnosť, ktoré boli testované pri daných úrovniach ochrany rozšírenia JavaScript Restrictor.

Číslo znázorňuje úroveň ochrany rozšírenia JSR. Skratka *ff* pri úrovni označuje danú úroveň špecificky na prehliadači Mozilla Firefox. Skratka *ch* pri úrovni označuje danú úroveň špecificky na prehliadačoch Google Chrome a Opera.

Stránka	Nefunkčná	Čiastočne funkčná	Takmer funkčná	Plne funkčná
google.com	-	-	2 ch, 3	1, 2 ff
youtube.com	3 ff	-	-	1, 2, 3 ch
facebook.com	-	-	-	1, 2, 3
wikipedia.org	-	-	-	1, 2, 3
yahoo.com	-	-	-	1, 2, 3
amazon.com	-	-	-	1, 2, 3
twitter.com	-	-	2, 3	1
live.com	-	-	-	1, 2, 3
instagram.com	-	-	-	1, 2, 3
reddit.com	-	-	-	1, 2, 3
blogspot.com	-	-	-	1, 2, 3
netflix.com	2, 3	-	-	1
linkedin.com	-	-	-	1, 2, 3
twitch.tv	-	-	-	1, 2, 3
aliexpress.com	-	-	-	1, 2, 3
microsoft.com	-	-	-	1, 2, 3
bing.com	-	-	3 ff	1, 2, 3 ch
ebay.com	-	-	-	1, 2, 3
github.com	-	-	-	1, 2, 3
stackoverflow.com	-	-	-	1, 2, 3
msn.com	-	-	-	1, 2, 3
office.com	-	-	-	1, 2, 3
imdb.com	-	-	-	1, 2, 3
ampproject.org	-	-	-	1, 2, 3
whatsapp.com	-	-	-	1, 2, 3
pinterest.com	-	-	2, 3	1
paypal.com	-	-	-	1, 2, 3

Stránka	Nefunkčná	Čiastočne funkčná	Takmer funkčná	Plne funkčná
duckduckgo.com	-	-	-	1, 2, 3
craigslist.org	-	-	-	1, 2, 3
vutbr.cz	-	-	-	1, 2, 3
fit.vutbr.cz	-	-	-	1, 2, 3
george.csas.cz	-	-	-	1, 2, 3
mapy.cz	-	-	-	1, 2, 3
seznam.cz	-	-	-	1, 2, 3
superhry.cz	-	-	-	1, 2, 3
livesport.cz	-	-	-	1, 2, 3
csfd.cz	-	-	-	1, 2, 3
jizdnirady.idnes.cz	-	-	-	1, 2, 3
bazos.sk	-	-	-	1, 2, 3
alza.sk	-	-	-	1, 2, 3
heureka.sk	-	-	-	1, 2, 3
bestbonus.sk	-	-	-	1, 2, 3
shmu.sk	-	-	-	1, 2, 3
sme.sk	-	-	-	1, 2, 3
dennikn.sk	-	-	-	1, 2, 3
topky.sk	-	-	-	1, 2, 3
tvnoviny.sk	-	-	-	1, 2, 3
mojevideo.sk	-	-	-	1, 2, 3
f1.sk	-	-	-	1, 2, 3
cycling-info.sk	-	-	-	1, 2, 3

Tabuľka A.1: Tabuľka zobrazuje mieru funkčnosti testovaných webových stránok podľa aplikovanej úrovne ochrany rozšírenia JavaScript Restrictor na danú stránku.