VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ROZŠÍŘENÍ PRO WEBOVÝ PROHLÍŽEČ ZAMĚŘENÉ NA OCHRANU SOUKROMÍ

PRIVACY-PRESERVING WEB BROWSER EXTENSION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ZBYNĚK ČERVINKA**
AUTHOR

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

BRNO 2018

Zadání diplomové práce/21274/2017/xcervi16

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Červinka Zbyněk, Bc.**
Obor: Informační systémy
Téma: **Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí**
Privacy-Preserving Web Browser Extension
Kategorie: Bezpečnost

Pokyny:

1. Nastudujte rozhraní WebExtensions používané webovými prohlížeči.
2. Seznamte se s přístupy ke sledování uživatele využívajícího protokol HTTP(S).
3. Popište aktuální možnosti rozšíření webových prohlížečů zaměřujících se na ochranu soukromí na Internetu.
4. Po dohodě s vedoucím práce navrhněte nové rozšíření webového prohlížeče, či upravte stávající tak, aby byla vylepšena ochrana soukromí na Internetu.
5. Návrh implementujte.
6. Vytvořené rozšíření otestujte na různých webových stránkách.
7. Zhodnoťte výsledky projektu a navrhněte další možná zlepšení.

Literatura:

- Kaplas, J. (2016). Possibilities and usability of various privacy preservation browser add-ons. Bakalářská práce, Lappeenranta University of Technology, Finsko, dostupné online http://www.doria.fi/handle/10024/129993
- Ikram, M., Asghar, H., Kaafar, M., et al. (2016). Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning. *Proceedings on Privacy Enhancing Technologies*, 2017(1), str. 79-99.
- Yu, Z.,, Macbeth, S., Modi, K., a Pujol, J.M. (2016). Tracking the Trackers. *Proceedings of the 25th International Conference on World Wide Web* (WWW '16), str. 121-132.
- Další po konzultaci s vedoucím.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Polčák Libor, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá bezpečností, soukromím a anonymitou uživatelů na internetu. V práci jsou popsány sledovací mechanismy a přístupy, které monitorují a odesílají informace o uživatelích. Takto uniknuté informace mohou být následně použity k identifikaci konkrétního uživatele, monitorování a analýze jeho chování na konkrétních webových stránkách nebo mohou být tyto informace zneužity jiným způsobem (např. odcizení a zneužití hesla nebo čísla platební karty). V rámci této práce je popsána a otestována funkcionalita a spolehlivost současných doplňků webových prohlížečů slibující uživatelům ochranu při pohybu na internetu. Následně je v rámci této práce navržen a implementován doplněk, který demonstruje přístup ke zvýšení soukromí a anonymity prostřednictvím techniky redefinování a zapouzdření původní implementace řady JavaScriptových funkcí a objektů, a to v době před začátkem zpracování načítané webové stránky. Spouštění obalovacího kódu v této době zajistí, že žádný jiný kód v načítané webové stránce nebude nikdy moci využívat původní implementaci. Doplněk je v rámci práce také důkladně testován. Závěr práce se zabývá celou řadou dalších možností rozšíření implementovaného doplňku.

Abstracts

This thesis deals with security, privacy and anonymity on the internet. In this thesis are described tracking mechanisms and approaches that are being used to collect and send away users' personal information. Information that leaks using this tracking approaches can be used to identify user, to monitor and analyse his behaviour on specific web pages and several leaked pieces of information can be misused (for example the leaked credit card number or password). In this thesis is described and tested the functionality and reliability of several current web add-ons providing the protection on the internet. New security increasing web add-on has been designed and developed to demonstrate technique, that redefines and wraps the original JavaScript implementation of several functions and objects, the wrapping is executed before the visited web page starts processing the source code. Running the wrapping code at this time will ensure, that no other code in loaded web page will ever have acces to the original implementation. This add-on is also well-tested. The final thesis‘ stage provides great amount of possibilities to improve implemented add-on.

Klíčová slova

Anonymita na internetu, ochrana soukromí uživatelů internetu, bezpečnost, HTTP, HTTPS, monitorování, WebExtensions, JavaScript.

Keywords

Anonymity on the Internet, privacy of internet users, security, HTTP, HTTPS, monitoring, WebExtensions, JavaScript.

Citace

ČERVINKA, Zbyněk. *Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polčák, Ph.D.

Rozšíření pro webový prohlížeč zaměřené na ochranu soukromí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Libora Polčáka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Zbyněk Červinka

20. května 2018

Poděkování

Rád bych tímto poděkoval Ing. Liboru Polčákovi, Ph.D. za odbornou pomoc, konzultace a čas věnovaný vedení mé diplomové práce.

Obsah

Kapitola 1

Úvod

S postupem času a rozvojem technologií začíná být soukromí návštěvníků webových stránek stále více narušováno monitorovacími mechanismy pro sběr informací o uživatelích pohybujících se po internetu [1, 2, 3]. Tyto mechanismy dokáží detailně a v reálném čase monitorovat chování uživatelů na jednotlivých internetových stránkách. Tato činnost probíhá bez vědomí monitorovaných uživatelů a představuje nejen narušení soukromí těchto uživatelů, ale v celé řadě situací může být spojena s potenciálním zneužitím takto nasbíraných dat. Příkladem takové situace je získání a následné zneužití hesel nebo čísel kreditních karet, které byly zachycené v rámci procesu monitorování návštěvníků webových stránkách [1]. Dalším příkladem narušení soukromí uživatele internetu může být také sbírání a šíření informací popisující chování uživatele na jednotlivých stránkách. Takovým chováním může být například, jaké stránky v rámci určité domény uživatel navštívil nebo jak dlouhý čas strávil zobrazováním určité části webové stránky. Pokročilé monitorovací mechanismy sbírají i podrobnější údaje o čase, výkonu počítače nebo GPS datech návštěvníka webové stránky [4, 6, 7, 8, 9].

V minulosti [1] byli uživatelé na internetu identifikování a monitorování především podle stránek, které navštěvovali. Monitorovat jakékoli chování uživatelů na konkrétní webové stránce bylo oproti dnešním možnostem velice omezené. Byly využívány nástroje jako například cookies nebo evidence IP adres [1]. Tyto přístupy jsou sice ve velkém používány i dnes a představují silný nástroj pro sběr informací a tvorbu statistik o chování uživatelů na internetu, jejich schopnosti však nepostačují k detailnímu monitorování chování uživatelů na konkrétním v reálném čase.

V současné době se na internetu začínají ve velkém využívat metody a přístupy, které dokáží sbírat detailní informace o chování uživatelů na konkrétních webových stránkách [7, 8, 9]. Dochází tak k podrobnému monitorování chování uživatelů v reálném čase, což představuje hlubší zásah do soukromí uživatelů a celou řadu dalších potenciálních bezpečnostních hrozeb. Tyto hrozby mají podobu úniků a dalšího možného zneužití celého spektra osobních informací uživatelů. Mezi sledované chování uživatelů patří pohyby myši, pohyb po webové stránce nahoru a dolů, údery uživatele do klávesnice, ale i události typu spuštění videa nebo kliknutí na reklamu a podobné. Z dosavadních výzkumů bylo zjištěno, že monitorovací mechanismy ve velkém také odesílají tato osobní data uživatelů. Vzhledem k tomu, že probíhá monitorování úderů do klávesnice, monitorovací služby běžně odesílají data z vyplněných formulářů, a to i předtím, než uživatel formulář odešle. Na servery monitorovacích služeb se tak dostávají i osobní informace jako čísla kreditních karet, rodná čísla a také hesla. Na serverech monitorovacích služeb jsou poté tato data skladována a často i prohlížena provozovateli webových stránek, kteří si nechávají dění na svých webových stránkách monitorovat za účelem odhalení chyb nebo optimalizace svých služeb.

Tato práce je zaměřená na zvýšení ochrany soukromí uživatelů internetu, a to prostřednictvím analýzy současných přístupů k monitorování uživatelů internetu, dále analýzy současných možností ochrany uživatelů internetu [1, 2, 3, 8, 9] a následného návrhu a implementace nového doplňku webového prohlížeče, který zvýší ochranu uživatelů internetu. Tento doplněk využívá techniku redefinování a obalení původní implementace JavaScriptových konstrukcí, a to takovým způsobem, že původní implementace je zapouzdřena uvnitř prostoru s lokální platností proměnných. K redefinici původní implementace dochází ještě před začátkem zpracování zdrojového kódu načítané webové stránky. Tímto je zajištěno, že žádný kód obsažený v načítané a zpracovávané webové stránce nebude nikdy mít přístup k původní implementaci napřímo, bude tedy muset použít pravidla pro využívání původní implementace nastavená obalovacím kódem implementovaného doplňku.

Práce je rozdělena do několika kapitol. Kapitola 2 se zabývá přístupy a technikami využívanými k monitorování uživatelů internetu [1, 7, 8, 9]. Kapitola 3 popisuje současné doplňky webových prohlížečů zaměřené a na ochranu soukromí a srovnává jejich funkcionalitu a nedostatky [1]. V kapitole 4 je popsána technologie WebExtensions, která slouží k tvorbě doplňků pro webové prohlížeče. Kapitola 5 uvádí návrh doplňku, který bude v rámci této diplomové práce implementován. Kapitola 6 je věnována podrobnosti implementace doplňku a kapitola 7 popisuje, jak byl implementovaný doplněk testován. Kapitola 8 se zabývá závěrem této práce.

Kapitola 2

Přístupy k monitorování uživatelů

V této kapitole se nachází definice osobních a citlivých informací a dále také popis technik a přístupů, která prostřednictvím těchto osobních a citlivých informací umožňují identifikaci a sledování uživatelů na internetu. Jsou zde popsány podstaty technik pro monitorování a sbírání informací o uživatelích a také, jaká data jsou sbírána, kam jsou data odesílána a jakým způsobem představuje tato činnost nebezpečí pro uživatele internetu.

2.1 Osobní a citlivé informace

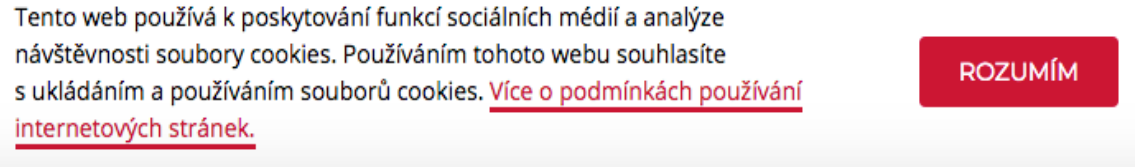
Některé osobní informace mohou být použity k identifikaci osob. Mezi takové informace patří například jméno, e-mail, telefon, adresa, biometrické údaje a další obdobné typy informací, dále také informace popisující počítačovou stanici, na které uživatel navštíví konkrétní internetové stránky. Mezi takové informace patří například údaje o času a výkonu počítače, dále také informace o poloze uživatele a celá řada dalších informací. Celá řada osobních informací může být po odcizení také zneužitelná, mezi takové informace patří například hesla, čísla platebních karet, CVV/CVC[1] platebních karet a data expirací platebních karet [1].

2.2 Monitorování pomocí cookies

Cookies jsou krátké textové záznamy, které umožňují zapamatování informací pro pozdější použití. Data uložená v cookies jsou v rámci klient-server architektury ukládány na straně klienta, tedy v počítači uživatele internetu. K ukládání a modifikaci dat uložených v cookies dochází prostřednictvím webového prohlížeče na straně klienta nebo na straně serveru. Tyto operace provádí samotný server, nebo webový prohlížeč na pozadí, a často se tak děje bez vědomí uživatele [1].

[1] CVV/CVC – „Card Verification Value“ a „Card Verification Code“ jsou hodnoty uvedené na kreditních kartách, bližší informace na https://help.gopay.com/cs/tema/bezpecnost/co-to-je-cvv-cvc-a-kde-se-nachazi

V řadě případech je uživatel informován o skutečnosti, že daná stránka používá cookie, a to prostřednictvím dialogového okna. Na obrázku 1.1 je zachycen příklad takového dialogového okna, které však nabízí pouze možnost odsouhlasení používání cookies prostřednictvím tlačítka „ROZUMÍM“ a také odkaz na podmínky používání internetových stránek. Nenachází se zde možnost odmítnout používání cookies na dané webové stránce.



Obrázek 1.1: Dialogové okno zachycující informaci o používání cookies na dané webové stránce, nachází se zde tlačítko pro potvrzení souhlasu s používáním a odkaz na podrobnější informace o podmínkách používání daných webových stránek, tlačítko sloužící k odmítnutí používání cookies chybí.[2]

Protokol HTTP je nestavovým protokolem a využití cookies představuje důležitý nástroj pro implementaci celé řady funkcí webových stránek. Mezi tyto funkce patří například přihlašování uživatelů do webové služby nebo uchování informací o obsahu nákupního košíku e-shopu zatímco uživatel prochází další stránky. Pokud by zde cookies nebyly využity a celá funkcionality přihlašování by byla implementována pouze s použitím HTTP protokolu, po uskutečnění přechodu na další webovou stránku by již nebyl uživatel přihlášen (a musel by tedy zadávat přihlašovací údaje znovu). V případě funkcionality nákupního košíku by se po přechodu na jinou webovou stránku v rámci stejného e-shopu buď jevil nákupní košík prázdný, nebo by se musela použít alternativní technika jako například uchování dat v URL.

Cookies však mohou být také využívány k monitorování uživatelů. Tyto tzv. sledovací cookies si poznamenávají a uchovávají celou řadu informací o chování uživatele, jako například jaké webové stránky navštívil nebo přes které reklamy uživatel přejel kurzorem myši. Velikost dat uložených v cookies je omezena a není tedy možné poznamenávat si další a další informace o chování uživatele pouhým přidáváním dat na konec záznamu uloženého v cookies. Tento limit však bývá často obcházen způsobem, kdy jsou data o chování uživatele skladována na webovém serveru a v záznamu cookies je uložen pouze jednoznačný identifikátor sledovaného uživatele.

Přestože cookies představují nástroj široce používaný ke sbírání informací o chování uživatelů na internetu, možnosti samotných cookies jsou omezené a nelze je použít ke sbírání detailních dat o uživatelích a monitorování uživatelů v reálném čase.

[2]Dialogové okno pořízeno na webové stránce https://www.kb.cz/

2.3 Monitorování v reálném čase

V současné době se začíná stále více rozšiřovat monitorování uživatelů internetu v reálném čase. Tato technika umožňuje na rozdíl od používání cookies detailní monitorování chování uživatelů na webových stránkách, sesbíraná data jsou poté často přenášena na servery třetích stran, kde jsou skladována a dále analyzována.

2.3.1 Co jsou to „session replay“ scripty

Session replay scripty jsou části kódu vložené do webových stránek, které monitorují chování návštěvníků webových stránek v reálném čase. V této práci budou tyto scipty nazývány sledovacími prvky. Mezi monitorované aktivity patří pohyby myši, pohyb uživatele na stránce nahoru a dolů, údery do klávesnice, ale také události typu kliknutí na reklamu nebo spuštění přehrání videa. Nasbíraná data jsou poté průběžně odesílána prostřednictvím protokolu HTTP na servery poskytovatelů monitorovacích služeb. Zde jsou uskladněna a připravena k dalšímu použití. Data bývají často dále zkoumána, většinou za účelem analýzy chování uživatelů na konkrétních webových stránkách. Celý proces shromažďování, odesílání a skladování nasbíraných dat se děje bez vědomí uživatelů.

2.3.2 Jak se sledovací prvky dostanou do webových stránek

Kód webové stránky je často složen z částí, které pocházejí z různých zdrojů. Jakýkoli kód, který webová stránka poskytuje ze svého vlastního zdroje je označován jako tzv. first-party zdroj. Webové stránky také mohou obsahovat kód třetí strany, tedy kód, který poskytuje třetí strana a je integrován do webové stránky. Běžným příkladem takového obsahu je reklama poskytovaná třetí stranou [1].

Obsah třetích stran je načítán do webové stránky dynamicky při každém zobrazení stránky uživatelem. Obsah třetích stran je většinou skladován na serveru třetí strany a v průběhu času se může měnit. Z toho vyplývá, že ani tvůrce webové stránky často nemá kompletní kontrolu nad celým kódem na své webové stránce.

2.3.3 Jaké informace o uživateli jsou monitorovány

Monitorovací nástroje zaznamenávají celou řadu informací o chování uživatele na konkrétní webové stránce. Mezi monitorované aktivity patří pohyby myši, pohyb uživatele na stránce nahoru a dolů, údery do klávesnice, ale také události typu kliknutí na reklamu nebo spuštění přehrání videa. Nasbíraná data jsou poté vizualizována, aby bylo možné lépe zkoumat chování uživatelů na webových stránkách. Mezi běžně používané vizualizace patří například tzv. heatmapy, které pro daný časový interval dokáže vizualizovat četnost pohybů myši po jednotlivých částech webové stránky. Mezi další běžné způsoby vizualizace patří také videa chování konkrétního uživatele na webové stránce.

2.3.4 Příklady služeb využívající „session replay“ scripty

Mezi provozovatele služeb, nabízející monitorování návštěvníků webů pomocí „session replay“ scriptů, patří například služby SmartLook, HotJar, SessionCam, Yandex, UserReplay a FullStory. Cílem těchto služeb je monitorovat chování návštěvníků webových stránek a

poskytnout jejich administrátorům detailní popis chování návštěvníků. Takto nasbíraná data jsou dále prohlížena a využívána administrátory webů k celé řadě analýz, jako například analýza situací, kdy mají uživatelé obtíže použít některou funkci webové stránky.

Nasbíraná data mohou obsahovat celou řadu osobních a jiných citlivých údajů a každá z těchto služeb takové údaje do jisté míry cenzuruje. Následující tabulka zachycuje, které typy údajů jsou cenzurovány a jakým způsobem.

	SmartLook	SessionCam	Yandex	UserReplay	HotJar	FullStory
Jméno	○	◐	○	◐	○	○
E-mail	○	◐	○	◐	○	○
Telefon	○	◐	○	◐	○	○
Adresa	○	◐	○	◐	○	○
Heslo	◐	●	●	◐	●	●
Číslo platební karty	●	◐	○	◐	◐	●
CVV/CVC platební karty	●	◐	○	◐	○	●
Expirace platební karty	●	◐	○	◐	○	●

● Kompletně vynecháno
◐ Nahrazeno stejně dlouhým řetězcem hvězdiček nebo jiných maskovacích znaků
○ Ponecháno beze změny

Tabulka 2.1: Podpora pro anonymizaci osobních informací u služeb SmartLook, SessionCam, Yandex, UserReplay, HotJar a FullStory.

Automatické cenzurování osobních údajů není vždy úplně spolehlivé a v celé řadě případech dochází k situaci, kdy není soukromá informace cenzurována. Citlivé údaje jsou pak prohlíženy dalšími lidmi a hrozí jejich zneužití.

2.3.5 Monitorování prostřednictvím technologie JavaScript

Sběr dat o uživatelích pohybující se po internetu je realizováno mimo jiné prostřednictvím technologie JavaScript. Technologie JavaScript nabízí širokou škálu nástrojů, které mohou být vykonávány na straně klienta a mají tedy přístup k celé řadě dat, která mohou být citlivá a jejich únik může narušit soukromí uživatelů internetu.

Technologie JavaScript nabízí celou řadu funkcí a objektů obsahující funkce a data, které mají možnost zjišťovat o uživatelích internetu celou řadu potenciálně citlivých informací. Monitorovací nástroje jako například Smartlook využívají JavaScriptový kód k inicializaci funkcionality své služby, a dále také k běhu a sběru dat, které jsou dále posílány přes síť.

2.3.6 Popis služby Smartlook

Služba Smartlook je jedna ze služeb, které využívají monitorování v reálném čase postaveném na technologii JavaScript. V kódu 2.1 je zachycen inicializační kód služby Smartlook, který je službou vygenerován a administrátor webové stránky jej vloží do hlavičky HTML stránky svého webu. Kód se nejdříve ujistí, že objekt *window.smartlook* neexistuje a až poté jej vytvoří. Následně kód zinicializuje příkazem *smartlook('init', '39dfa55283318d31afe5a3ff4a0e3253e2045e43')*, jehož první parametr stanovuje, že při tomto zavolání funkce dojde k inicializaci služby a druhý parametr určuje jednoznačný identifikátor webové stránky monitorované službou Smartlook.

```
<script type="text/javascript">
  window.smartlook || (function(d) {
  var o=smartlook=function() {
    o.api.push(arguments)},
    s=d.getElementsByTagName('script')[0];
    var c=d.createElement('script');
    o.api=new Array();
    c.async=true;
    c.type='text/javascript';
    c.charset='utf-8';
    c.src='//rec.getsmartlook.com/bundle.js';
    s.parentNode.insertBefore(c,s);
  })(document);
  smartlook('init', '39dfa55283318d31afe5a3ff4a0e3253e2045e43');
</script>
```

Kód 2.1: Příklad inicializačního JavaScriptového kódu aplikace Smartlook, který je vygenerován samotným Smartlookem a je vložen do webové stránky, která má být monitorována.

Kapitola 3

Existující doplňky webových prohlížečů zaměřené na ochranu soukromí

Tato kapitola je zaměřena na analýzu a popis současných doplňků webových prohlížečů, které nabízejí zvýšení anonymity a ochranu soukromí uživatelů při procházení internetu. Konkrétně zde budou popsány doplňky *Javascript Control*[1], *NoScript*[2], *AdBlock*[3], *Adblock Plus*[4], *Privacy Badger*[5] a *Ghostery*[6].

Při popisu a testování jednotlivých doplňků byl kladen důraz na zjištění, jaké funkce jednotlivé doplňky nabízejí a které funkce naopak nenabízejí, jaký je princip fungování jednotlivých doplňků, míra schopnosti správně detekovat sledovací a jiné nežádoucí prvky na webové stránce, možnosti konfigurace doplňku (a úroveň znalostí potřebná ke správné konfiguraci) a také schopnost fungovat automaticky bez nutnosti interakce s uživatelem.

V poslední sekci této kapitoly (sekce 3.7) se nachází srovnání všech zkoumaných doplňků. Tato sekce se primárně soustředí na oblasti, ve kterých doplňky neposkytují dostatečnou ochranu soukromí nebo neumožňují jednoduché a pohodlné procházení internetu při používání doplňku. Právě z těchto informací je vycházeno při návrhu (popsán v kapitole 5) a následné implementaci (popsána v kapitole 6) doplňku zaměřeného na ochranu soukromí uživatelů internetu.

3.1 Javascript Control

Nástroj *Javascript Control*[7] je doplněk webového prohlížeče, který umožňuje kompletně zablokovat vykonávání veškerého JavaScriptového kódu na webové stránce. Ihned po instalaci doplňku je funkce blokování JavaScriptu zapnuta a JavaScript je kompletně zablokován na všech nově navštívených webových stránkách. Webové stránky, které byly otevřené a načtené před tím, než došlo k instalaci (nebo aktivaci) doplňku zůstávají funkcionalitou doplňku nedotčeny a JavaScript je na nich tedy plně funkční (pro aktivaci blokování JavaScriptu i na těchto webových stránkách je tedy nutné webovou stránku znovu načíst).

[1] Doplněk Javascript Control je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/javascript-control/
[2] Doplněk NoScript je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/noscript/
[3] Doplněk AdBlock je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/adblock-for-firefox/
[4] Doplněk Adblock Plus je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/adblock-plus/
[5] Doplněk Privacy Badger je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/privacy-badger17/
[6] Doplněk Ghostery je dostupný zde - https://addons.mozilla.org/en-US/firefox/addon/ghostery/
[7] V rámci testování byl použit doplněk Javascript Control ve verzi 1.1

Ovládání doplňku probíhá prostřednictvím ikony, která je umístěna vpravo v adresním řádku webového prohlížeče. Kliknutím na ikonu je možné funkci blokování JavaScriptu zapnout a vypnout. Grafická vzhled ikony se pro zapnutou a vypnutou funkcionalitu doplňku liší, pro zapnutou funkcionalitu, a tedy zablokovaný JavaScript je barva ikony šedá, pro vypnutou funkcionalitu doplňku a tedy povolený JavaScript je barva ikony žlutá (vzhled a umístění ikon je znázorněno na obrázku 3.1).

Obrázek 3.1: Adresní řádky obsahující na pravé straně ikonu doplňku Javascript Control. V horním adresním řádku je ikona žlutá, barva zde symbolizuje zapnutý JavaScript. V dolním adresním řádku je ikona šedá, barva zde symbolizuje zablokovaný JavaScript.

Blokování JavaScriptu je možné vypnout, nebo opětovně zapnout pro každou webovou stránku zvlášť, doplněk si pamatuje volbu uživatele (při příští návštěvě stránky bude doplněk opět nastaven stejným způsobem jako při minulé návštěvě této stránky). V případě, že se uživatel rozhodne důvěřovat zdrojovému kódu určité webové stránky a využít na ní technologii JavaScript, má možnost povolit JavaSctipt jen pro určitou doménu a nemusí vypínat celý doplněk.

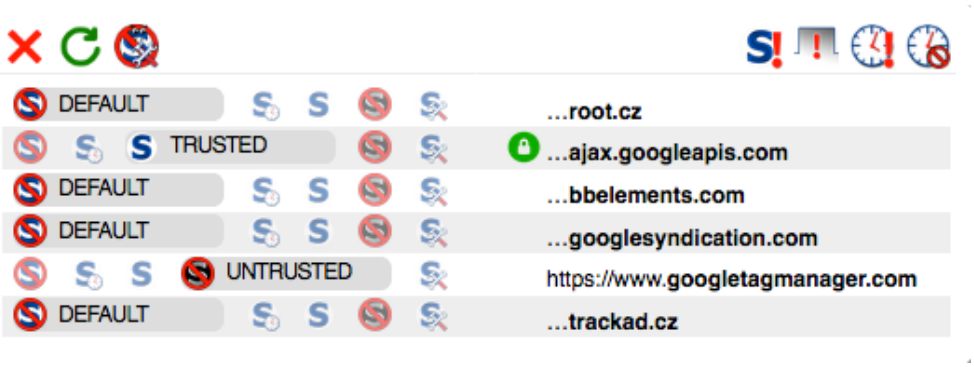
Doplněk nenabízí vedle možnosti zapnutí a vypnutí funkcionality blokování JavaScriptu žádné dodatečné nastavení, doplněk je možné pouze kompletně vypnout, nebo odstranit z webového prohlížeče v okně správy doplňků znázorněném na obrázku 3.2. Ovládání doplňku je tedy jednoduché a intuitivní, není zde potřeba rozhodovat, který JavaScriptový kód zablokovat a který povolit a používání doplňku bez problémů zvládne i méně zkušený uživatel.

Obrázek 3.2: Položka reprezentující nainstalovaný doplněk Javascript Control v okně správy doplňků prohlížeče Mozilla Firefox. Položka nabízí možnost vypnout doplněk, nebo jej kompletně odstranit z webového prohlížeče.

Na druhou stranu doplněk nabízí pouze funkci kompletního zablokování technologie JavaScript v rámci každé webové stránky. V případě aktivace funkcionality blokování pro konkrétní webovou stránku JavaScriptu dojde sice spolehlivě k zablokování veškerého sbírání a odesílání dat prostřednictvím technologie JavaScript, technologie JavaScript je však v dnešní době běžnou součástí většiny webových stránek a může hrát klíčovou roli při správném fungování a zobrazování webové stránky. Jako příklad narušené funkcionality způsobené vypnutým JavaScriptem lze uvést nemožnost přehrát video, nemožnost dynamicky zobrazovat nový obsah na webové stránce nebo nefunkční fotogalerie, která při zapnutém JavaScriptu nabízí možnost zvětšování nebo automatického střídání jednotlivých fotografií.

3.2 NoScript

Doplněk *NoScript*[8] zajišťuje, že JavaScriptvý kód, Java a Flash bude spouštěn a prováděn pouze na důvěryhodných webových stránkách, dále nabízí také ochranu proti technikám *Cross-site scripting*[9] a *Clickjacking*[10].



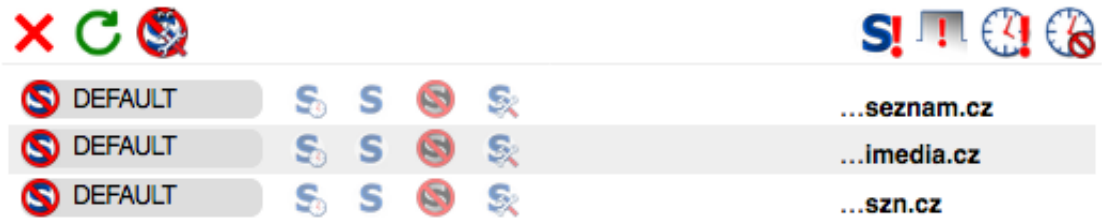
Obrázek 3.3: Uživatelské rozhraní doplňku NoScript dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče.

Na obrázku 3.3 je zachyceno uživatelské rozhraní doplňku NoScript. Toto rozhraní je dostupné po kliknutí na ikonu reprezentující přítomnost aktivovaného doplňku v pravém horním rohu okna webového prohlížeče. V rámci tohoto okna jsou v levém horním rohu znázorněné možnosti pro zavření okna, znovunačtení aktuální webové stránky a tlačítko pro přesun na stránku s nastavením doplňku (tlačítka popisována odleva doprava). V pravém horním rohu tohoto okna jsou dostupná tlačítka pro aktivaci možnosti vypnutí funkcionality celého doplňku na všech webových stránkách, vypnutí funkcionality doplňku pro aktuální webovou stránku (volba uživatele bude zapamatována a při příští návštěvě stejně webové stránky bude aplikována stejná volba, stejné nastavení chování doplňku). Další tlačítko nabízí možnost dočasně důvěřovat všem externě načítaným zdrojům v rámci konkrétní webové stránky. Následující tlačítko umožňuje zrušit volbu předcházejícího tlačítka, tedy umožnit důvěru těm externě načítaným zdrojům, které doplněk NoScript považuje za důvěryhodné a kterým umožní možnost běhu v rámci aktuální webové stránky (tlačítka jsou brána odleva doprava podle návrhu v doplňku NoScript).

Následně je v okně doplňku NoScript seznam celé řady externě načítaných zdrojů, ke každému zdroji je vedena doména, ze které je externí obsah načítán. Doplněk NoScript každý takto detekovaný externě načítaný zdroj posuzuje a dle výsledku posouzení poté zvolí, zda se externě načítanému zdroji bude, nebo nebude důvěřovat, popřípadě, jakým způsobem. Toto posouzení je viditelné v levé části každého pruhu, který symbolizuje externě načítaný prvek. Uživatel má dále možnost tuto volbu změnit kliknutím na jinou volbu, uživatelem provedená volba bude zapamatována i pro příští návštěvy stejné webové stránky.

[8] V rámci testování byl použit doplněk NoScript ve verzi 10.1.8.1
[9] Technika Cross-site scripting je blíže popsána zde - https://cs.wikipedia.org/wiki/Cross-site_scripting
[10] Technika Clickjacking je blíže popsána zde - https://cs.wikipedia.org/wiki/Clickjacking

Při testování v reálném provozu občas docházelo k situaci, při kterých byly zablokovány externě načítané prvky potřebné k provozu webové stránky. Jako příklad je možné uvést webovou stránku https://www.seznam.cz/, kde bylo po instalaci doplňku NoScript viditelné jen prázdné bílé okno. Po prozkoumání seznamu detekovaných a zablokovaných načítaných prvků (seznam prvků zachycen na obrázku 3.4) bylo objeveno, že doplněk NoScript zablokoval načítání dat i ze stejné domény, na které je umístěna aktuálně otevřená webová stránka. Zrovna v tomto případě zablokováním tohoto prvku došlo k zamezení načtení potřebných dat a místo obsahu webové stránky bylo viditelné jen prázdné bílé okno. Povolením právě tohoto prvku (na obrázku 3.4 znázorněn na prvním řádku seznamu nalezených prvků) se po znovunačení webové stránky objeví standardní obsah webové stránky.



Obrázek 3.4: Uživatelské rozhraní doplňku NoScript zobrazující nalezené prvky na doméně https://www.seznam.cz/. Mezi zablokovanými nalezenými prvky se nachází i kód načítaný ze stejné domény, zablokování tohoto kódu došlo k znemožnění zobrazení webové stránky.

3.3 AdBlock

AdBlock[11] je doplněk pro webový prohlížeč, jehož cílem je blokovat na navštívených webových stránkách reklamu a další externě načítané zdroje. Doplněk je plně aktivovaný a funkční ihned po instalaci, jeho funkcionalita se projeví na všech nově načtených webových stránkách.

Na obrázku 3.5, je zachyceno uživatelské rozhraní doplňku AdBlock dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče. Pod logem doplňku AdBlock se nachází statistiky informující o počtu nalezených a zablokovaných externě načítaných sledovacích prvků a dále je zde uveden celkový počet zablokovaných sledovacích prvků na všech stránkách od doby instalace tohoto doplňku. Pod těmito informacemi se nachází tlačítka umožňující uživateli interakci s doplňkem. Nachází se zde možnost deaktivovat funkcionalitu doplňku AdBlock, dále možnost definovat a zablokovat reklamu výběrem objektu na obrazovce, možnost vypnout funkcionalitu doplňku pro konkrétní webovou stránku, nebo možnost vypnout funkcionalitu pro celou aktuální doménu. Dále se zde nachází tlačítko „Show all requests“, které uživatele přesune na stránku, kde se nachází seznam všech dotazů provedených v rámci aktuální webové stránky. Seznam provedených dotazů je rozšířen o informace, zda byl daný dotaz zablokovaný a zda daný prvek byl poskytovaný třetí stranou.

[11] V rámci testování byl použit doplněk AdBlock ve verzi 3.13.2

Obrázek 3.5: Uživatelské rozhraní doplňku AdBlock dostupné po kliknutí na ikonu v pravém horním rohu webového prohlížeče.

Při testování doplňku AdBlock na reálných webových stránkách se ukázalo, že tento doplněk chybuje při určování, která externě načítaná data zablokovat a která povolit. Na obrázku 3.6 je zachycena situace, kdy doplněk AdBlock povolil 8 dotazů a načítání dat na doméně patřící službě Smartlook. Fakt, že doplněk AdBlock službu Smartlook povoluje bylo pozorováno na běžných webových stránkách prostřednictvím funkce „Show all requests“ dostupné v okně doplňku AdBlock. Následně došlo k ověření, že doplněk dané dotazy a načítání povoluje, a to na webové stránce vytvořené za tímto účelem. Tato webová stránka neobsahovala kromě kódu Smartlooku žádný jiný kód, byl založen také účet ve službě Smartlook, aby bylo možné zkontrolovat, zda se návštěva nahrála. Po navštívení testovací webové stránky doplněk indikoval 0 nalezených a zablokovaných prvků, návštěva této stránky byla v aplikaci Smartlook nahrána.

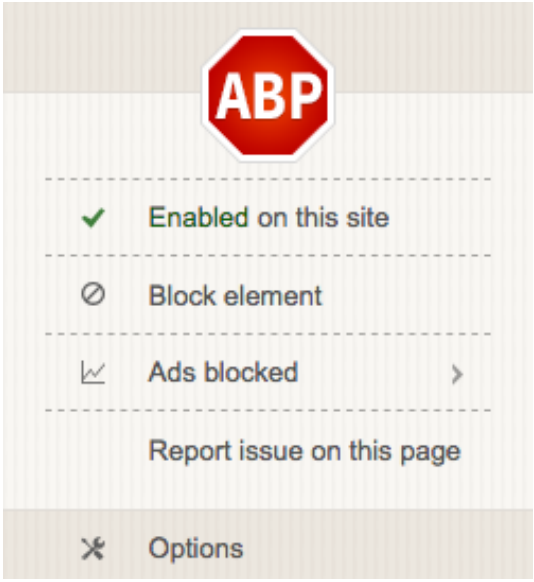
Resource	Type	Matched filter	Third-party
https://rec.getsmartlook.com/bundle.js	script		Yes
https://manager.smartlook.com/rec/check	xmlhttprequest		Yes
https://rec.smartlook.com/analytics-20180509081457.js	script		Yes
https://rec.smartlook.com/bundle-20180509081457.js	script		Yes
https://manager.smartlook.com/rec/init	xmlhttprequest		Yes
https://writer.smartlook.com/rec/write?rid=pEUdxJJaDWY&index=0&tim[...]	xmlhttprequest		Yes
https://manager.smartlook.com/rec/events?rid=pEUdxJJaDWY&sid=nx[...]	xmlhttprequest		Yes
https://writer.smartlook.com/rec/write?rid=pEUdxJJaDWY&index=1&tim[...]	xmlhttprequest		Yes

Obrázek 3.6: Celá řada povolených dotazů a načítání dat z domén patřící službě Smartlook.

3.4 Adblock Plus

Doplněk *Adblock Plus*[12] zajišťuje nejen detekci a blokování reklam, ale také sledovacích prvků. Doplněk funguje automaticky a nevyžaduje žádnou dodatečnou konfiguraci.

Uživatelské rozhraní doplňku je zachyceno na obrázku 3.7. Nachází se zde možnost vypnout doplněk na aktuální stránce, tato volba bude zapamatována při příští návštěvě této webové stránky. Doplněk ve svém okně dále nabízí funkci zablokování reklamy prostřednictvím výběru objektu na webové stránce a také zobrazení statistik, kolik prvků bylo nalezeno a zablokováno na aktuální webové stránce a celkem od nainstalování doplňku. Okno doplňku také nabízí možnost přejít do sekce s nastavením doplňku. V této sekci se nachází možnost konfigurovat vlastní bílou listinu a přidávat nebo ubírat stránky, na nichž bude doplněk Adblock Plus vypnutý. Také se zde nachází možnost konfigurace jazyka a kontaktní údaje na autory doplňku.



Obrázek 3.7: Uživatelské rozhraní doplňku Adblock Plus dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče.

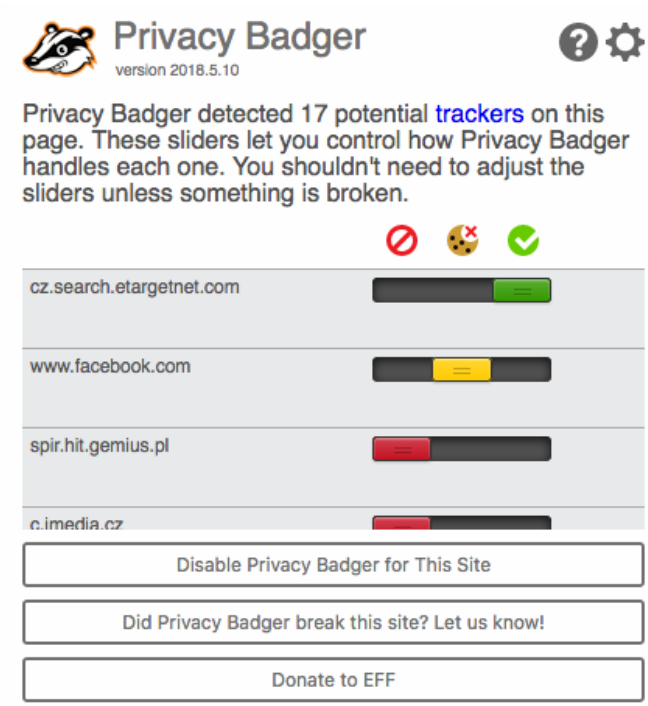
Vývojáři doplňku Adblock Plus si u tohoto nástroje vedou tzv. whitelist s doménami, u kterých je za poplatek zajištěno, že na těchto doménách bude nástroj deaktivovaný a nebude tedy blokovat žádné externě načítané zdroje. V situaci, kdy uživatel navštíví webovou stránku evidovanou na whitelistu aplikace Adblock Plus, neposkytuje tento doplněk uživateli žádnou ochranu soukromí. Uživatel není o této skutečnosti informován.

[12] V rámci testování byl použit doplněk Adblock Plus ve verzi 3.1

3.5 Privacy Badger

Privacy Badger[13] je doplněk webového prohlížeče, který detekuje externě načítané prvky na webové stránce. Tyto prvky jsou poté doplňkem hodnoceny a doplněk rozhoduje, které externě načítané prvky budou povoleny a které budou zablokovány, uživatel má také možnost volbu doplňku změnit prostřednictvím okna uživatelského rozhraní.

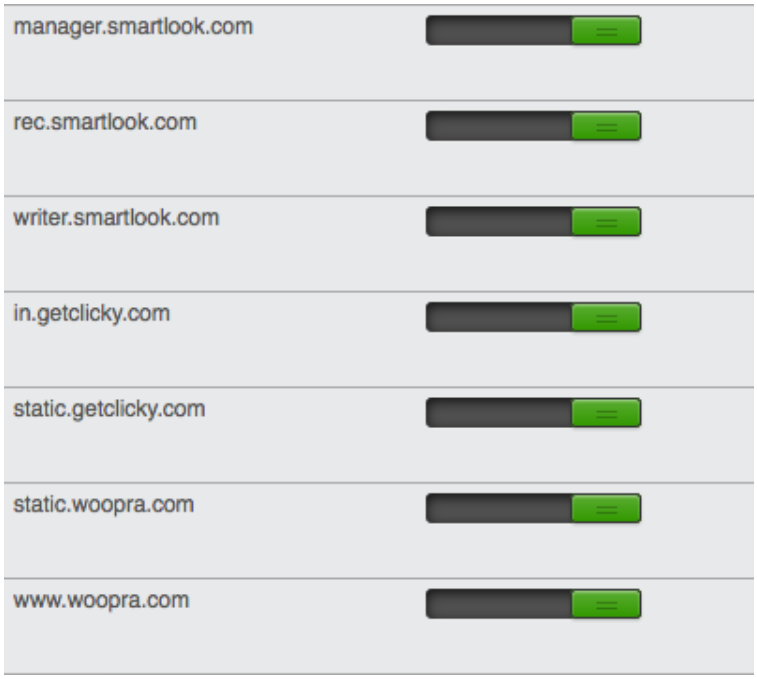
Na obrázku 3.8 se nachází uživatelské rozhraní doplňku Privacy Badger dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče. Toto okno v sobě mimo jiné zachycuje seznam s položkami nalezených prvků, tento seznam je tříděn do dvou kategorií. Nejdříve je uveden seznam sledovacích prvků, které doplněk Privacy Badger hodnotí jako sledovací prvky narušující soukromí uživatele, dále následuje seznam sledovacích prvků, u kterých je nepravděpodobné, že uživatele sledují. Každá položka seznamu reprezentuje jeden sledovací prvek a skládá se z domény sledovacího prvku (URL adresy sledovacího prvku) a z posuvné lišty. Posuvná lišta nabízí 3 polohy – zablokovat sledovací prvek, povolit sledovací prvek, nebo blokovat pouze cookies daného sledovacího prvku. Výchozí nastavení posuvné lišty stanovuje sám doplněk, uživatel může dodatečně měnit volbu doplňku. Uživatelem definovaná nové nastavení bude v doplňku uloženo a bude pro daný prvek aplikováno při příští návštěvě této webové stránky, ale také pro všechny ostatní webové stránky, na kterých bude detekovaný stejný externě načítaný prvek.



Obrázek 3.8: Uživatelské rozhraní doplňku Privacy Badger dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče.

[13] V rámci testování byl použit doplněk Privacy Badger ve verzi 2018.5.10

Při testování na reálných webových stránkách se ukázalo, že doplněk Privaci Badger často chybuje v určování, které externě načítané prvky zablokovat a které nikoli. Na obrázku 3.9 je zachyceno 7 externě načítaných prvků, které jsou doplňkem Privacy Budger hodnoceny jako bezpečné a jsou povoleny. Po prozkoumání detailů a také zdrojového kódu webové stránky bylo prokázáno, že tyto prvky realizují monitorování a sledování uživatele, jedná se nástroje Smartlook[14], Clicky[15] a Woopra[16].



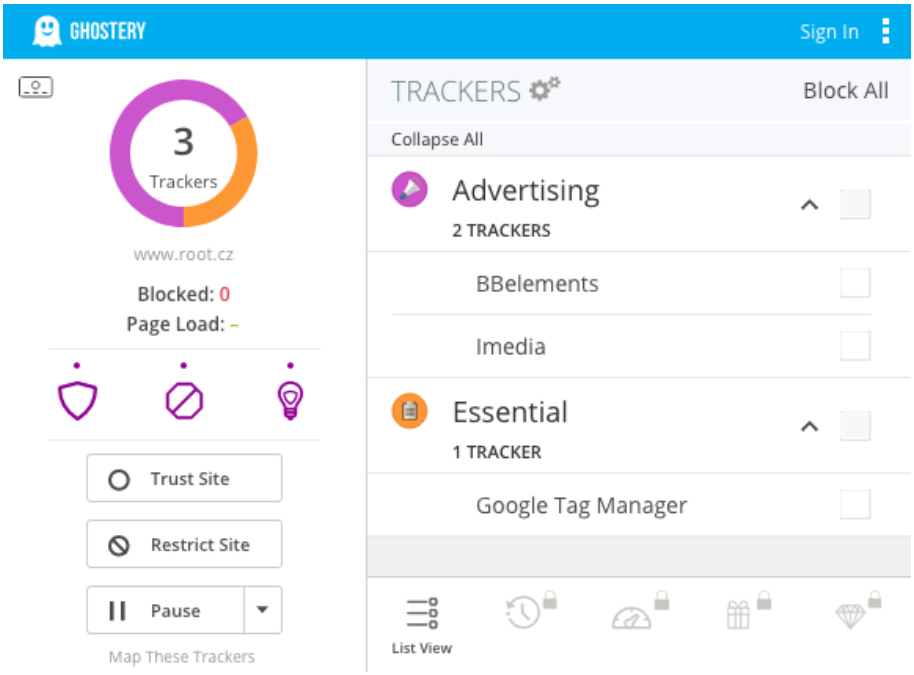
Obrázek 3.9: Automatická detekce doplňku Privacy Badger posoudila služby Smartlook, Clicky a Woopra jako bezpečné a povolila běh jejich kódu.

Při testování doplňku se také ukázalo, že Privacy Badger velice často vyhodnotí externí zdroj nutný pro správnou funkcionalitu stránky jako sledovací prvek, následně dojde k jeho zablokování. V takovém případě dochází k nesprávnému zobrazování nebo fungování stránky. V takovémto případě je nutné v okně uživatelského rozhraní doplňku Privacy Badger použít tlačítko „Vypnout Privacy Badger pro tuto stránku“, nebo ručně nalézt zablokovaný prvek a povolit jej. V případě vypnutí funkcionality doplňku pro konkrétní webovou stránku dochází k situaci, kdy uživatel již není doplňkem na této stránce chráněn. Ruční vyhledání a povolení prvku zase vyžaduje úsilí, které narušuje jednoduchost používání doplňku při procházení internetu, navíc vyžaduje po uživateli určitou míru znalostí pro správné nalezení a určení prvku, jehož konfigurace bude měněna.

[14]Domovská stránka projektu zde - https://www.smartlook.com/
[15] Domovská stránka projektu zde - https://www.clicky.com/
[16] Domovská stránka projektu zde - https://www.woopra.com/

3.6 Ghostery

Doplněk webového prohlížeče *Ghostery*[17] poskytuje ke každé webové stránce mimo jiné informace o počtu nalezených sledovacích prvků. Tyto informace jsou zobrazeny v okně doplňku, jak je možné vidět na obrázku 3.10. Vedle nalezených externě načítaných dat je zde klasifikace detekovaných prvků do skupin, příkladem takových skupin jsou externě načítané prvky typu reklama, analytické a monitorovací nástroje, prvky sociálních sítí a jiné kategorie. Doplněk nabízí možnost kliknutím na určitou položku v seznamu v pravé části doplňku zobrazit podrobnosti o nalezeném prvku včetně URL externě načítaného prvku.



Obrázek 3.10: Uživatelské rozhraní doplňku Ghostery dostupné po kliknutí na ikonu doplňku v pravém horním rohu prohlížeče.

Doplněk Ghostery dále nabízí možnost všechny prvky nalezené na webové stránce povolit nebo zablokovat. Nabízí se zde možnost každý konkrétní nalezený prvek zablokovat na konkrétní webové stránce, na všech webových stránkách nebo jej na aktuální webové stránce povolit. Dále je zde možnost kompletně povolit nebo zablokovat celou kategorii s nalezenými prvky (kliknutím na políčko na pravé straně hlavičky každé kategorie), nebo kompletně povolit nebo zablokovat všechny nalezené prvky ve všech uvedených kategoriích na aktuální webové stránce (prostřednictvím tlačítek „Trust Site“ a „Restrict Site“ v levém dolním rohu okna doplňku Ghostery, znázorněno na obrázku 3.10).

Doplněk dále nabízí dodatečnou funkcionalitu jako možnost dočasně pozastavit fungování doplňku (a tím i blokování nalezených prvků), a to na dobu 30 minut, 1 hodinu, nebo 24 hodin.

[17] V rámci testování byl použit doplněk Ghostery ve verzi 8.1.2

Při testování na skutečných webových stránkách se ukázalo, doplněk Ghostery dokáže na webové stránce detekovat celou řadu externě načítaných dat a dodává k nim i další informace jako název prvku, podrobnější informace o prvku a URL adresu externě načítaného prvku. Doplněk dokáže detekované prvky také klasifikovat na kategorie, příklady takových kategorií jsou externě načítané prvky typu reklama, analytické nástroje, prvky sociálních sítí a jiné kategorie. K zablokování těchto detekovaných prvků však nedochází automaticky, zablokování musí provádět uživatel sám prostřednictvím okna doplňku Ghostery. Tento fakt velice znesnadňuje procházení internetu a také to vyžaduje jistou znalost ze strany uživatele. Uživatel musí být schopný podle názvu prvku rozhodnout, zda jej chce zablokovat, nebo nikoli. Příkladem názvů externě načítaných prvků je například *OnThe.io*[18], *BBelements*[19], *Imedia*[20] a *DoubleClick*[21]. Názvy těchto prvků většinou reflektují název služby, která je vytvořila, většina těchto názvů však není příliš známá a běžný uživatel může mít problém vyznat se v takto označených položkách a bude to od něj vyžadovat určité znalosti problematiky, což znesnadňuje používání a správnou konfiguraci doplňku. Nesprávná konfigurace doplňku poté může mít za následek, že některé sledovací prvky nebudou zablokovány a chování uživatele na webové stránce bude nadále monitorováno.

Uživatelské rozhraní doplňku Ghostery je navíc plné celé řady funkcí a používání tohoto doplňku není tak pohodlné a intuitivní, jako například používání doplňků AdBlock nebo Javascript Control.

3.7 Porovnání funkcionality doplňků

V rámci testováni byly zkoumány doplňky Javascript Control, NoScript, AdBlock, Adblock Plus, Privacy Badger a Ghostery. První z řady testovaných doplňků, Javascript Control, funguje na principu kompletního zablokování technologie JavaScript na webové stránce. Další z testovaných doplňků používají přístup detekce externě načítaných prvků a následné klasifikace na typ prvku a snaží se určit míru jeho závadnosti, popřípadě zablokovat vybrané prvky.

Z testování v praxi vyplývá, že princip kompletního zablokování technologie JavaScript (technika využívaná doplňkem Javascript Control, ten je popsán v sekci 3.1) spolehlivě zablokuje všechny sledovací mechanismy a monitorovací mechanismy, které využívají technologii JavaScript k zajištění svého provozu. Na druhou stranu technologie JavaScript je v dnešní době nedílnou součástí většiny webových stránek a jejím kompletním zablokováním často dochází ke znemožnění určité funkcionality navštívené webové stránky. Tento fakt uživateli znemožní funkcionalitu webové stránky postavené na technologii JavaScript využívat, nebo bude nucen doplněk Javascript Control pro určité webové stránky deaktivovat. Deaktivací blokování JavaScriptu dohází k situaci, kdy je JavaScript plně funkční a uživatel již není na takové webové stránce chráněný.

[18]Služba monitorující a analyzující chování uživatelů na webové stránce, domovská stránka projektu zde - https://t.onthe.io/media
[19]Reklamní služba provozovaná společností Internet BillBoard a.s., dostupná zde - http://eu.bbelements.com/
[20]Cílená behaviorální reklama; domovská stránka projektu zde - http://www.imedia.cz/
[21]Reklamní služba společnosti Google, domovská stránka projektu zde - https://www.doubleclickbygoogle.com/

Princip detekce a klasifikace externě načítaných dat využívaný testovanými doplňky NoScript, AdBlock, Adblock Plus, Privacy Badger a Ghostery nabízí jemnější posuzování a následné blokování jednotlivých částí JavaScriptového kódu, než je tomu u přístupu kompletní blokace JavaScriptu na aktuální webové stránce využívaný testovaným doplňkem Javascript Control. Z testování však také vyšlo najevo, že tyto testované doplňky mají často problém s rozhodováním, které externí zdroje je bezpečné a nezbytné zablokovat a které naopak nikoli. V rámci testování bylo objeveno, že doplňky AdBlock (blíže popsán včetně nalezených nedostatků sekci 3.3) a Privacy Badger (blíže popsán včetně nalezených nedostatků sekci 3.5) na řadě webových služeb nezablokují například nalezený sledovací prvek nástroje Smartlook, nebo obdobné monitorovací a analytické služby.

V případě nástroje Ghostery dochází k situaci, kdy je dokonce jen na uživateli, které dekované prvky má doplněk blokovat. Právě použití nástroje Ghostery pak vyžaduje určitou míru spolupráce od uživatele, což znesnadňuje pohodlné procházení internetu, také to vyžaduje od uživatele jistou míru znalostí a schopností detekovaný prvek správně posoudit (bližší popis doplňku Ghostery včetně podrobností o náročnosti správně detekovat sledovací prvky je k nalezení v sekci 3.6).

Občas se stává, že doplněk zablokoval externí zdroj, který byl nutný pro provoz a funkcionalitu navštívené webové stránky. V sekci 3.2 je popsána situace, při které doplněk NoScript zablokoval JavaScriptem načítaný kód potřebný pro provoz webové stránky, navštívená webová stránka byla celá bíla a neobsahovala žádný obsah. Po bližším prozkoumání vyšlo najevo, že doplněk NoScript vyhodnotil jako závadný obsah načítaný ze stejné domény jako ta, na které byla hostována navštívená webová stránka. Povolením tohoto prvku v nastavení doplňku NoScript došlo k načtení potřebných dat a stránka poté byla opět funkční. Bylo však nutné správně určit a ručně povolit prvek, který je pro funkcionalitu stránky potřebný.

Při zkoumání doplňků bylo také objeveno, že doplněk Adblock Plus (detailně popsán v sekci 3.4) využívá tzv. bílou listinu, na které jsou vedeny webové stránky, které mají za poplatek zaručeno, že na jejich doméně nebude doplněk Adblock Plus blokovat externě načítané zdroje. Uživatel pak na těchto webových stránkách není chráněný.

Vedle faktu, že některé doplňky nedokáží správně detekovat, který nalezený prvek zablokovat a který naopak povolit, bylo v rámci testování objeveno, že často také dochází k situacím, kdy na jedné webové stránce každý z testovaných doplňků nalezne různé sledovací prvky. Dá se tedy usuzovat, že doplňky v řadě situacích nedetekují všechny sledovací a jiné nežádoucí prvky, a tedy neochrání soukromí uživatele doplňku. Takový uživatel může být sledován i v případě, že používá nějaký doplněk pro zajištění anonymity.

Kapitola 4

Technologie WebExtensions

Cílem technologie *WebExtensions*[1] je vysoká kompatibilita napříč řadou současných webových prohlížečů, důraz se klade také na bezpečnost a rychlost doplňků implementovaných za použití této technologie. Technologie WebExtenions dále také předpokládá kompatibilitu s budoucími verzemi podporovaných webových prohlížečů.

V této kapitole je popsána celá řada detailních informací týkajících se technologie WebExtensions. Tato kapitola se zabývá podporou technologie WebExtensions napříč současnými webový prohlížeči. Dále jsou zde popsány možnosti uživatelského rozhraní doplňků webových prohlížečů vytvořených v této technologii a také nevýhody implementace doplňku za pomoci technologie WebExtensions ve srovnání s předchůdci této technologie. Dále se tato kapitola zabývá strukturou doplňku implementovaného v technologii WebExtensions a je zde detailně popsán soubor manifest.json jakožto hlavní vstupní bod této technologie. Následně je uveden popis uložiště, které používá technologie WebExtensins k ukládání řady informací, jako například konfigurace samotného doplňku. Poté je v této kapitole popsána stránka s nastavením, kterou je možné ke každému doplňku v technologii WebExtensions implementovat. V závěru kapitoly je uveden popis rozhraní webRequest API.

4.1 Podpora WebExtensions v současných prohlížečích

Technologie WebExtensions je podporována v současných verzích následujících webových prohlížečů – Google Chrome, Microsoft Edge, Mozilla Firefox, Mozilla Firefox pro Android a Opera. Technologie WebExtensions předpokládá také kompatibilitu s budoucími verzemi těchto webových prohlížečů.

I přes skutečnost, že výše zmíněné prohlížeče podporují technologii WebExtensions, kompatibilita doplňků vytvořených v této technologii mezi jednotlivými prohlížeči není vždy úplná. Celá řada funkcí technologie WebExtensions je kompatibilní se všemi podporovanými webovými prohlížeči, vyskytují se však případy, kdy určité funkce podporovány nejsou[2].

[1] Technologie WebExtensions je rozhraní pro tvorbu doplňků webových prohlížečů a je postavena na webových technologiích HTML, JavaScript a CSS.

[2] Bližší informace o nekompatibilitě s prohlížečem Google Chrome - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Chrome_incompatibilities a bližší informace o kompatibilitě JS podpory pro jednotlivé volání - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Browser_support_for_JavaScript_APIs

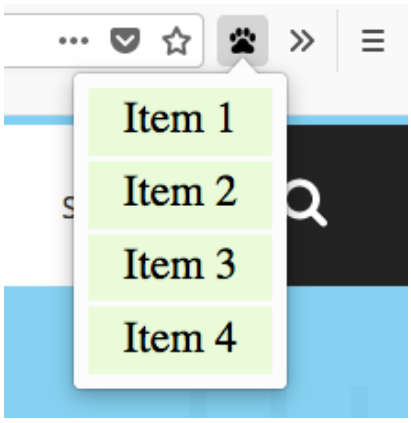
Případy nekompatibility se však týkají pouze menší množiny funkcí technologie WebExtensions a tyto funkce nepatří mezi ty nejvýznamnější. Dá se tedy usuzovat, že u většiny doplňků implementovaných za použití technologie WebExtensions problém s kompatibilitou nenastane. Většinou jsou také dostupné alternativní způsoby, jak případnou nekompatibilitu řešit. V případě přenosu (*tzv. portování*[3]) doplňku vyvinutém v jednom konkrétním prohlížeči na jiný webový prohlížeč je dostupná celá řada návodů a dokumentací, jak daný proces vykonat[4].

4.2 Uživatelské rozhraní doplňku ve WebExtensions

Technologie WebExtensions definuje, jaké prvky uživatelského rozhraní lze při tvorbě doplňku využívat. Vzhledem k důrazu na bezpečnost si tvůrce doplňku implementovaného při použití technologie WebExtensions nemůže definovat jakékoli uživatelské rozhraní, ani není technologií WebExtensions povoleno zasahovat do rozhraní samotného webového prohlížeče. Právě tak tomu bylo například u technologie XUL[5], která je považována za předchůdce technologie WebExtensions a která umožňovala vytvářet programátorem definované prvky uživatelského rozhraní včetně možnosti zasahovat do rozhraní samotného prohlížeče.

Tvůrce doplňku implementované za použití technologie WebExtensions musí využít předem připravené postupy a prvky pro definici uživatelského rozhraní. Rozhraní takového rozšíření poté tvoří celá řada předdefinovaných prvků, jako například tlačítka, tzv. *popup okna*, notifikace a další. Na obrázku 4.1 se nachází ukázka toolbar tlačítka jako příklad uživatelského rozhraní technologie WebExtensions, po jehož rozkliknutí se zobrazí popup se 4 položkami.

Dále je také možné přidávat vlastní části menu do kontextové nabídky, definovat boční lišty prohlížeče a vlastní stránky s nastavením (detailním popisem stránky s nastavením se zabývá sekce 4.7). Mimo to mohou doplňky přidávat i vlastní panely do *tzv. Developer tools*[6].



Obrázek 4.1: Ukázka toolbar tlačítka, po jehož rozkliknutí se zobrazí popup okno se 4 položkami.

[3]Bližší popis, co je portování – https://en.wikipedia.org/wiki/Porting

[4]Portování WebExtensions doplňku z Google Chromu do prohlížeče Mozilla Firefox - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Porting_a_Google_Chrome_extension

[5]Podrobnější informace o technologii XUL – https://en.wikipedia.org/wiki/XUL

[6]Developer tools prohlížeče Mozilla Firefox – https://developer.mozilla.org/son/docs/Tools

4.3 Nevýhody a omezení technologie WebExtensions

WebExtensions je technologie, jejíž jednou z hlavních priorit je bezpečnost. Z toho vyplývá celá řada omezení této technologie. Vývojáři se musí smířit s omezeným zásahem do uživatelského rozhraní implementovaného doplňku a do prohlížeče samotného. Pokud srovnáme technologii WebExtensions a předchůdce této technologie používané v prohlížeči Mozilla Firefox, technologii XUL, nalezneme zásadní odlišnosti. Technologie XUL umožňovala velmi rozsáhlé možnosti zásahu a redefinování uživatelského rozhraní, například možnost takřka neomezeného zásahu do prohlížeče včetně například redefinování vzhledu celého prohlížeče. Technologie WebExtensions má z důvodu zaměření na bezpečnost přesně definované, co implementovaný doplněk může a co nemůže provádět, programátor tedy nemůže redefinovat vzhled prohlížeče ani všech prvků uživatelského rozhraní libovolným způsobem, tak jako u technologie XUL. Musí k tomu využít předem definované prvky uživatelského rozhraní.

V technologie WebExtensions také nemůže samotný kód doplňku zasahovat do kódu stránky napřímo, tato skutečnost je dána odděleností procesů. Zobrazování obsahu panelů a tedy i běh doplňku implementovaného v technologii WebExtensions běží v jednom procesu, zatímco uživatelské rozhraní webového prohlížeče včetně webové stránky běží v procesu druhém. Zásah do stránky je však možný provést za pomoci JavaScriptového kódu, který bude vložen přímo do aktuální webové stránky. Tento JavaScriptový kód může dále s WebExtensions doplňkem komunikovat, a to například prostřednictvím systému zasílání zpráv. Tím způsobem může doplněk ovládat webovou stránku.

V některých prohlížečích byla podpora této technologie zavedena teprve nedávno (například u webového prohlížeče Mozilla Firefox došlo k zavedení podpory technologie WebExtensions od verze 48[7] a to 2.8.2017[8]) a některé funkce a části technologie WebExtensions nebyly v době zveřejnění verze 48 tohoto prohlížeče doimplementované a také některé funkce a rozhraní vykazovaly potřebu vylepšení. Jako příklad lze uvést rozhraní *requestBody*[9], které po vydání prohlížeče Mozilla Firefox ve verzi 48 nebylo podporováno vůbec a v pozdějších verzích bylo doimplementováno[10]. Jako příklad vylepšení, které bylo provedeno v pozdějších verzích, lze uvést drobné úpravy uživatelského rozhraní[11].

[7] O zavedení podpory technologie WebExtensions do prohlížeče Mozilla Firefox verze 48 je informováno - https://blog.mozilla.org/addons/2016/04/29/webextensions-in-firefox-48/ ,verze 47 obsahuje beta verzi https://blog.mozilla.org/addons/2016/03/11/webextensions-in-firefox-47/ a verze 46 alpha verzi https://blog.mozilla.org/addons/2016/02/02/webextensions-in-firefox-46/

[8] Datum zavedení zmíněné zde - https://wiki.mozilla.org/Release_Management/Calendar

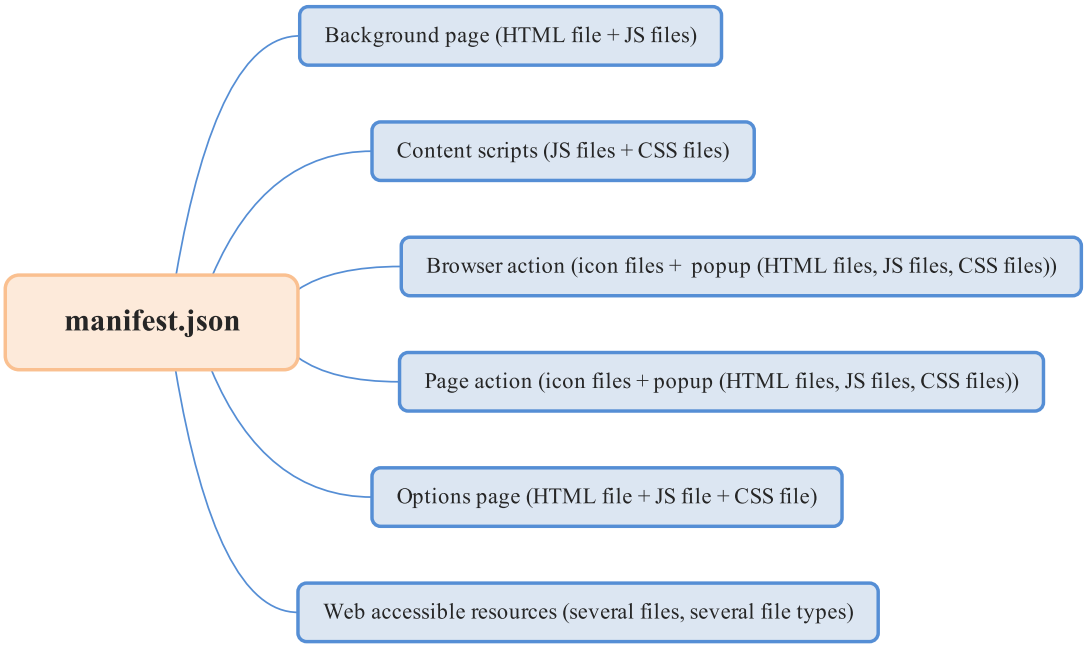
[9] Rozhraní je několikrát zmíněn v rámci dokumentace - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest/onBeforeRequest

[10] Zmíněno zde - https://blog.mozilla.org/addons/2016/04/29/webextensions-in-firefox-48/

[11] Vylepšení rozhraní zmíněno - https://blog.mozilla.org/addons/2016/09/29/webextensions-in-firefox-51/

4.4 Struktura doplňku v technologii WebExtensions

Struktura samotného doplňku je dána především technologií WebExtensions, která mimo jiné stanovuje pravidla pro rozložení návrhu celého doplňku do souborů a pro propojení souborů navzájem. Na obrázku 4.2 je tato struktura zachycena. Hlavním vstupním bodem každého doplňku implementovaného s použitím technologie WebExtensions je soubor *manifest.json*[12], který je podrobněji popsán v sekci 4.5.



Obrázek 4.2: Obrázek zachycuje strukturu doplňku implementovaného při použití technologie WebExtensions. Struktura zachycuje soubor manifest.json a celou řadu dalších souborů.

Vedle souboru manifest.json může být doplněk tvořen celou řadou dalších souborů. Tyto soubory jsou poté umístěny v kořenovém adresáři doplňku společně se souborem manifest.json, nebo se nachází v podadresářích kořenového adresáře. Jakékoli další soubory poté musí být připojeny prostřednictvím záznamu v souboru manifest.json, nebo prostřednictvím jiného souboru, které je sám buď napřímo nebo tranzitivně připojen k souboru manifest.json. Příkladem napřímo připojených JavaScriptových souborů jsou například *tzv. background scripty*[13], jejich připojení k souboru manifest.json je demonstrováno v kódu 4.3.

Příkladem napřímo připojeného HTML souboru je například soubor implementující stránku s nastavením doplňku, příklad záznamu souboru manifest.json definující cestu k souboru se stránkou nastavení se nachází v kódu 4.6.

[12]Kompletní popis souboru manifest.json v oficiání dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json

[13]Background scripty popsané v dokumentaci - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/background

4.5 Soubor manifest.json

Doplněk vytvořený prostřednictvím technologie WebExtesions má pevně danou strukturu souborů, kterou je potřeba dodržet při tvorbě každého doplňku v této technologii. Hlavním vstupním bodem každého doplňku je soubor *manifest.json*[14]. Tento soubor obsahuje informace o dalších souborech tvořících kód doplňku, definuje cestu k umístění jednotlivých JavaScriptových souborů s vlastním kódem, cesty k souborům s CSS styly, cesty k HTML souborům a celou řadu dalších informací, metainformací a nastavení.

Soubor manifest.json je ve formátu JSON, tedy se jedná o *tzv. JavaScript Object Notation* formát[15]. V této sekci je popsána řada dvojic klíč-hodnota, jejichž přítomnost v souboru manifest.json reprezentuje určitý typ informace, nebo nastavení, které má následně vliv na funkcionalitu implementovaného doplňku.

Soubor manifest.json musí povinně obsahovat implementované minimálně 3 základní klíče, konkrétně se jedná o klíče „manifest_version“, „name“ a „version“. V kódu 4.1 je zachycena minimální implementace souboru manifest.json, která je interpretována dle technologie WebExtensions bez chyb.

```
{
  "manifest_version": 2,
  "name": "Extension name",
  "version": "1.0"
}
```

Kód 4.1: Příklad minimální validní implementace souboru manifest.json, která obsahuje všechny povinné klíče, a tedy „manifest_version“, „name“ a „version“.

V případě pokusu o instalaci doplňku do webového prohlížeče, jehož soubor manifest.json neimplementuje správně tyto 3 základní klíče, dojde k zobrazení chyby znázorněné na obrázku 4.3 a prohlížeč odmítne doplněk nainstalovat. Chyba však může nastat i v případě správné implementace těchto 3 klíčů v souboru manifest.json, a to v případě syntaktické nebo jiné chyby v souboru manifest.json. Tento soubor pak nebude možné správně interpretovat, což povede opět k chybě a neúspěchu při instalaci.

Obrázek 4.3: Obrázek zachycuje chybu, která se zobrazí při pokusu přidat do prohlížeče doplněk, jehož soubor manifest.json neimplementuje správně základní tři klíče („manifest_version“, „name“ a „version“), nebo obsahuje jinou, například syntaktickou chybu[16].

[14]Dokumentace k souboru manifest.json a implementovaným klíčům je dostupná na adrese https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json

[15]Detailní informace o JSON jsou dostupné na stránce https://www.json.org/

[16]Chybová zpráva poskytnutá prohlížečem Mozilla Firefox. Chybové zprávy v jiných prohlížečích se mohou vzhledově i textem chybového popisku lišit.

Vedle výše zmíněných povinných klíčů může soubor manifest.json obsahovat celou řadu dalších volitelných klíčů. Některé z nich stanovují hodnoty doplňkových informací a metainformací implementovaného doplňku, jiné připojují k doplňku další soubory a stanovují jejich význam a použití. Kód 4.2 uvádí příklad implementace dvou vybraných nepovinných klíčů „descriptiton“[17] a „hamepage_url“[18], které stanovují popisek daného rozšíření a adresu domovské stránky.

```
"description": "Description of the extension",
"homepage_url": "https://www.homepage.com"
```

Kód 4.2: Příklad implementace nepovinných klíčů „descriptiton“ a „hamepage_url“.

Technologie WebExtensions umožňuje prostřednictvím záznamu v souboru manifest.json definovat spuštění *tzv. background scriptů*. Jedná se o JavaScriptový kód, který je vykonáván na pozadí v rámci načtené webové stránky. V kódu 4.3 je zachycena část souboru manifest.json, který implementuje klíč „background“[19]. Ten zde konkrétně definuje cestu k souboru background.js, který obsahuje JavaScriptový kód, který bude vykonáván.

```
"background": {
  "scripts": ["background.js"]
}
```

Kód 4.3: Příklad implementace klíče „background“ s cestou k souboru background.js, který bude spuštěn jako *tzv. background script*.

V souboru manifest.json je dále možnost nastavit klíč „permissions“[20], jehož hodnoty stanovují, které úkony a operace je doplněk oprávněn provádět. V kódu 4.4 je zachyceno nastavení hodnoty „storage“, která reprezentuje oprávnění pracovat s uložištěm technologie WebExtensions. Do uložiště je možné data zapisovat a opětovně je z něj načítat. Uložiště technologie WebExtensions a práce s ním je podrobněji popsána v sekci 4.6.

```
"permissions": [
  "storage"
]
```

Kód 4.4: Příklad implementace klíče „permissions“, které v tomto konkrétním případě dává doplňku oprávnění pracovat s uložištěm technologie WebExtensons.

[17] https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/description
[18] https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/homepage_url
[19] https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/background
[20] https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/permissions

Vedle oprávnění pracovat s uložištěm technologie WebExtensions je možné prostřednictvím klíče „permissions“ definovat desítky dalších oprávnění[21]. V kódu 4.5 je uveden příklad zápisu definice více hodnot, tedy udělení více typů oprávnění, konkrétně se jedná o oprávnění „storage“ (oprávnění pracovat s uložištěm) a oprávnění „webRequest“ (oprávnění pracovat s rozhraním webRequest, které je blíže popsáno v sekci 4.8).

```
"permissions": [
  "storage",
  "webRequest"
]
```

Kód 4.5: Příklad implementace klíče „permissions“ s více hodnotami, konkrétně se jedná o hodnotu „storage“ a o hodnotu „webRequest“.

V souboru manifest.json je dále možné definovat klíč „options_ui“[22], který definuje objekt ve formátu JSON jako svoji hodnotu. Datové složky tohoto objektu určují cestu k HTML souboru s implementací stránky nastavení a dále se zde může nacházet další dodatečná konfigurace týkající se stránky s nastavením doplňku. V kódu 4.6 je uveden příklad nastavení klíče „options_ui“ a jeho hodnoty s cestou k souboru options.html, který obsahuje definovanou stránku s nastavením a definice dodatečné vlastnosti „browser style“ (dodatečná vlastnost stanovuje, zda má být vzhled implementované stránky konzistentní se vzhledem prohlížeče, ve kterém doplněk aktuálně běží).

```
  "options_ui": {
    "page": "options.html",
    "browser_style": true
  }
```

Kód 4.6: Příklad implementace klíče „options_ui“ v rámci souboru manifest.json.

V souboru manifest.json je možné definovat klíč s názvem „content_scripts“[23]. Tento klíč se odkazuje na soubory s JavaScriptovým kódem, které jsou spouštěné na načítané webové stránce. V kódu 4.7 zachycena implementace klíče „content_scripts“ i s jeho hodnotou. Hodnota je zde tvořena JSON objektem a obsahuje odkaz na soubor nebo soubory s JavaScriptovým kódem, které budou spuštěny v definovaném okamžiku a další doplňující parametry.

[21]Popis klíče „permissions“ včetně všech hodnot - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/permissions

[22]Stránka dokumentace s popisem klíče „options_ui“ - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/options_ui

[23]Dokumentace ke klíči „content_scripts“ včetně popisu jednotlivých hodnot - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/content_scripts

```
"content_scripts": [
   {
      "matches": ["<all_urls>"],
      "js": ["document_start.js"],
      "run_at": "document_start"
   }
]
```

Kód 4.7: Příklad implementace klíče s názvem „content_script“ včetně jeho hodnoty v souboru manifest.json.

Hodnota klíče „run_at“ uvedeného v kódu 4.7 nám zde definuje, ve který konkrétní okamžik bude odkazovaný JavaScriptový soubor spuštěn. Technologie WebExtensions zde nabízí 3 možnosti okamžiků, kdy bude daný JavaScriptový kód vykonáván – „document_start“, „document_end“ a „document_idle“. Hodnota „document_start“ klíče „run_at“ definuje, že se příslušný script bude spouštět ještě před tím, než dojde k vyhodnocování zdrojového kódu načtené webové stránky a k tvorbě stromové struktury DOM. Hodnota „document_end“ klíče „run_at“ stanovuje pro spouštění JavaScriptového kódu v takový moment, kdy stromová struktura DOM je již plně vytvořená, ale není zde zaručené, že došlo k načtení všech externích zdrojů typu skripty a obrázky. Ty se mohou stále načítat. Hodnota „document_idle“ klíče „run_at“ definuje, že daný skript se bude spouštět v momentě, kdy došlo ke kompletnímu načtení a zpracování celé webové stránky. Nejen stromová struktura DOM je již plně vytvořená, ale všechny externí zdroje jako skripty nebo obrázky jsou také plně načtené. Výchozí hodnotou klíče „run_at“ je "document_idle", daný stav je možné předefinovat a to právě nastavení klíče "run_at" na jinou hodnotu, jak je to uvedeno v kódu 4.7. Tyto stavy nastavované v technologii WebExtensions přímo odpovídají JavaScriptovým stavům dostupným v *document.readyState*.

4.6 Uložiště technologie WebExtensions

Technologie WebExtensions nabízí možnost využít uložiště[24] pro ukládání a načítání dat. Toto uložiště se mimo jiné využívá také pro ukládání nastavení konkrétního doplňku implementovaného za použití technologie WebExtensions. Čtení i zápis do uložiště se děje asynchronně. V kódu 4.8 je uvedena ukázka zachycující práci s uložištěm technologie WebExtensions, konkrétně se jedná o ukládání dat do uložiště.

```
browser.storage.sync.set({
   data_name: data_value
});
```

Kód 4.8: Příklad uložení hodnoty „data_value“ pod názvem klíče „data_name“ do uložiště technologie Webextensions.

[24]Uložiště technologie WebExtensions je detailně popsáno v dokumentaci - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/storage

Vedle ukládání dat do uložiště je dostupná také možnost načíst data z uložiště. K načtení dat dochází pro konkrétní položku, která je označena identifikátorem klíče. V kódu 4.9 je zachycena ukázka kódu, který načítá hodnotu klíče „data_name“ z uložiště.

```
var itemFromStorage = browser.storage.sync.get('data_name');
itemFromStorage.then((res) => {
   document.querySelector("#data_from_storage").value = res.data_name;
});
```

Kód 4.9: Příklad načtení hodnoty uložené v uložišti technologie WebExtensions.

Datová složka klíče je reprezentována položkou ve formátu JSON, nemusí se tedy jednat pouze o jednu řetězcovou položku, ale o celou strukturu obsahující data. Načítání dat z uložiště je jako zapisování do uložiště asynchronní.

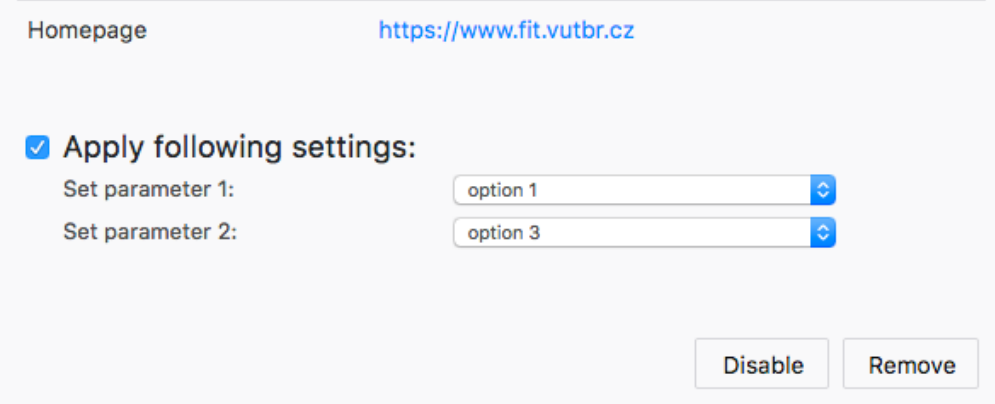
4.7 Stránka s nastavením doplňku

Doplněk vytvořený v technologii WebExtensions může implementovat stránku s nastavením[25]. Stránka s nastavením je implementována za pomoci technologií HTML, CSS a JavaScript. Vstupním bodem je soubor HTML, na který vede odkaz ze souboru manifest.json ze záznamu „options_ui“ jak je zachyceno v kódu 4.6. Tento HTML dokument může dále načítat další CSS a JavaSctiptové soubory ze své hlavičky a implementovat základní strukturu a obsah stránky s nastavením doplňku. Na obrázku 4.4 je zachycena stránka s nastavením, jejíž HTML soubor neimplementuje žádné dodatečné prvky sloužící k interakci s uživatelem. Nabízí pouze základní možnosti typu zvolit si nastavení automatických aktualizací, navštívit domovskou stránku tvůrce doplňku, deaktivovat funkčnost doplňku nebo doplněk kompletně odebrat z prohlížeče. Dále se zde nachází název samotného doplňku (definovaný v souboru manifest.json jak je uvedeno v kódu 4.1) a jeho popisek (definice uvedena v kódu 4.2).

Obrázek 4.4: Stránka s nastavením obsahující pouze základní povinné položky.

[25]Dokumentace k implementaci stránky s nastavením – https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Implement_a_settings_page

Vedle uvedených základních možností může stránka s nastavením implementovat také další nepovinné položky sloužící uživateli k detailnější konfiguraci doplňku nebo mu poskytnout a zobrazit určitá data. Příklad takových položek je zachycen na obrázku 4.5 (položky jsou modelovány pomocí HTML, nejčastěji se používají formulářové prvky[26] jako zaškrtávací políčka, tlačítka, políčko pro vyplnění textu, popisky a další).



Obrázek 4.5: Stránka s nastavením obsahující vedle povinných položek také implementované dodatečné volby pro konfiguraci doplňku.

HTML soubor implementující položky pro interakci s uživatelem může používat veškeré JavaScriptové volání technologie WebExtensions, ke kterém má oprávnění. Tento JavaScriptový kód bude spuštěn v odděleném adresním prostoru, než v jakém jsou spouštěny background scripty. Jakákoli komunikace poté může probíhat přes uložiště technologie WebExtension popsaném v sekci 4.6. Další možnost komunikace je prostřednictvím vzájemného poskytnutí reference na objekt window[27] zavoláním funkce extension.getBackgroundPage(). Komunikace může také probíhat prostřednictvím zasílání zpráv, k tomuto účelu slouží funkce runtime.sendMessage(), runtime.onMessage a runtime.connect()[28].

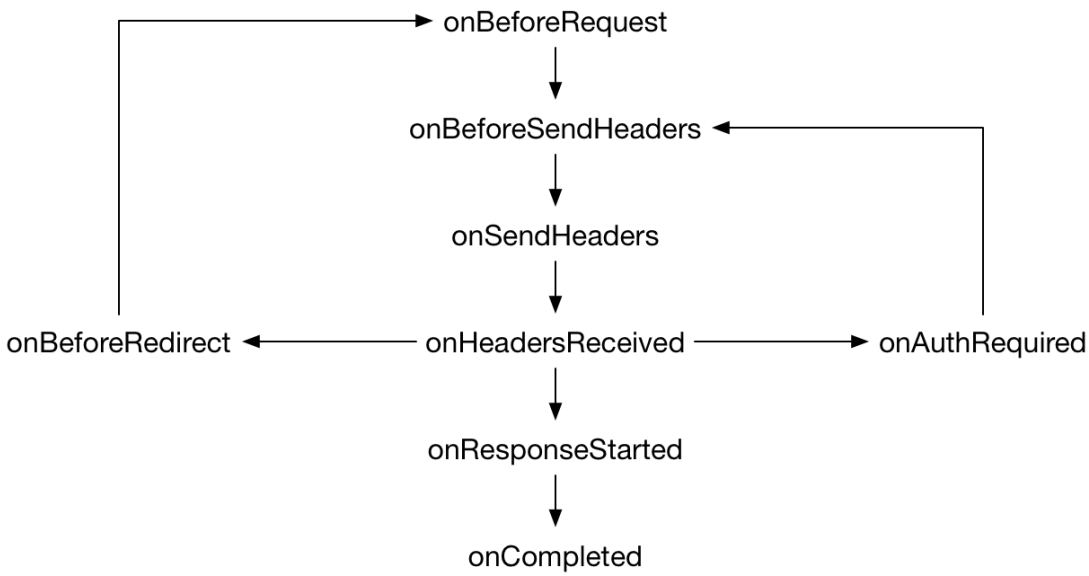
[26]Formulářové prvky HTML - https://www.w3schools.com/html/html_form_elements.asp

[27]Objekt window technologie JavaScript je detailně popsán na webové stránce https://www.w3schools.com/js/js_window.asp

[28]Možnosti vzájemné komunikace jsou podrobně popsány v dokumentaci - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/options_ui

4.8 Rozhraní webRequest API

Rozhraní webRequest API je rozhraní technologie WebExtensions, které umožňuje přidat tzv. *event listenery*[29] pro různé části zpracování HTTP požadavku. Každý konkrétní implementovaný event listener je spojen s kódem, který je spouštěn v době výskytu určité konkrétní události. V rámci zpracování HTTP požadavku jsou jednotlivé události vyvolávány postupně, a to v pořadí, které je znázorněno na obrázku 4.6. Kód event listeneru má možnost změnit HTTP požadavek, přesměrovat jej, vložit autentizační údaje, nebo jej zrušit. Možnosti zasahovat do HTTP požadavku jsou však dostupné jen u některých vyvolaných událostí.



Obrázek 4.6: Schéma postupného spouštění jednotlivých událostí při zpracování HTTP požadavku rozhraním webRequest v technologii WebExtensions[30].

Kromě událostí zmíněných na obrázku 4.6 může nastat ještě další událost. Událost *onErrorOccurred*[31] může být spuštěna v kterýkoli okamžik v rámci průběhu HTTP požadavku, jedná se o událost, která je spuštěna v době výskytu chyby a její listener umožňuje na tuto situaci reagovat.

[29] Event listener je funkce, k jejíž invokaci dojde při výskytu určitého typu události, funkce poté obdrží podrobnější informace o situaci

[30] Schéma převzato z dokumentace k technologie WebExtensions společnosti Mozilla - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest

[31] Dokumentace k události onErrorOccurred - https://developer.mozilla.org/en-US/Add-ons/WebExtensions/API/webRequest/onErrorOccurred

Kapitola 5

Návrh implementovaného doplňku

V této kapitole se nachází návrh doplňku pro internetový prohlížeč zaměřený na ochranu soukromí, který je implementován v rámci této diplomové práce. Při návrhu je vycházeno především ze studia problematiky monitorovacích přístupů a mechanismů, dále také z výsledku studia a porovnání současných doplňků webových prohlížečů a také z poznatků implementačních technologií, především z informací o technologii WebExtensions (detailně popsána v kapitole 4), dále také z poznatků o technologiích HTML, JavaScript a CSS[1].

5.1 Návrh funkcionality implementovaného doplňku

Princip fungování doplňku je postaven na obalení a předefinování vybraných JavaScriptových objektů, funkcí a prototypů a to takovým způsobem, aby implementace původní konstrukce byla ukryta a zapouzdřena uvnitř našeho kódu a byla využívána pouze podle námi stanovených pravidel. Blíže je tento princip zapouzdření popsán v sekci 6.1. K obalení původních konstrukcí dochází v době před začátkem zpracování zdrojového kódu načítané webové stránky. Tímto principem je zajištěno, že námi obalený JavaScriptový kód neobalil nikdo před námi. Dále vlivem ukrytí a zapouzdření původní JavaScriptové konstrukce uvnitř našeho kódu se docílí toho, že žádný jiný kód kromě našeho nebude mít k původní implementaci přístup.

Funkcionalita doplňku je navržena následovně – obalovány budou původní implementace následujících JavaScriptových konstrukcí:

- **Objekt window.Date** – objekt window.Date[2] v sobě nese údaje o čase a datu. Tento objekt nabízí mimo jiné funkce getMilliseconds() a getTime() poskytující jemné časové údaje s přesností na milisekundy. Obalením tohoto objektu a přidáním dalšího pomocného kódu budou instance window.Date poskytovat časová data se sníženou přesnost vzhledem k originálním datům. Míra snížení přesnosti bude záležet na aktuálním nastavení doplňku uživatelem.

[1]Technologie HTML, CSS a JavaScript tato práce detailně nepopisuje, podrobnosti zde - https://www.w3schools.com/html/html_intro.asp, https://www.w3schools.com/js/js_intro.asp a https://www.w3schools.com/css/css_intro.asp

[2]Objekt window.Date je detailně popsaný v dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

- **Funkce window.performance.now()** – objekt window.performance[3] obsahuje funkci now()[4], která vrací data ve formátu DOMHighResTimeStamp[5]. Obalení funkce window.performance.now() zajistí po zavolání této funkce možnost upravovat návratová data ještě před tím, než jsou vrácena uživateli. Upravování zde bude probíhat formou snížení přesnosti poskytovaných dat (tedy zaokrouhlení vrácené hodnoty), snížení přesnosti bude probíhat dle nastavení doplňku uživatelem.

- **Funkce window.HTMLCanvasElement.prototype.getContext**() – HTML elementy typu Canvas se mohou na webové stránce vyskytovat vícekrát, také mohou dynamicky vznikat a zanikat. Tato skutečnost je brána na zřetel a při filtrování a blokování zápisu do HTML elementu typu Canvas je nutné pracovat s jeho prototypem. Díky prototypové dědičnosti jazyka JavaScript bude funkce obalena pouze jednou a to pro všechny HTML elementy typu Canvas. Při úpravě prototypu window.HTMLCanvasElement.prototype[6] bude obalena její funkce getContext()[7], která je volána pro získání kontextu pro kreslení do canvasu. Toto obalení a další manipulační kód zajistí kontrolu nad zápisem do HTML elementu typu Canvas dle aktuálního nastavení doplňku.

- **Funkce navigator.geolocation.getCurrentPosition()** – objekt navigator.geolocation obsahuje funkce getCurrentPosition()[8], která poskytuje objekt s GPS informacemi o poloze uživatele. Obalením této funkce zajistíme možnost upravovat poskytnutý objekt s GPS daty ještě před tím, než je objekt předán dále. V rámci obalení je možné za použití pomocného kódu snížit přesnost poskytovaných GPS dat dle aktuálního nastavení doplňku, nebo GPS data úplně anonymizovat (např. nastavením všech hodnot objektu s GPS daty na hodnotu 0).

- **Objekt window.XMLHttpRequest** – instance tohoto objektu slouží k odesílání tzv. XMLHttpRequest[9] dotazů. Tyto dotazy v sobě mohou obsahovat data, která jsou bez vědomí uživatelů sbírána a odesílána. Obalením tohoto objektu a přidáním dalšího pomocného kódu získáme možnost analyzovat, zobrazovat, povolovat a blokovat dotazy tohoto typu dle aktuálního nastavení implementovaného doplňku.

[3]Objekt window.performance je blíže popsaný v dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/docs/Web/API/Window/performance

[4]Funkce window.performance.now() je detailně popsána v dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/docs/Web/API/Performance/now

[5]Datový typ DOMHighResTimeStamp je detailněji popsán v dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/docs/Web/API/DOMHighResTimeStamp

[6]Blíže je tento prototyp popsán v dokumentaci - https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement

[7]Detailní popis funkce getContext() v dokumentaci - https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/getContext

[8]Detailní popis funkce getCurrentPosition() je dostupný v dokumentaci společnosti Mozilla - https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/getCurrentPosition

[9]Detailní popis objektu window.XMLHttpRequest - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Dostatečným snížením přesnosti časových dat, které jsou poskytovány objektem window.Date a funkcí window.performance.now() dojde k znemožnění identifikace uživatele podle časových značek [5, 7, 8, 9]. Dostatečným snížením přesnosti dat poskytovaných funkcí navigator.geolocation.getCurrentPosition() omezíme možnost získat polohu uživatele na základě JavaScriptem poskytovaných GPS dat. Blokováním zápisu dat do HTML elementu typu Canvas zamezíme tvorbě otisků grafické karty a tím i jednomu z řady způsobů, který se používá k identifikaci počítačových stanic a tím i uživatelů na internetu [4, 6]. Blokováním nežádoucích dotazů typu window.XMLHttpRequest zamezíme nežádoucímu odesílán dat (doplněk umožňuje kompletní blokování těchto dotazů, nebo zobrazení podrobností a dotázání se uživatele na akci – tedy jestli zakázat nebo povolit window.XMLHttpRequest dotaz).

5.2 Návrh implementace doplňku v rámci technologií WebExtensions, HTML, JS a CSS

Implementovaný doplněk je vytvořen za použití technologie WebExtensions a má tedy souborovou strukturu danou touto technologií (struktura daná technologií WebExtensions je zachycena na obrázku 4.2). Konkrétní implementace využívá následující soubory:

- **soubor manifest.json** – jedná se o hlavní vstupní bod implementovaného doplňku. Popis pravidel pro vytvoření validního souboru manifest.json je daný technologií WebExtensions a je popsán v sekci 4.5. Konkrétní návrh implementace souboru manifest.json pro doplněk vytvořený v rámci této diplomové práci se nachází v sekci 5.2.1.

- **soubor options.html** – soubor implementující stránku s nastavením doplňku. Tento soubor je ve formátu HTML a udává obsah stránky s nastavením, definuje tedy prvky tvořící tuto stránku jako zaškrtávací políčka, popisky, tlačítka, nebo například prvky pro výběr možnosti.

- **soubor options.js** – JavaScriptový soubor, na který vede reference ze souboru options.html. Tento soubor obsahuje JavaScriptový kód pro načítání konfigurace doplňku z uložiště, nastavování položek uživatelského rozhraní na příslušnou hodnotu dle dat získaných z uložiště a ukládání aktuálního nastavení do uložiště (po kliknutí na tlačítko „Uložit nastavení“)

- **soubor options.css** – soubor definující CSS styly pro vzhled a rozložení prvků na stránce s nastavením implementovaného doplňku

- **soubor document_start.js** – soubor obsahující JavaScriptový kód zajišťující obalení vybraných JavaScriptových komponent

- **soubor background.js** – soubor obsahující JavaScriptový kód, který bude spouštěn na pozadí

- **soubor README.md** – soubor obsahující popis doplňku

5.3 Návrh souboru manifest

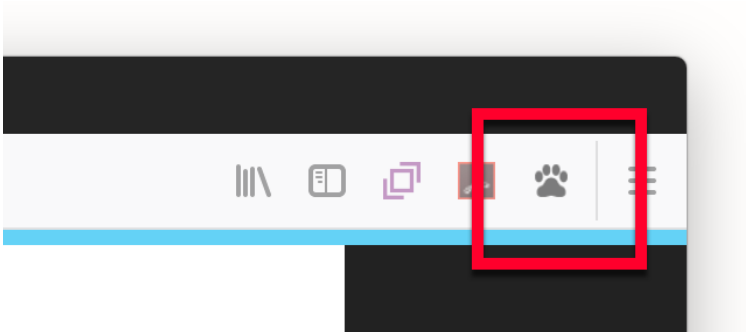
Soubor manifest jakožto vstupní bod doplňku je implementován podle pravidel daných technologií WebExtensions (detailní popis pravidel pro validní implementaci souboru manifest.json je popsán v sekci 4.5).

Soubor manifest.json námi navrhovaného doplňku tedy musí obsahovat implementované všechny 3 povinné klíče (podrobnosti implementace těchto 3 klíčů jsou popsány v sekci 4.5 a příklady uvedeny v kódu 4.1). Dále musí také obsahovat referenci na všechny ostatní soubory, se kterými doplněk pracuje (s výjimkou souboru README.md, který v sobě nese pouze popis daného doplňku). Tato reference musí být součástí definice hodnoty odpovídajícího klíče, aby při interpretaci souboru manifest.json byl zřejmý význam připojeného souboru. Některé soubory obsahující kód, který implementovaný doplněk bude využívat, a nemusí být připojeny k souboru manifest napřímo. Takovými soubory jsou například CSS soubory se styly k HTML dokumentům, nebo některé JavaScriptové soubory s pomocným kódem (například JavaScriptový soubor připojený k HTML souboru reprezentující stránku s nastavením).

Soubor manifest.json implementovaného doplňku definuje také celou řadu dalších nepovinných klíčů, například klíče „description“ a „homepage_url“ s dodatečnými informacemi.

5.4 Návrh uživatelského rozhraní

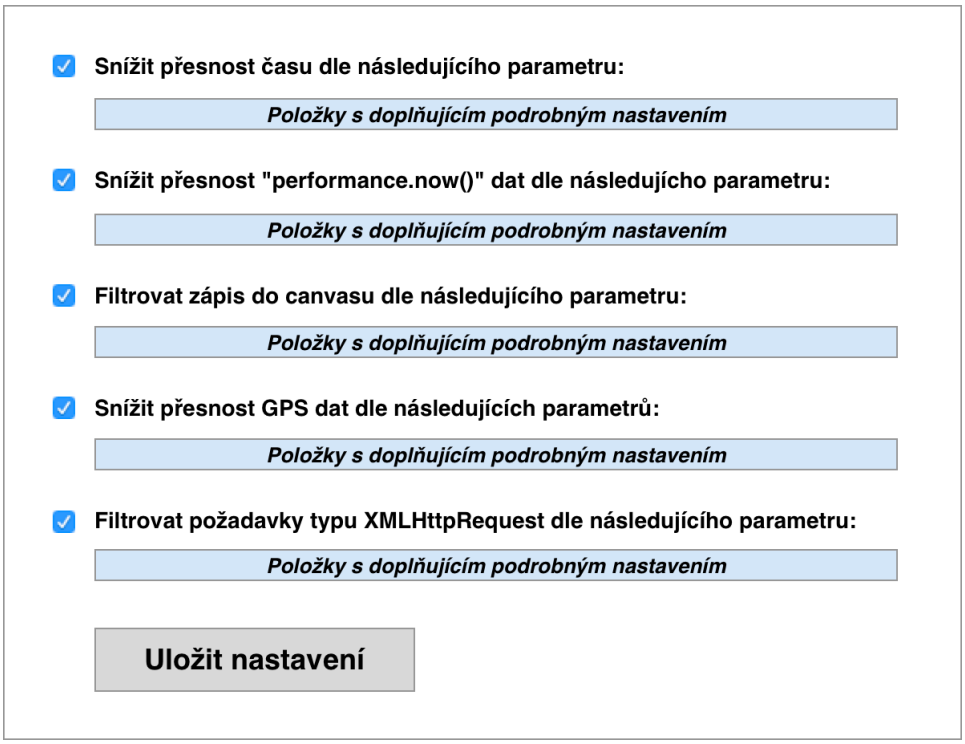
V rámci implementace doplňku je nutné poskytnout uživatelům způsob, jakým si budou moci doplněk konfigurovat. Dle návrhu doplněk poskytuje uživateli toolbar ikonu umožňující kliknutím přejít do stránky s nastavením. Na obrázku 5.1 je možné vidět ukázku právě takové ikony (obrázek takové ikony se může lišit v závislosti na konkrétní implementaci doplňku v technologii WebExtensions).



Obrázek 5.1: Obrázek zachycuje vzhled a umístění toolbar ikony která tvoří základní uživatelské rozhraní implementovaného doplňku[10].

[10]Obrázek zachycující toolbar ikonu převzat z https://developer.mozilla.org/en-US/Add-ons/WebExtensions/user_interface

Po kliknutí na tuto ikonu je uživatel přesunut na stránku s nastavením. Zkrácená verze návrhu stránky s nastavením doplňku je zachycen na obrázku 5.2 (plná verze je z důvodu rozsahu zachycena v příloze A na obrázku A.1). Tento návrh v sobě zachycuje 5 hlavních kategorií, přičemž každá z nich odpovídá jedné JavaScriptové konstrukci popsané v úvodu sekce 5.1. Každá z těchto kategorií má de návrhu jedno zaškrtávací políčko, které aktivuje funkcionality implementovaného doplňku pro danou sekci (dojde tedy k obalení příslušné JavaScriptové konstrukce dle nastavených parametrů v dané kategorii). V případě odškrtnutého políčka a tedy vypnuté funkcionality dané kategorie se bez ohledu na jakékoli jiné nedojde ani k redefinování a obalení příslušné JavaScriptové konstrukce (zůstane tedy nedotčena).



Obrázek 5.2: Návrh stránky s nastavením.

5.5 Návrh práce s uložištěm technologie WebExntensions

Implementovaný doplněk bude využívat k zajištění své funkcionality také uložiště technologie WebExtensions (detailně popsáno v sekci 4.6). Do uložiště bude zapisována konfigurace implementovaného doplňku, a to při kliknutí na tlačítko „Uložit nastavení“ na stránce s nastavením (návrh stránky a nastavením včetně ukládacího tlačítka je vyobrazen na obrázku 5.2). Dále bude z uložiště tato konfigurace doplňku také načítána, a to hned ve dvou situacích, při spouštění obalovacích scriptů, které přepisují původní JavaScriptové konstrukce (z důvodu konfigurace obalovacích scriptů) a při zobrazení stránky s nastavením (zde je načítání nutné z důvodu správného nastavení formulářových HTML prvků tvořící rozhraní stránky s nastavením).

5.6 Návrh stránka s nastavením

Implementovaný doplněk nabízí stránku s nastavením, přes kterou je možné tento doplněk konfigurovat. Tato stránka nabízí uživateli možnost kompletně aktivovat nebo deaktivovat celý doplněk, dále možnost kompletně zapnout, nebo vypnout funkcionalitu pro jednotlivá zapouzdření (přehled JavaScriptových konstrukcí, které jsou doplňkem zapouzdřovány a upravovány, je uvedeny v sekci 5.1). Dále je možné konfigurovat podrobnější nastavení chování zapouzdření pro jednotlivé JavaScriptové konstrukce. Seznam obalovaných konstrukcí včetně návrhu konfigurace parametrů obalování je dostupný v následujícím seznamu:

- **Objekt window.Date** – pro obalení tohoto objektu je v nastavení implementovaného doplňku dostupná možnost zaokrouhlení poskytovaného času v rámci instance tohoto objektu s přesností na desítky milisekund, stovky milisekund a celé sekundy. Jako výchozí možnost je zde nastaveno zaokrouhlování na stovky milisekund.

- **Funkce window.performance.now()** – pro obalení této funkce je dostupná konfigurace zaokrouhlení vrácené časové hodnoty na jednotky, na desítky, na stovky, nebo na tisíce. Výchozí hodnotou je zde zaokrouhlování na stovky, přesnost takto zaokrouhleného časového údaje řádově odpovídá přesnosti času poskytnutého objektem window.Date při aktivaci obalení s výchozí hodnotou zaokrouhlování na stovky milisekund.

- **Funkce window.HTMLCanvasElement.prototype.getContext**() – pro obalení této funkce je v nastavení dostupná možnost zakázat veškerý zápis do HTML elementů typu Canvas, nebo možnost dotázat se uživatele pro každý pokus zápisu do Cavasu.

- **Funkce navigator.geolocation.getCurrentPosition()** – pro obalení této funkce je v nastavení celá řada možností konfigurace. Nejdříve uživatel zvolí, jestli poskytované GPS údaje vynulovat, nebo pouze snížit jejich přesnost. Při volbě snížení přesnosti se zde nachází 7 dalších možností konfigurace pro stanovení míry snížení přesnosti poskytovaných časových dat. Konkrétně se jedná o snížení přesnosti zeměpisné šířky a zeměpisné délky (včetně informací o přesnosti s jakou jsou tyto souřadnice sbírány), nadmořské výšky (včetně informací o přesnosti naměřené nadmořské výšky), tzv. heading a rychlost pohybu. Výchozí hodnoty těchto položek jsou nastaveny na zaokrouhlování na stovky. Součástí objektu s GPS daty, který je poskytován funkcí navigator.geolocation.getCurrentPosition() je i časové razítko. Data v tomto razítku jsou udávány s přesností na milisekundy. V případě aktivované funkce obalení objektu window.Date je hodnota časového razítka v poskytovaném objektu s GPS daty zaokrouhlena podle nastavení pro obalování objektu window.Date.

- **Objekt window.XMLHttpRequest** – pro obalení tohoto objektu je na stránce s nastavením možnost zablokovat všechny požadavky typu window.XMLHttpRequest, nebo je zde možnost zobrazit uživateli podrobnosti o dotazu a nabídnout rozhodnutí provést požadavek, nebo jej zamítnout. Výchozím nastavením pro zapnutou funkcionalitu obalování tohoto objektu je možnost poskytnout uživateli informace a nechat jej udělat rozhodnutí.

Kapitola 6

Podrobnosti implementace doplňku

V této kapitole jsou popsány podrobnosti konkrétní implementace doplňku, který je vytvářen v rámci této diplomové práce. V následujících sekcích se nachází popis částí konkrétní implementace a vysvětlení principu funkčnosti. [1]

6.1 Princip obalování JavaScriptových prvků

Doplněk využívá techniku obalení, při které dochází k vytváření tzv. obálky. Tato obálka obaluje určitý úsek JavaScriptového kódu tak, že dochází k vytváření oblasti s lokální platností proměnných[2]. To znamená, že veškeré proměnné definované uvnitř tohoto prostoru nejsou přístupné mimo tento prostor, jsou tedy zapouzdřeny uvnitř tohoto prostoru a manipulovat s nimi může pouze kód uvnitř tohoto prostoru. Tohoto faktu využijeme k tomu, abychom z vnějšku znemožnili přístup k řadě JavaScriptových funkcí a objektů napřímo, jediná možnost využívat tento kód bude prostřednictvím definovaných pravidel. Vytvoříme tedy novou definici pro řadu JavaScriptových funkcí, tato nová definice využívá techniky obalení a zabaluje původní definice do lokálního prostoru, původní definice není přístupná z venku tohoto prostoru. Nová definice objektu pak původní definici může využít k zajištění své vlastní funkčnosti, ale původní definice nebude dostupná zvenku oblasti s lokální platností proměnných.

Syntaxe obalení používající anonymní funkci, která je okamžitě invokovaná a uvnitř které se nachází JavaScriptový kód určený k zapouzdření, je znázorněna v kódu 6.1.

```
(
    function () {
        // place to put code, that redefines original implementaion
    }
) ();
```

Kód 6.1: Příklad JavaScriptového kódu, který obaluje vybraný kód do prostoru s lokální platností proměnných a ihned jej invokuje.

[1] Při implementaci bylo vycházeno z volně dostupné knihy popisující technologii JavaScript. Kniha je dostupná zde http://speakingjs.com/es5/index.html

[2] Oblast rozsahu platnosti JavaScriptových proměnných je podrobněji popsána na webové stránce https://www.zdrojak.cz/clanky/javascript-a-oblast-pusobnosti-promennych-dil-prvni

K invokaci JavaScriptového kódu využívající syntaxi popsanou v kódu 6.1 dochází v době, kdy ještě prohlížeč nezačal zpracovávat zdrojový kód načítané webové stránky, tento okamžik technologie WebExtensions označuje jako „document_start“ (v rámci implementovaného doplňku je veškerý kód využívající tuto konstrukci invokován v okamžiku „document_start“). Blíže je tento okamžik popsán v sekci 4.5 a v kódu 4.7. Je tedy jisté, že při obalování původní implementace nedošlo k dalšímu dřívějšímu obalení způsobenému kódem, který je uložen ve zdrojovém kódu načítané webové stránky, ani není implementace původní konstrukce přístupná jiným způsobem. Použití tohoto přístupu umožňuje redefinovat vybrané JavaScriptové konstrukce a zapouzdřit původní implementaci uvnitř prostoru s lokální platností proměnných. Původní implementace JavaScriptových konstrukcí tedy není přístupná zvenku a její využití je možné pouze prostřednictvím kódu, který redefinuje původní funkci.

6.2 Obalení konkrétních typů konstrukcí v JavaScriptu

Při obalování původní implementace v JavaScriptu se setkáváme se dvěma typy konstrukcí, konkrétně se jedná o funkce a objekty. Každý z těchto 2 konstrukcí se obaluje odlišným způsobem, tyto způsoby obalení jsou blíže demonstrovány a popsány v kódech 6.2 a 6.3.

V kódu 6.2 je zachycen příklad redefinování konstrukce typu objekt. V rámci tohoto kódu je nejdříve reference na původní objekt uložena do proměnné *original_object*. Následně je původní objekt přepsán funkcí tvořící nový konstruktor. Uvnitř této funkce je využita implementace původního objektu a to způsobem, kdy je nejdříve vytvořena nová instance, následně jsou pozměněny hodnoty jeho datových složek (konkrétně hodnota datové složka *atribut_1* je nastavena na 0, hodnota datové složky *atribut_2* je nastavena na „str_1“ a hodnota datové složky *atribut_3* je nastavena na *null*). Po provedení těchto změn je tato nová upravená instance vrácena.

```
(
   function () {
      var original_object = path_to_object.obj_name;
      path_to_object.obj_name = function() {
         var my_obj = new original_object();
         // perform changes to the original object
         my_obj.atribut_1 = 0;
         my_obj.atribut_2 = „str_1“;
         my_obj.atribut_3 = null;
         // return changed object
         return my_obj;
      };
   }
) ();
```

Kód 6.2: Příklad redefinování a obalení JavaScriptové konstrukce typu objekt.

V kódu 6.3 je uveden příklad redefinování původní implementace funkce s názvem *function_name*, která je umístěna v objektu *object_name*. V rámci tohoto kódu je využívána konstrukce z kódu 6.1, která je rozšířena o kód provádějící obalení původní funkce. Při zavolání této nové redefinované funkce dojde k obdržení zaokrouhlené návratové hodnoty původní funkce, kód 6.3 konkrétně nastavuje zaokrouhlení přesnosti návratové hodnoty na desítky.

```
(
   function () {
      var original_function = object_name.function_name;
      object_name.function_name = function() {
         return Math.floor((original_function.call(object_name)) / 10.0) * 10.0;
      };
   }
) ();
```

Kód 6.3: Příklad obalení JavaScriptové konstrukce typu funkce.

6.3 Obalení konkrétních konstrukcí JavaScriptu

Za použití principu obalení ze sekce 6.1 je vytvořeno 5 konkrétních implementací zapouzdření, které jsou popsány v následujících sekcích. Jedná se o obalení objektu window.Date (sekce 6.3.1), obalení funkce window.performance.now() (sekce 6.3.2), obalení funkce z prototypu HTML elementu typu Canvas (sekce 6.3.3), dále poté obalení funkce navigator.geolocation.getCurrentPosition() (sekce 6.3.4) a na závěr obalení objektu window.XMLHttpRequest (sekce 6.3.5).

6.3.1 Redefinování a obalení objektu window.Date

Původní implementace objektu window.Date byla redefinována a obalen způsobem, který je zachycen v kódu 6.4. Tento kód používá přístup uvedený v kódu 6.1, zapouzdřuje původní implementaci objektu window.Date a snižuje přesnost poskytovaných časových dat na stovky milisekund pro všechny instance tohoto objektu.

```
(
   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 6.4: Obalení a redefinování původní implementace objektu window.Date.

6.3.2 Redefinování a obalení funkce window.performance.now()

Funkce window.performance.now() V kódu 6.5 je zachycena ukázka redefinování a obalení původní funkce window.performance.now() způsobem, který původní implementaci zapouzdřuje a využívá takovým způsobem, že jakékoli volání této funkce v technologii JavaScript využije hodnotu vrácenou původní implementací, ale zaokrouhlí ji a tím sníží její přesnost. Tuto zaokrouhlenou hodnotu poté vrátí.

```
(
    function() {
        var original = window.performance.now;
        window.performance.now = function() {
            return Math.floor(original.call(window.performance)/100) * 100;
        };
    }
) ();
```

Kód 6.5: Příklad obalení a redefinování původní implementace funkce window.performance.now(), jejíž obslužný kód zajišťuje zaokrouhlení poskytované hodnoty.

6.3.3 Redefinování a obalení funkce z prototypu HTML elementu typu Canvas

Kód 6.6 zachycuje redefinování a zapouzdření funkce getContext(), která se nachází v rototypu HTML elementu typu Canvas. Skutečnost, že dochází k obalování funkce v prototypu, je dána skutečností, že HTML elementů typu Canvas se může být na stránce více a mohou také dynamicky vznikat a zanikat.

```
(
    function() {
        var original = window.HTMLCanvasElement.prototype.getContext;
        window.HTMLCanvasElement.prototype.getContext = function(param1) {
            if (confirm('Enable drawing to canvas?')) {
                return original.call(this, param1);
            }
            else {
                return null;
            }
        };
    }
) ();
```

Kód 6.6: Kód zachycující obalení funkce getContext() umístěné v prototypu HTML elementu typu Canvas.

6.3.4 Redefinování a obalení funkce navigator.geolocation.getCurrentPosition()

Při redefinování a obalení původní implementace funkce getCurrentPosition() v objektu navigator.geolocation byl opět použit princip zapouzdření využívající prostor s lokální platností proměnných. V rámci obslužného kódu uvnitř tohoto prostoru dochází k snižování přesnosti poskytnutých GPS dat a to dle aktuálního nastavení doplňku. Kompletní kód redefinování a obalení funkce navigator.geolocation.getCurrentPosition() je z důvodu jeho délky uveden v příloze D, konkrétně v kódu D.1.

6.3.5 Redefinování a obalení objektu window.XMLHttpRequest

Při redefinování a zapouzdření objektu window.XMLHttpRequest obslužný kód dle aktuálního nastavení doplňku zablokuje veškeré pokusy dotaz typu XMLHttpRequest se pro každý dotaz zeptá uživatele (včetně poskytnutí informací o prováděném dotazu). Kompletní kód redefinování a obalení objektu window.XMLHttpRequest je z důvodu jeho délky definován v příloze E, konkrétně v kódu E.1.

6.4 Spouštění obalovacích kódů

V sekci 6.3 je popsána řada JavaScriptových scriptů, které realizují konkrétní obalení a zapouzdření původních JavaScriptových konstrukcí. Tyto scritpy musí být před samotným provedením nejdříve sestaveny, a to z důvodu, aby se do nich promítlo aktuální nastavení implementovaného doplňku.

Celé sestavení tedy probíhá následujícím způsobem – nejdříve je z uložiště technologie WebExtensions načteno aktuální nastavení doplňku (pokud toto nastavení chybí nebo není validní, jsou použity výchozí hodnoty nastavení). Poté je toto nastavení zohledněno a dojde k sestavení konkrétních obalovacích scriptů, které jsou v této fázi reprezentovány jako textový řetězec. Následně jsou tyto scripty vykonávány, a to prostřednictvím vytvoření HTML elementu typu „script“ a následné naplnění tohoto elementu příslušnými atributy a vložení stromu DOM tree (vykonáním tohoto procesu prostřednictvím postupu a nástrojů, které jsou uveden v kódu 6.7 dochází k okamžitému provedení těchto scriptů).

```
var scriptTag = document.createElement('script');
scriptTag.type = 'text/javascript';
scriptTag.text = "---";
document.getElementsByTagName('html')[0].appendChild(scriptTag);
```

Kód 6.7: Kód popisující vykonání obalovacích scriptů.

Celý proces obalení původních JavaScriptových konstrukcí je vykonáván JavaScriptovým kódem. Tento kód je v rámci souborové struktury implementovaného doplňku uložen v souboru „document_start.js“. Kód zajišťující zapouzdření je invokován v době před tím, než začne proces zpracovávání zdrojového kódu načítané webové stránky. Tento fakt je velice důležitý, spouštění obalovacího kódu v této době zajistí, že žádný jiný kód v načítané webové stránce nebude nikdy schopný využívat původní implementaci. Ta je zapouzdřena uvnitř obalovacího kódu v oblasti s lokální platností proměnných.

6.5 Implementace souboru manifest.json

Soubor manifest.json je implementován s ohledem na pravidla dané technologií WebExtensions (podrobněji jsou tato pravidla popsána v sekci 4.5). V kódu 6.8 je popsána základní struktura souboru manifest.json implementovaného doplňku. Tato základní struktura popisuje implementaci všech 3 povinných klíčů () a dále je zde zchycena implementace nepovinných klíčů.

```
{
  "browser_action": {
    "default_title": "JavaScript Restricter"
  },
  "description": "JavaScript Restricter",
  "homepage_url": "https://www.fit.vutbr.cz",
  "manifest_version": 2,
  "name": "JavaScript Restricter",
  "permissions": ["storage"],
  "version": "1.0",
  "applications": {
    "gecko": {
      "id": "username@server.com",
      "strict_min_version": "57.0a1"
    }
  }
}
```

Kód 6.8: Implementace klíče základních dat v souboru manifest.json v implementovaném doplňku

V kódu 6.9 je zachycen část souboru manifest.json, který implementuje klíč „options_ui“. V rámci hodnoty tohoto klíče se nachází odkaz na HTML stránku options.html a nastavení klíče „broser_style“ na hodnotu true.

```
"options_ui": {
  "page": "options.html",
  "browser_style": true
},
```

Kód 6.9: Implementace klíče „options_ui“ v souboru manifest.json v implementovaném doplňku

Soubor manifest.json implementovaného doplňku v sobě nese také klíč „backgound“, jehož konkrétní implementace je zachycena v kódě 6.10. Hodnota tohoto klíče odkazuje na soubor background.js, který obsahuje JavaScriptový kód běžící na pozadí webové stránky.

```
"background": {
   "scripts": ["background.js"]
}
```

Kód 6.10: Implementace klíče „background“ v souboru manifest.json.

Dalším důležitým klíčem, který je obsažen v souboru manifest.json implementovaného doplňku je klíč „content_scripts“. Hodnota tohoto klíče definuje, že obsah souboru document_start.js se bude spouštět v okamžiku „document_start“ a toto platí pro všechny URL adresy. Na obrázku 6.11 je uveden výňatek souboru manifest.json s implementací tohoto klíče.

```
"content_scripts": [
   {
      "matches": ["<all_urls>"],
      "js": ["document_start.js"],
      "run_at": "document_start"
   }
]
```

Kód 6.11: Implementace klíče „content_scripts“ v souboru manifest.json.

6.6 Implementace stránky s nastavením doplňku

Stránka s nastavením implementovaného doplňku je tvořena HTML souborem s názvem „options.html“ a vede na něj reference ze souboru manifest.json. Soubor „options.html“ v sobě obsahuje referenci na soubor s CSS styly „options.css“, který definuje formátování stránky s nastavením a soubor „options.js“, který implementuje JavaScriptový zajišťující práci s uložištěm, tedy načítání a ukládání konfigurace doplňku (blíže je práce s uložištěm popsána v sekci 6.6).

Soubor „options.css“ obsahuje definici řady CSS stylů, které definují vzhled a zarovnání prvků tvořící stránku s nastavení. Definováno je zde zarovnání a odsazení jednotlivých položek, velikost a barva textu, styl zobrazování zaškrtávacích políček a políček typu select.

Soubor „options.js“ obsahuje JavaScriptový kód definující funkce pro zápis aktuálního nastavení doplňku do uložiště a funkci pro načtení nastavení doplňku z uložiště. Načítání nastavení se děje při zpracovávání zdrojového kódu souboru „options.html“ (tedy při návštěvě stránky s nastavením nebo při znovunačtení stránky s nastavením) a po získání dat z uložiště dochází k nastavení HTML prvků na příslušné hodnoty. K zápisu dat do uložiště dochází po kliknutí na tlačítko „Uložit nastavení“, které je implantované v souboru „options.html“. V rámci procesu zápisu dochází nejdříve k zjištění aktuální konfigurace stránky s nastavením (nastavení jednotlivých HTML elementů, tedy zaškrtávacích políček, elementů typu select a obdobných), poté dochází k sestavení dat ve formátu JSON a tato data jsou poté zapsána do uložiště

technologie WebExtensions (samotný zápis a načtení dat s nastavením je detailně popsáno v sekci 6.6).

Vzhled uživatelského rozhraní stránky s nastavením je blíže popsán v sekci 6.8, kde se také nacházejí i obrázky zachycující konkrétní podobu stránky s nastavením (konkrétně se jedná o obrázky 6.2 a 6.3).

6.7 Načítání a ukládání dat s nastavením

Načítání a ukládání dat s nastavením se děje prostřednictvím uložiště, které je blíže popsáno v sekci 4.6. V rámci implementovaného doplňku je ukládání dat do uložiště implementováno pouze ve stránce reprezentující položky nastavení. Načítání dat z uložiště je v rámci doplňku implementováno na dvou místech, v rámci JavaScriptového scriptu „document_start.js“, který obsahuje kód provádějící zapouzdření JavaScriptových konstrukcí, a v rámci stránky reprezentující položky nastavení (aby bylo možné po otevření stránky zobrazit prvky nastavená způsobem, jakým byly v minulosti nastaveny uživatelem a tedy způsobem, který odpovídá současné konfiguraci funkčnosti doplňku).

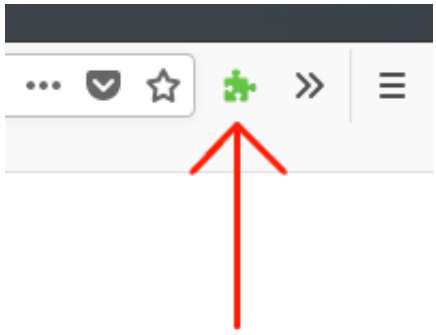
Data reprezentující konfiguraci doplňku jsou ve formátu JSON, jehož zkrácená struktura je zachycena v kódu 6.12. (z důvodu délky kódu je celá implementace popsána v příloze C, konkrétně v kódu C.1). Tato JSON struktura je tvořena dvojící klíč-hodnota, kde ke klíči *extensions_settings_data* je přiřazena hodnota tvořená opět objektem ve formátu JSON. Tento objekt v sobě implementuje 5 párů klíč-hodnota, názvy klíčů těchto párů jsou následující - *window_date*, *window_performance_now*, *window_html_canvas_element*, *navigator_geolocation* a *window_xmlhttprequest*. Každý z těchto klíčů reprezentuje jednu JavaScriptovou konstrukci, která je v rámci implementovaného doplňku zapouzdřována. Hodnota každého odpovídající klíče implementuje JSON objekt obsahující konfiguraci obalovacího JavaScriptového kódu.

```
extension_settings_data: {
    window_date: {
        // several key-value pairs representing settings
    },
    window_performance_now: {
        // several key-value pairs representing settings
    },
    window_html_canvas_element: {
        // several key-value pairs representing settings
    },
    navigator_geolocation: {
        // several key-value pairs representing settings
    },
    window_xmlhttprequest: {
        // several key-value pairs representing settings
    }
}
```

Kód 6.12: JSON struktura reprezentující konfiguraci doplňku (zkrácená verze).

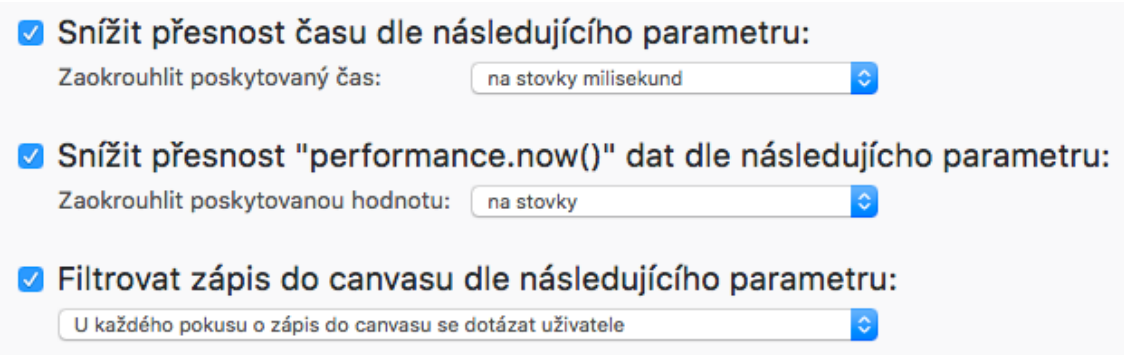
6.8 Implementace uživatelské rozhraní doplňku

Implementovaný doplněk nabízí v pravém horním rohu okna webového prohlížeče toolbar tlačítko, které je znázorněno na obrázku 6.1. Tlačítko je automaticky viditelné po celou dobu, kdy je doplněk nainstalovaný a aktivovaný a slouží jako hlavní vstupní bod pro přesun na stránku s podrobnějším nastavením jednotlivých funkcí doplňku.



Obrázek 6.1: Toolbar tlačítko implementovaného doplňku, které se nachází v pravém horním rohu okna internetového prohlížeče. Po kliknutí na toto tlačítko se otevře stránka s nastavením implementovaného doplňku.

Po kliknutí na toolbar tlačítko implementovaného doplňku se otevře nastavení doplňku. V nastavení je uvedena celá řada možností konfigurace funkcionality doplňku. Na obrázku 6.2 je zachycena ukázka části nastavení umožňující konfigurovat první 3 funkce implementovaného doplňku, konkrétně se jedná o funkce snížení přesnosti časových dat poskytovaných JavaScriptovým objektem window.Date, dále snížit přesnost dat poskytovaných funkcí window.performance.now() a třetí implementovanou funkcí je možnost filtrovat zápis do HTML objektu typu Canvas.



Obrázek 6.2: Výňatek stránky s nastavením zachycující funkce snižující přesnost dat poskytovaných objektem window.Date a funkcí window.performance.now() a funkce filtrovat zápis nebo dotázat se uživatele při pokusu o zápis do HTML elementu typu Canvas.

Na obrázku 6.3, je zachyceno uživatelské rozhraní části nastavení, konkrétně se jedná o konfiguraci funkcí pro snížení přesnosti poskytovaných GPS dat a o funkci filtrování požadavků typu XMLHttpRequest. V rámci snížení přesnosti GPS dat se dne nachází možnost kompletně anonymizovat poskytovaný GPS data (všechny údaje poskytnutého objektu budou vynulovány), nebo snížit jejich přesnost dle aktuální konfigurace této funkce. Dále se zde nachází možnost konfigurace filtrování dotazů typu XMLHttpRequest a nabízí se zde možnost úplného zablokování všech požadavků na dotazy tohoto typu, nebo lze doplněk nastavit, aby se při každém takovém dotazu zeptal uživatele na povolení.

Obrázek 6.3: Výňatek stránky s nastavením zachycující funkce snižující přesnost GPS dat poskytovaných funkcí navigator.geolocation.getCurrentPosition() a funkce umožňující filtrování window.XMLHttpRequest dotazů

Kompletní uživatelské rozhraní stránky s nastavením je uvedeno z důvodu jejího rozsahu v příloze B, konkrétně na obrázku B.1. Na tomto obrázkus je zachycené jednak uživatelské rozhraní znázornění na obrázcích 6.2 a 6.3, ale také hlavička stránky s nastavením s doplňujícími informacemi a také zápatí stránky s nastavením umožňující uložit nastavení, doplněk deaktivovat, nebo jej kompletně odstranit.

Kapitola 7

Testování

Tato kapitola popisuje testování, které bylo provedeno na implementovaném doplňku. Testování probíhalo ve webovém prohlížeči Mozilla Firefox ve verzi 60.0. Celý proces testování se skládá ze 3 částí, každá část je v rámci této kapitoly popsána v samostatné sekci. První částí je testování implementovaného doplňku na ukázkovém příkladu. Tento příklad je obsahem přiloženého DVD a byl vyvinut přímo pro účel ověření fungování všech pěti implementovaných funkcí, které doplněk nabízí. Druhá část testování je prováděno na skutečných webových stránkách, aby bylo ověřeno chování a fungování doplňku v běžném provozu. Třetí část procesu testování probíhala prostřednictvím JavaScriptové konzole webového prohlížeče a prostřednictvím řady JavaScriptových scriptů vyvinutých pro tento účel. Tento proces byl zaměřen na cílené prozkoumání a ověření chování samotného JavaScriptu při různé konfiguraci implementovaného doplňku.

7.1 Testování na ukázkovém příkladu

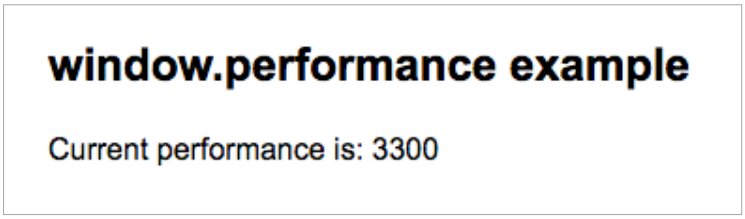
K doplňku je sestaven testovací ukázkový příklad, který je dostupný na přiloženém DVD (konkrétně na adrese */demo_example/index.html*). Tento testovací příklad nabízí demonstraci práce s objektem window.Date(), s funkcí window.performance.now(), s HTML elementy typu Canvas, s polohovými GPS daty poskytnutými prostřednictvím funkce navigator.geolocation.getCurrentPosition() a s dotazy typu window.XMLHttpRequest. Data poskytnuté těmito funkcemi a objekty jsou v rámci demonstračního příkladu na stránce zobrazovány. V případě dat posyktnutých objektem window.Date() a funkcí window.performance.now() jsou data neustále aktualizována z důvodu jednoznačné identifikovatelnosti přesnosti poskytovaných dat. V případě polohových GPS dat jsou tato data zobrazována po kliknutí na příslušné tlačítko, je tedy možné žádat o data opakovaně bez nutnosti znovunačítání webové stránky. Provedení dotazu typu window.XMLHttpRequest je dostupné také na stisknutí tlačítka, v případě provedení dotazu se jeho výsledek zobrazí ve vyskakovacím okně.

V rámci procesu testování na ukázkovém příkladu byl doplněk nejdříve vypnutý kompletně, poté docházelo k aktivaci vždy jen jedné funkcionality a probíhalo testování s různými konfiguracemi této funkcionality. V závěru tohoto testování byla postupně aktivována různá kombinace funkcionality s různou kombinací konfigurace každé funkcionality, aby byla důkladně otestována funkčnost implementovaného doplňku v co nejvíce situacích.

Na obrázku 7.1 jsou zachycena data z objektu window.Date, konkrétně se jedná o časové údaje popisující aktuální čas a jsou zobrazena ve formátu *hodiny:minuty:sekundy:milisekundy*. Tento obrázek zobrazuje data poskytnutá po aktivaci doplňku a nastavení jeho funkcionality na snížení přesnosti časových dat zaokrouhlení na stovky milisekund. Obrázek sice zachycuje pouze jeden aktuální čas, při testování na ukázkovém příkladu však dochází k dynamické aktualizaci zobrazovaných časových dat, uživatel tedy jasně vidí rozdíl mezi zapnutou a vypnutou funkcionalitou doplňku (při vypnuté funkcionalitě je viditelná neustálá změna hodnoty s přesností na milisekundy, při zapnuté funkcionalitě a při snižování přesnosti zaokrouhlením na milisekundy je viditelná hodnota času měnící se po stovkách milisekund).

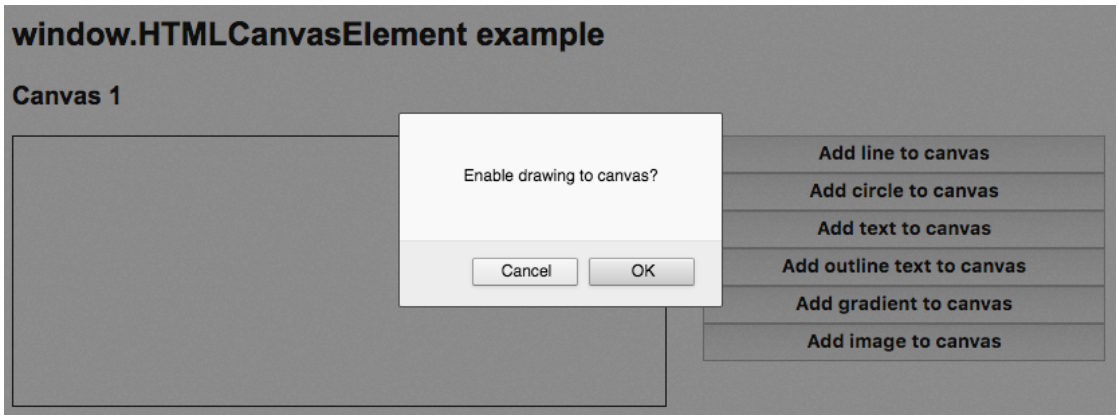
Obrázek 7.1: Ukázka zachycující aktuální čas, jehož hodnota je vlivem aktivovaného doplňku zaokrouhlena na stovky milisekund.

Na obrázku 7.2 jsou zachycena data poskytnutá funkcí window.performance.now(), a to v době, kdy byl implementovaný doplněk aktivovaný a jako parametr bylo nastaveno snížení přesnosti vrácené hodnoty zaokrouhlením na stovky. Na tomto obrázku je opět jako v případě snížení přesnosti časových údajů poskytovaných objektem window.Date zachycen jeden časový okamžik. Při testování na ukázkovém příkladu dochází k neustálé aktualizaci poskytovaných hodnot, uživatel tedy vidí v reálném čase neustálé změny v poskytované hodnotě (v případě vypnuté funkcionality snížení přesnosti implementovaného doplňku), nebo hodnoty zaokrouhlené na stovky měnící se každých 100 milisekund (v případě zapnuté funkcionality snížení přesnosti poskytovaných dat).



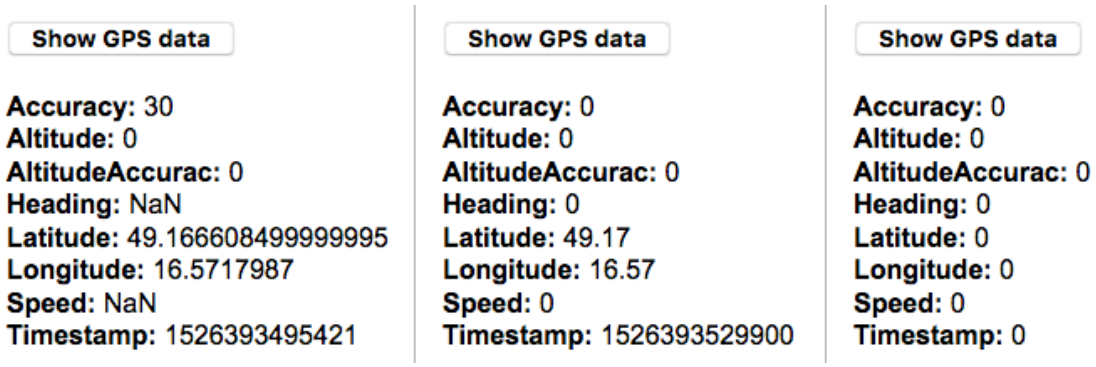
Obrázek 7.2: Ukázka zachycující aktuální hodnotu funkce window.performance.now(), jejíž hodnota je vlivem funkce doplňku zaokrouhlena na stovky.

Na obrázku 7.3 je znázorněno dialogové okno, které se zobrazí při každém pokusu o zápis do HTML elementu typu Canvas. Toto dialogové okno se zobrazí v případě zapnuté funkcionality dotázání se uživatele na povolení zápisu do HTML elementu typu Canvas. Vedle této možnosti implementovaný doplněk umožňuje také kompletní blokování všech pokusů o zápis do všech HTML elementů typu Canvas.



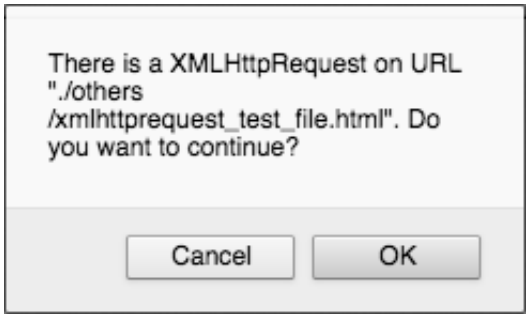
Obrázek 7.3: Obrázek zachycující žádost, která uživatele vyzývá k udělení povolení zápisu do HTML elementu typu Canvas.

Obrázek 7.4, zachycuje tři případy s informacemi o GPS datech poskytnuté funkcí navigator.geolocation.getCurrentPosition() v rámci testovacího příkladu. V prvním ze tří sloupečků znázorněných na tomto obrázku jsou data poskytnutá při vypnuté funkcionalitě doplňku. Zeměpisná šířka a zeměpisná délka jsou zde uvedeny s přesností na řadu desetinných míst, časové razítko je zde uvedeno s přesností na milisekundy. Ve druhém sloupečku je funckionalita doplňku aktivovaná, dochází k zaokrouhlení zeměpisné šířky a délky na 2 desetinná místa (dle aktuálního zvoleného nastavení doplňku), časové razítko je zaokrouhleno na stovky milisekund (dle aktuálního nastaveného zaokrouhlování časových dat) a ostatní položky jsou zaokrouhleny na stovky. Ve třetím sloupečku se nachází GPS data, která jsou poskytnuta při nastavení doplňku na možnost, při které dojde k vynulování všech hodnot.



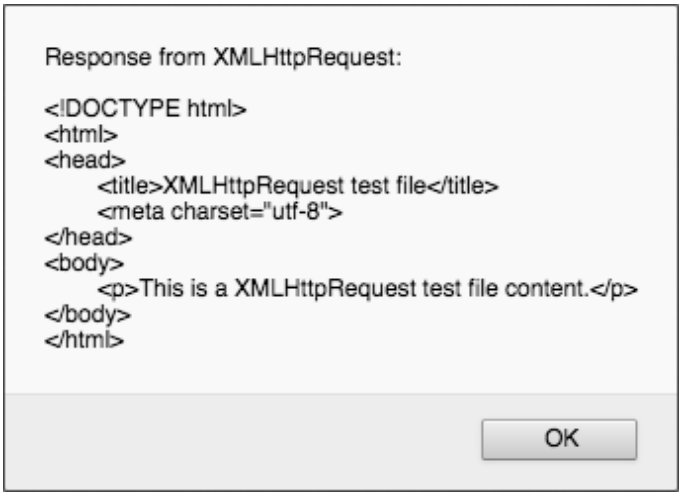
Obrázek 7.4: Obrázek zachycující obsah hodnot uvnitř objektu poskytnutého funkcí navigator.geolocation.getCurrentPosition().

Na obrázku 7.5 je znázorněno dialogové okno, které se objeví při pokusu o provedení dotazu typu window.XMLHttpRequest při zapnuté funkcionalitě doplňku pro filtrování tohoto typu dotazů na nastavení této funkcionality na možnost nechat o provedení dotazu rozhodnout uživatele. Uživatel poté buď odsouhlasí provedení dotazu, nebo jej zablokuje. V případě odsouhlasení dojde v rámci testovacího příkladu k následnému zobrazení dalšího dialogového okna, které zobrazí podrobnosti provedeného dotazu.



Obrázek 7.5: Dialogové okno vyzývající uživatele k rozhodnutí, zda daný dotaz typu window.XMLHttpRequest povést, nebo zablokovat.

Na obrázku 7.5 je znázorněno informativní okno s výsledkem provedeného dotazu typu window.XMLHttpRequest. Toto okno se v rámci testovacího příkladu objeví vždy, pokud je dotaz proveden, tedy buď je funkcionalita implementovaného doplňku pro filtrování těchto dotazů kompletně deaktivovaná, nebo je nastavena na potvrzení a uživatel udělil souhlas s provedením dotazu.



Obrázek 7.6: Okno poskytující uživateli detaily týkající se provedeného window.XMLHttpRequest dotazu.

7.2 Testování na běžných webových stránkách

Vedle testování na připraveném ukázkovém příkladu bylo provedeno také testování na reálných webových stránkách, a to z důvodu, abychom prověřili funkcionalitu implementovaného doplňku také v reálném provozu, tedy na skutečných webových stránkách. Provedením tohoto procesu dojde k ověření, že omezování a snižování přesnosti vybraných JavaScriptových nástrojů funguje správným způsobem. V rámci této sekce je provedena řada testování, každé testování se specializuje na prověření jedné funkcionality implementovaného doplňku.

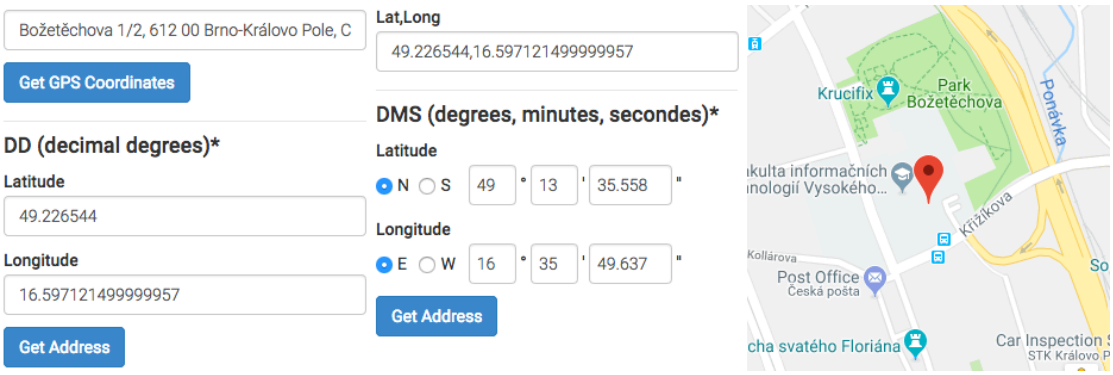
K testování omezení poskytování časových údajů z JavaScriptového objektu window.Date byla použita webová stránka http://www.estopwatch.net/, která poskytuje službu měření časových intervalů s přesností na milisekundy. V první fázi testování byla v nastavení doplňku zapnuta pouze funkce snížení přesnosti časových dat dostupných z objektu window.Date, ostatní funkce doplňky byly vypnuté. Přesnost poskytování času byla nastavena na zaokrouhlování času na stovky milisekund, obrázek 7.7 zachycuje výřez testované stránky a poskytuje náhled na čas se sníženou přesností.

V druhé fázi testování byly provedeny stejné úkony jako v první fázi, jediný rozdíl byl ve skutečnosti, že daný doplněk měl kromě funkce snížení dat poskytovaných objektem window.Date zapnuté i další funkce. Tato druhá fáze testování proběhla opět bez problémů.

Obrázek 7.7: Výřez stránky http://www.estopwatch.net/ zachycující spuštěnou časomíru a zapnutou funkcionalitu implementovaného doplňku zajišťující snížení přesnosti časových dat poskytovaných objektem window.Date zaokrouhlením na stovky milisekund.

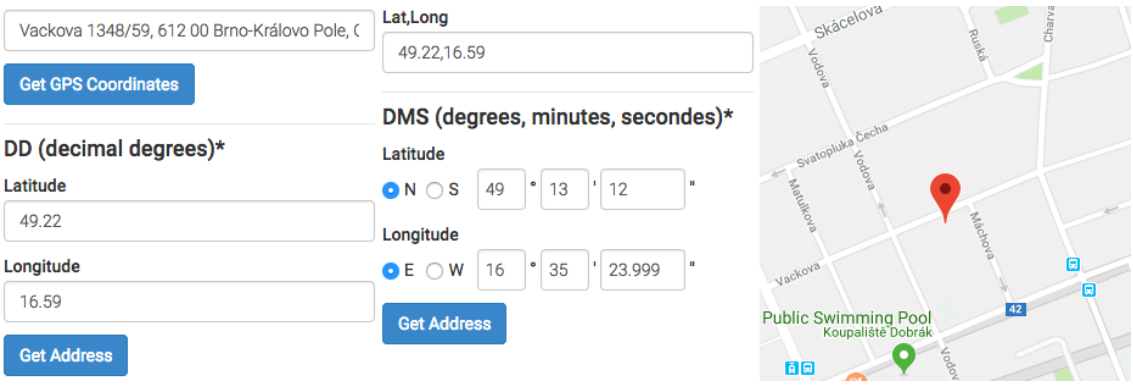
Testování dotazování se uživatele a blokování zápisu do HTML elementu typu Canvas bylo prováděno na celé řadě webových stránek. Jako příklad lze uvést webové služby https://www.mapy.cz a https://www.google.cz/maps, nebo jen jednoduché webové stránky jako například https://www.seznam.cz/. Všechny tyto webové stránky žádali o povolení zápisu do HTML elementu typu Canvas. U služeb typu online mapy se dá usuzovat, že tento zápis slouží k vyobrazování mapových podkladů a dalších objektů a k zajištění fungování poskytovaných služeb. U celé řady jiných webových stránek slouží především k vykreslování reklam.

Testování snížení přesnosti a nulování GPS dat poskytnutých objektem navigator.geolocation.getCurrentPosition() bylo testováno na celé řadě webových stránek. Příkladem takové webové stránky je https://www.gps-coordinates.net/, která zobrazuje podrobné informace o GPS datech návštěvníka stránky včetně zobrazení polohy na mapě a na níž bude demonstrována funkčnost implementovaného doplňku. Na obrázku 7.8 je znázorněn výsledek poskytnutý touto stránkou v době, kdy byla funkcionalita implementovaného doplňku vypnuta. Na obrázku je vidět zeměpisná šířka i zeměpisná délka zachycená na řadu desetinných míst, stránka poskytla i informace o názvu příslušné ulice a znázornila polohu na mapě (udávané údaje jsou tedy bez snížení přesnosti dat a odpovídají realitě).



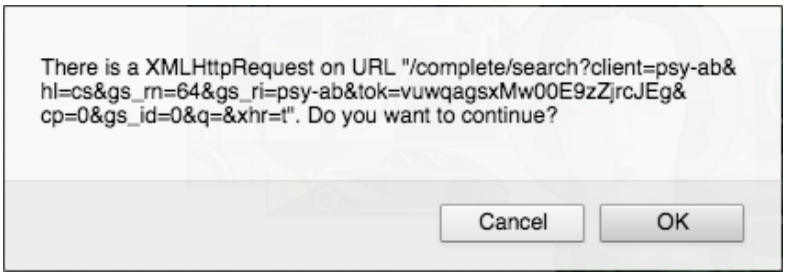
Obrázek 7.8: Data poskytnutá webovou stránkou https://www.gps-coordinates.net/ před aktivací funkcionality snižování přesnosti v implementovaném doplňku

Na obrázku 7.9 je znázorněn výsledek poskytnutý stránkou https://www.gps-coordinates.net/ při aktivované funkci snižování přesnosti dat s konkrétním nastavení zaokrouhlování zeměpisné šířky a zeměpisné délky na 2 desetinná místa. Dle obrázku je zřejmé, že snížení přesnosti tímto způsobem zvýší anonymitu uživatele, jeho zdánlivá poloha po zaokrouhlení se nachází stovky metrů od skutečné polohy uživatele (skutečná poloha uživatele se nachází v kampusu VUT FIT na adrese Božetěchova 1/2 612 00 Brno-Královo pole).



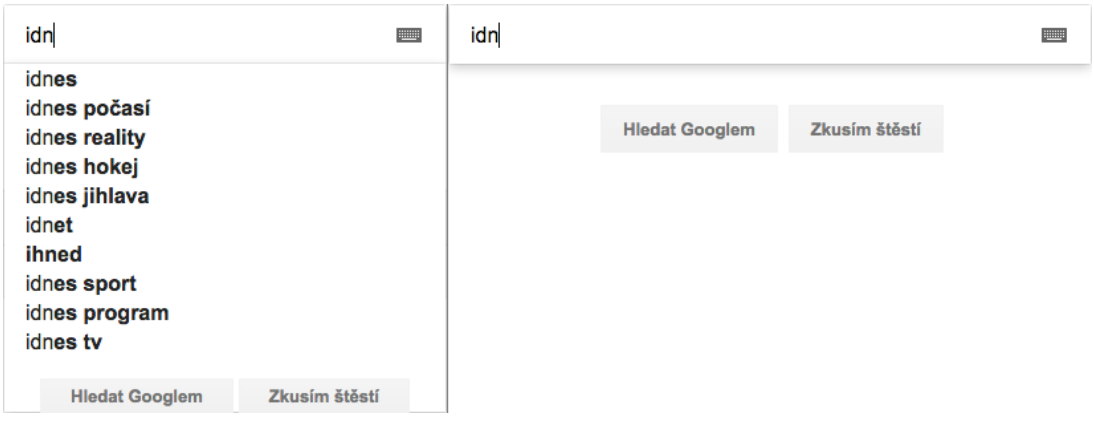
Obrázek 7.9: Data poskytnutá webovou stránkou https://www.gps-coordinates.net/ po aktivaci funkcionality snižování přesnosti v implementovaném doplňku (nastaveno zaokrouhlování souřadnic na 2 desetinná místa)

Testování blokování dotazů typu window.XMLHttpRequest probíhalo na celé řadě webových stránek. Při vypnuté funkcionalitě webových stránek probíhali všechny tyto dotazy bez vědomí uživatele a standardním způsobem. Při zapnuté funkcionalitě doplňku a při konfiguraci na vyzývání uživatele k rozhodnutí o zablokování, nebo provedení dotazu docházelo při procházení internetu k řadě detekovaných dotazů. Podrobnosti jednoho takového prováděného dotazu jsou zachyceny na obrázku 7.10, kde je vyobrazen dialog, který se uživatele dotazuje na rozhodnutí povolit, nebo zablokovat probíhající dotaz.



Obrázek 7.10: Dialogové okno nechávající uživatele rozhodnout, zda daný dotaz typu window.XMLHttpRequest provést, nebo zablokovat.

Obecně se tedy dá říci, že v rámci procházení internetu je prováděno velké množství dotazů typu window.XMLHttpRequest, velká část z nich slouží k zajištění funkcionality dané webové stránky. Kompletním zablokování by docházelo ke znemožnění určité funkcionality. Jako příklad dotazu, který slouží k poskytnutí funkcionality webové stránky lze uvést našeptávací nabídku při zadávání vyhledávacího výrazu ve vyhledávači Google. Na obrázku 7.11 je vlevo zachycena situace, při které nedochází k blokování dotazů prostřednictvím doplňku, našeptávací okno je tedy zobrazeno a plně funkční (reaguje na každou stlačenou klávesu a upravuje zobrazené nabízené položky). Vpravo je znázorněna situace, při které jsou implementovaným doplňkem blokovány všechny dotazy typu window.XMLHttpRequest. Následkem tohoto blokování nedochází k žádnému našeptávání, nejsou viditelné žádné možnosti.



Obrázek 7.11: Průběh zadávání vyhledávacího příkazu do vyhledávače Google.com, v levé části je zachycena situace, kdy dotazy typu window.XMLHttpRequest nejsou blokovány vůbec, v pravé části je zachycena situace, kdy jsou tyto dotazy doplňkem kompletně zablokovány.

7.3 Testování v konzoli JavaScriptu

Vedle testování na připraveném ukázkovém příkladu a testování na skutečných webových stránkách bylo provedeno také testování přímo v konzoli JavaScriptu ve webovém prohlížeči Mozilla Firefox ve verzi 60.0. Toto testování umožňuje na rozdíl od předchozích dvou způsobů detailněji otestovat JavaScriptem poskytované hodnoty a prováděné akce. V této sekci jsou uvedeny konkrétní kódy v jazyce JavaScript, kterými byla funkcionalita testována.

Objekt Date v sobě obsahuje informace o čase a to s přesností na milisekundy. V kódu 7.1 je uvedena ukázka JavaScriptového kódu, jehož spuštění vypíše do konzole webového prohlížeče aktuální čas v milisekundách.

```
var dateObject = new Date();
console.log(dateObject.getTime());
```

Kód 7.1: Příklad testovacího JavaScriptového kódu, který do konzole vypíše aktuální čas v milisekundách.

V rámci prováděného testování byl kód 7.1 opakovaně spouštěn a hodnoty vypsané do terminálu byly vzájemně porovnávány. Testování bylo prováděno pro různé nastavení doplňku, tedy pro vypnutou funkcionalitu snížení přesnosti času i pro zapnutou funkcionalitu s nastavením zaokrouhlování času postupně na desítky milisekund, stovky milisekund a na celé sekundy.

Po vyhodnocení výstupních hodnot v terminálu vyšlo najevo, že doplněk funguje správně pro zapnutou i vypnutou funkcionalitu snížení přesnosti času, a to pro všechny nastavení zaokrouhlení času, tedy pro zaokrouhlení na desítky milisekund, stovky milisekund i pro zaokrouhlení na samotné sekundy.

U JavaScriptové funkce window.performance.now() dochází po zapnutí funkcionality doplňku k zaokrouhlování přesnosti vrácené hodnoty na jednotky, desítky, stovky, nebo tisíce dle aktuálního nastavení doplňku. V kódu 7.2 je uveden příklad JavaScriptového kódu, který do konzole vypíše aktuální hodnotu vrácenou funkcí window.performance.now().

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

Kód 7.2: Příklad testovacího JavaScriptového kódu, který do konzole vypíše aktuální hodnotu vrácenou funkcí window.performance.now().

Po vyhodnocení výstupních hodnot v terminálu bylo naznáno, že doplněk funguje správně pro zapnutou i vypnutou funkcionalitu snížení přesnosti hodnoty vrácené funkcí window.performance.now() a to pro všechny možnosti nastavení. Tedy pro nastavení zaokrouhlení vrácené hodnoty na jednotky, desítky, stovky i na tisíce.

Při testování blokace zápisu do HTML elementu typu Canvas byl použit JavaScriptový kód uvedený v kódu 7.3. Tento kód nejdříve dynamicky vytvoří HTML element typu Canvas a poté se do něj pokusí zapsat.

```
var canvasTag = document.createElement('canvas');
canvasTag.id = 'dynamically-created-canvas';
document.body.appendChild(canvasTag);
var canvas = document.getElementById('dynamically-created-canvas');
var ctx = canvas.getContext('2d');
ctx.font = '24px Arial';
ctx.fillText('Hello World!', 10, 50);
```

Kód 7.3: Příklad testovacího JavaScriptového kódu, který vytvoří HTML element typu Canvas a pokusí se do něj zapsat.

Testování probíhalo opakovaně pro všechny možnosti konfigurace implementovaného doplňku. Pro vypnutou funkcionalitu filtrování zápisu do Canvasu proběhly všechny pokusy o zápis úspěšně. Pro zapnutou funkcionalitu filtrování a pro nastavenou konfiguraci na dotázání se uživatele na rozhodnutí došlo vždy k zobrazení dialogového okna. Pro obě možnosti, tedy povolení nebo zamítnutí zápisu do Canvasu, došlo vždy ke správnému chování, tedy provedení, nebo neprovedení operace. V případ konfigurace doplňku na možnost zablokování všech pokusů o zápis do Canvasu docházelo skutečně v rámci testování k blokaci všech pokusů zápisu do Canvasu.

Testování GPS dat poskytnutých funkcí navigator.geolocation.getCurrentPosition() probíhalo prostřednictvím zadávání kódu 7.4 do konzole webového prohlížeče. Tento kód získá a následně vloží objekt *Position* do konzole. Uživatel má poté možnost tento objekt v konzoli prozkoumat, kliknutím na objekt se zobrazí jeho vnitřní struktura a hodnoty vnitřních datových složek. Díky tomu je možné ověřit, jestli poskytnutá data skutečně odpovídají aktuální konfiguraci doplňku. V rámci prováděného testování data ve všech případech odpovídala nastavení implementovaného doplňku.

```
function getLocationObject() {
   if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(showPositionObjectInConsole);
   } else {
      console.log("Geolocation not supported.");
   }
}
function showPositionObjectInConsole(positionObject) {
   console.log(positionObject);
}
getLocationObject();
```

Kód 7.4: Příklad testovacího JavaScriptového kódu, který do konzole webového prohlížeče přidá objekt Position, který reprezentuje polohová data a je možné jej v konzoli procházet.

Testování blokování dotazů typu window.XMLHttpRequest probíhalo v JavaScriptové konzoli webového prohlížeče prostřednictvím příkazů uvedených v kódu 7.5. Tento JavaScriptový kód byl v rámci testování opakovaně spouštěn na doméně https://www.seznam.cz, a to pro různou konfiguraci implementovaného doplňku. Při vypnuté funkcionalitě filtrování dotazů typu XMLHttpRequest byl do konzole přidán celý XMLHttpRequest[1] objekt, který uvnitř sebe obsahoval položky mimo jiné návratový kód statusu a vrácenou hodnotu, v našem případě kód reprezentující SVG obrázek, na který byl veden tento dotaz.

```
function runRequest(requestUrl) {
   var myRequest = new XMLHttpRequest();
   myRequest.open("GET", requestUrl);
   myRequest.send();
   myRequest.onload = () => showOutput(myRequest);
}

function showOutput(response) {
   console.log(response);
}

runRequest("https://www.seznam.cz/media/img/logo_v2.svg");
```

Kód 7.5: Příklad testovacího JavaScriptového kódu, který v konzoli na doméně https://www.seznam.cz provede dotaz typu XMLHttpRequest a vypíše do konzole vrácená data.

7.4 Zhodnocení výsledků testování

Doplněk implementovaný v rámci této diplomové práce prošel řadou testování na ukázkovém testovacím příkladu, na reálných webových stránkách a také cíleným testování a zkoumám přímo prostřednictvím konzole technologie JavaScript ve webovém prohlížeči Mozilla Firefox ve verzi 60.0.

Žádná z provedených tří částí testování neobjevila v implementovaném doplňku žádnou chybu znemožňující využívat jakoukoli implementovanou funkcionalitu doplňku, a to pro žádnou z testovaných konfigurací a kombinací konfigurací.

Funkce snížení přesnosti časových dat zaokrouhlením nabízí dostatečnou obranu proti identifikaci na základě časových značek, navíc snížení přesnosti nenaruší běžné fungování a používání služeb internetu. Funkce blokování zápisu do HTML elementu typu Canvas zablokuje řadu reklam a většinou neomezí funkcionalitu webových stránek (kromě případu online služeb, které zápis do Canvasu využívají pro zajištění své funkcionality – zobrazení mapových podkladů, online hry atd.).

[1] Celý objekt XMLHttpRequest je detailně popsán zde - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Kapitola 8

Závěr

Ze studia současného stavu problematiky ochrany soukromí uživatelů internetu vyplývá, že uživatelé jsou monitorováni celou řadou přístupů a mechanismů, které provádějí sbírání a následné šíření osobních a citlivých informací o těchto uživatelích. Dále se také ukázalo, že současné doplňky webových prohlížečů, které se specializují na ochranu soukromí návštěvníků webových stránek, nezajišťují uživatelům dostatečné bezpečí, spolehlivost ochrany ani komfort při jejich používání.

Při testování konkrétních doplňků webových prohlížečů se ukázalo, že tyto nástroje nabízejí jen určitý typ ochrany, většinou se jedná o blokování technologie JavaScript nebo blokování externě načítaných zdrojů. Vzhledem k faktu, že technologie JavaScript je dnes nedílnou součástí funkcionality většiny webových stránek, jejím blokováním se sice znemožní funkčnost některých sledovacích mechanismů, ale ve velkém také dochází k narušení důležitých funkcí celé řady webových stránek. U doplňků nabízejících blokování externě načítaných zdrojů dochází ve velkém k chybám v rozhodování, které takové zdroje zablokovat a které povolit. Dochází tak často k situacím, kdy některé sledovací prvky nebyly zablokovány a dále provádí sledovací činnost, a také k situacím, kdy byly zablokovány externí zdroje potřebné ke správnému zobrazování a fungování webové stránky. Sledovací mechanismy nemusejí být vždy nutně načítány externě ze serveru třetí strany. Mohou být do webové stránky vloženy přímo jejím autorem.

Studium problematiky soukromí a anonymity uživatelů internetu a studium současných doplňků pro zajištění ochrany těchto uživatelů vedlo k tvorbě návrhu funkcionality doplňku, který je implementován v rámci této diplomové práce. Implementovaný doplněk je postaven na principu redefinování a obalování původních konstrukcí jazyka JavaScript, při kterém je původní implementace zapouzdřena v tzv. oblasti s lokální platností proměnných a není již přístupná mimo tento prostor (nelze ji tedy žádným způsobem volat zvenku tohoto prostoru). Jakýkoli kód v načítané webové stránce musí využívat uživatelem definovaná pravidla pro využívání původní implementace zapouzdřených volání. Tento princip obalení a zapouzdření je v této diplomové práci demonstrován v rámci implementace obalení a zapouzdření pěti vybraných JavaScriptových konstrukcí včetně demonstrace možné konfigurace chování zapouzdření.

Implementovaný doplněk dále nabízí prostor pro další vývoj a rozšíření jeho současné funkcionality. Za použití přístupu redefinování a obalování je možné zapouzdřit celou řadu dalších funkcí a objektů jazyka JavaScript a dále omezovat, filtrovat, upravovat a analyzovat dynamické chování na webové stránce realizované prostřednictvím technologie JavaScript. Bližší informace o dalších možnostech vývoje tohoto doplňku jsou k popsány v sekci 8.1.

8.1 Další možnosti rozšíření

- **Obalení funkce Function.prototype.call()** – při aplikaci principu obalení a zapouzdření na tuto funkci se zde nabízí možnost filtrovat, upravovat a analyzovat všechny volání funkce call(). Následně je možné navrhnout a implementovat rozšíření současné verze implementovaného doplňku o funkcionalitu notifikace a blokování nežádoucích volání této funkce.

- **Obalení funkce Function.prototype.apply()** – obdobně jako u funkce Function.prototype.call() je zde možné tuto funkci obalit a dále analyzovat a fitrovat její volání.

- **Obalení funkce Function.prototype.bind()** – tak jako u funkcí Function.prototype.call() a Function.prototype.apply() i zde se nabízí možnost tuto funkci obalit a dále filtrovat a analyzovat volání této funkce.

- **Rozšíření funkcionality pracující s HTML elementy typu Canvas** – rozšíření funkcionality doplňku by přidalo jemnější rozhodování, kdy bude zápis do elementu typu HTML Canvas proveden, kdy naopak zablokován a kdy dojde k dotázání se uživatele. Dále je zde prostor pro návrh řešení situace, kdy se na stránce vyskytuje více elementů typu HTML Canvas nebo kdy jsou tyto elementy dynamicky vytvářeny a rušeny. Případně by bylo možné vytvořit podporu pro filtrování různých operací zápisu a čtení z HTML elementu typu Canvas. Nabízí se možnost klasifikovat různé operace zápisu a čtení na ty zcela bezpečné (nehrozí zneužití typu tvorba otisků grafické karty) a ty s potenciálem zneužití.

- **Rozšíření funkcionality snížení přesnosti GPS dat** – u JavaScriptem poskytovaných GPS dat je možnost vedle snížení přesnosti těchto polohových dat přidat podporu pro vkládání souřadnic a dalších dat s vysokou přesností, avšak reprezentující informace, které neodpovídají realitě. Příjemce takových dat by pravděpodobně nepochyboval o jejich správnosti ani přesnosti a skutečná poloha uživatele by zůstala ochráněna.

- **Rozšíření funkcionality pracující s dotazy typu XMLHttpRequest** – rozšíření by spočívalo v jemnějším způsobu rozhodování a posuzování, které dotazy typu XMLHttpRequest budou blokovány, které budou povoleny a kdy je vhodné se dotázat uživatele na souhlas s odesláním dotazu.

- **V případě rozšiřování a úpravy funkcionality vylepšený návrh rozhraní a nastavení doplňku** – v případě implementace nových funkcí by vylepšení uživatelského rozhraní doplňku (stránky s nastavením, interakce s uživatelem) pomohlo zpřehlednit a zjednodušit používání doplňku. Nejen přehlednější rozhraní, ale především odebrání zbytečné konfigurace a usnadnění uživateli ovládat a používat doplněk naplno.

- **Zveřejnění doplňku a umožnění podílet se na dalším vývoji** – doplněk by mohl být zveřejněný mezi ostatními doplňky prohlížečů, aby jej mohli začít používat uživatelé. Z tohoto zdroje by se také dala získávat a analyzovat zpětná vazba od uživatelů, která by pomohla v dalším vývoji doplňku, přidání chybějící funkcionality a zlepšení ergonomie

používání. Nabízí se také možnost zveřejnit zdrojové kódy prostřednictvím veřejného repozitáře (například prostřednictvím služby www.github.com) a umožnit tak dalším vývojářům (popřípadě komunitám) rozšiřovat funkcionalitu.

- **Implementace filtrování úderů do klávesnice** – Údery do klávesnice je také možné monitorovat pomocí JavaScriptu, nabízí se možnost implementovaný doplněk rozšířit o detekci a upozornění na monitorování v podezřelých situacích, dále možnost zablokovat monitorování a povolit jej jen v případě, kdy je v zájmu uživatele nechat údery do klávesnice zpracovávat JavaScriptem.

- **Implementace filtrování sbírání dat o pohybu myši** – pomocí technologie JavaScript je možné implementovat monitorování pohybu myši po obrazovce. Při pohybu myši po obrazovce se dá poloha kursoru zjistit z objektu události. Aplikací principu obalení bychom mohli na vybraných webových stránkách zamezit sbírání informací o pohybu kurzoru, nebo bychom mohli poskytovat infromace o souřadnicích, které neodpovídají realitě.

- **Kompletní detekce a zablokování vybraných session replay služeb** – celá řada služeb využívá pro monitorování uživatelů techniku session replay, jejíž funkcionalita monitorování a sbírání informací je postavena na JavaScriptu. Vedle již zmíněného filtrování a blokování konkrétních JavaScriptových funkcí využívaných ke sběru dat je možné detekovat přítomnost konkrétní monitorovacích služeb a zamezit inicializaci a spuštění samotného monitorovacího kódu. Toto rozšíření by pomohlo výrazně zvýšit soukromí uživatelů internetu, způsob jeho fungování by však nebyl založen na využívaní principu obalení a zapouzdření JavaScriptových konstrukcí, ale spíše na analýze aktuálního inicializačního kódu jednotlivých služeb.

- **Vytvoření detailního analytického nástroje** – vedle doplňku, který se specializuje na zvýšení anonymity a soukromí uživatelů internetu by bylo možné vytvořit analytický nástroj, který by se místo na filtrování, blokování a snižování přesnosti dat poskytovaných a zpracovávaných JavaScriptem soustředil především na tvorbu statistik. Nástroj by umožňoval analyzovat, vyhodnocovat a poměřovat volání JavaScriptu (jejich četnost, parametry, pořadí atd.) a na základě výsledků by bylo možné pro určitou stránku stanovit, které JavaScriptové nástroje a jaké chování stránky slouží pouze k nutnému zajištění funckionality dané stránky a které naopak představuje nežádoucí chování.

- **Vytvoření aktualizované databáze konfigurace doplňku pro jednotlivé webové stránky** – databáze by obsahovala doporučenou konfiguraci blokování, filtrování a povolení jednotlivých JavaScriptových operací pro řadu webových stránek. Databází doporučená konfigurace by místo uživatele nastavovala doplněk a tím by došlo ke zvýšení automatického fungování doplňku a zvýšení komfortu užívání. Databáze by se postupem času a vlivem působení komunitního vývoje neustále rozšiřovala a byla by udržována v aktuálním stavu. Pro stránky bez doporučené konfigurace by doplněk použil heuristiku nebo interakci s uživatelem pro zajištění své funkcionality.

Literatura

[1] Kaplas, J. (2016). Possibilities and usability of various privacy preservation browser add-ons. Bakalářská práce, Lappeenranta University of Technology, Finsko, dostupné online http://www.doria.fi/handle/10024/129993

[2] Ikram, M., Asghar, H., Kaafar, M., et al. 2016. Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning. Proceedings on Privacy Enhancing Technologies, 2017(1), str. 79-99.

[3] Yu, Z.,, Macbeth, S., Modi, K., a Pujol, J.M. 2016. Tracking the Trackers. Proceedings of the 25th International Conference on World Wide Web (WWW '16), str. 121-132.

[4] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. Remote physical device fingerprinting. IEEE Transactions on Dependable and Secure Computing, 2(2): 93–108, 2005. ISSN 1545-5971.

[5] Polčák Libor a Franková Barbora. Clock-Skew-Based Computer Identification: Traps and Pitfalls. Journal of Universal Computer Science. 2015, roč. 21, č. 9, s. 1210-1233. ISSN 0948-6968.

[6] Polčák Libor, Jirásek Jakub a Matoušek Petr. Comment on "Remote Physical Device Fingerprinting". IEEE Transactions on Dependable and Secure Computing. Los Alamitos: IEEE Computer Society, 2014, roč. 11, č. 5, s. 494-496. ISSN 1545-5971.

[7] Polčák, Libor. Zákonné odposlechy: detekce identity. Brno, 2017. Dostupné z: http://www.fit.vutbr.cz/study/DP/PD.php?id=679. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2017-10-13. Vedoucí práce Švéda Miroslav.

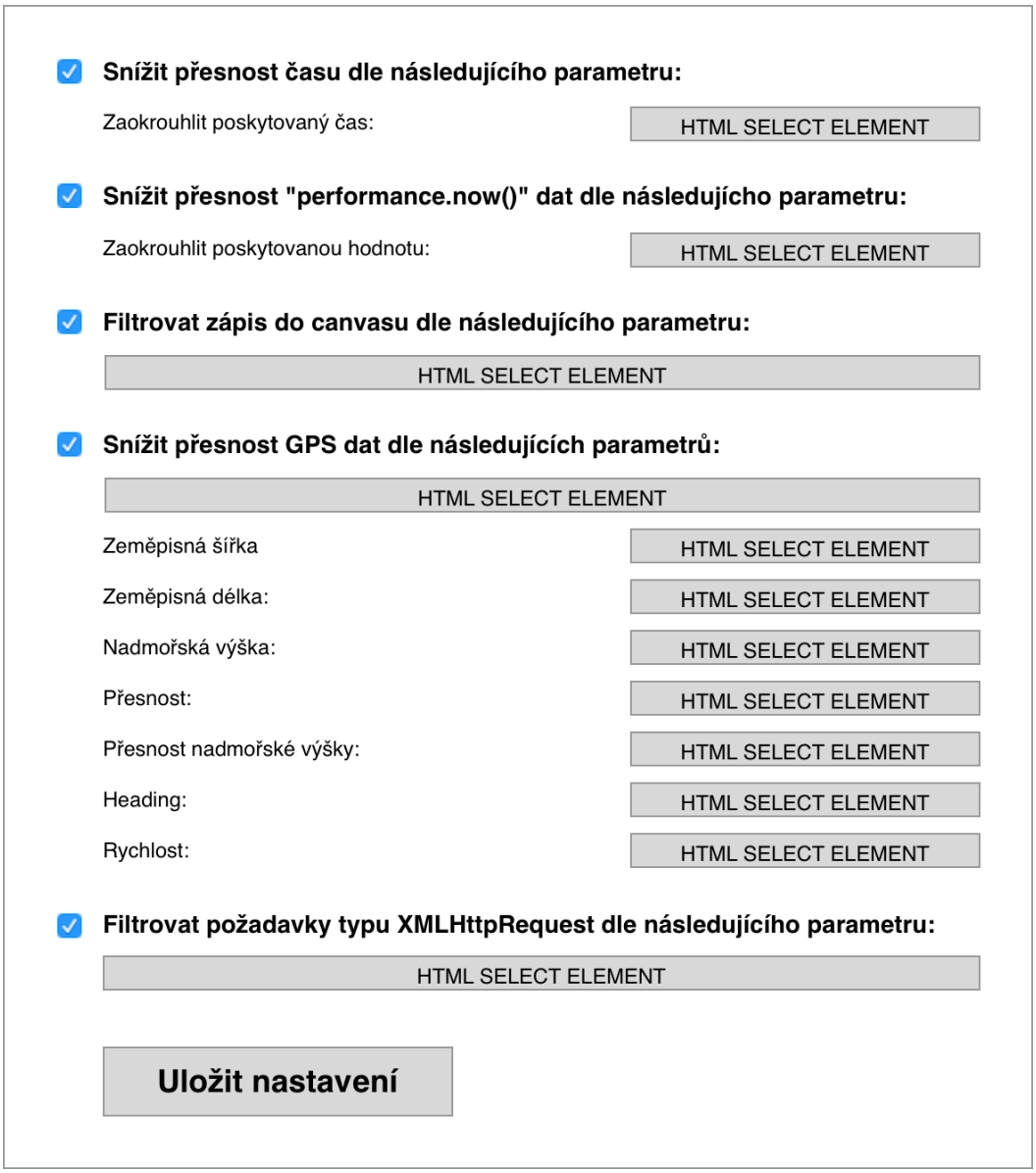
[8] Franková, Barbora. Určování identity počítače pomocí odchylky vnitřních hodin. Brno, 2013. Dostupné z: http://www.fit.vutbr.cz/study/DP/BP.php?id=14704. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2013-06-12. Vedoucí práce Polčák Libor.

[9] Jirásek, Jakub. Využití časových informací pro identifikaci počítače. Brno, 2012. Dostupné z: http://www.fit.vutbr.cz/study/DP/DP.php?id=14040. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2012-06-15. Vedoucí práce Polčák Libor.

Přílohy

Příloha A

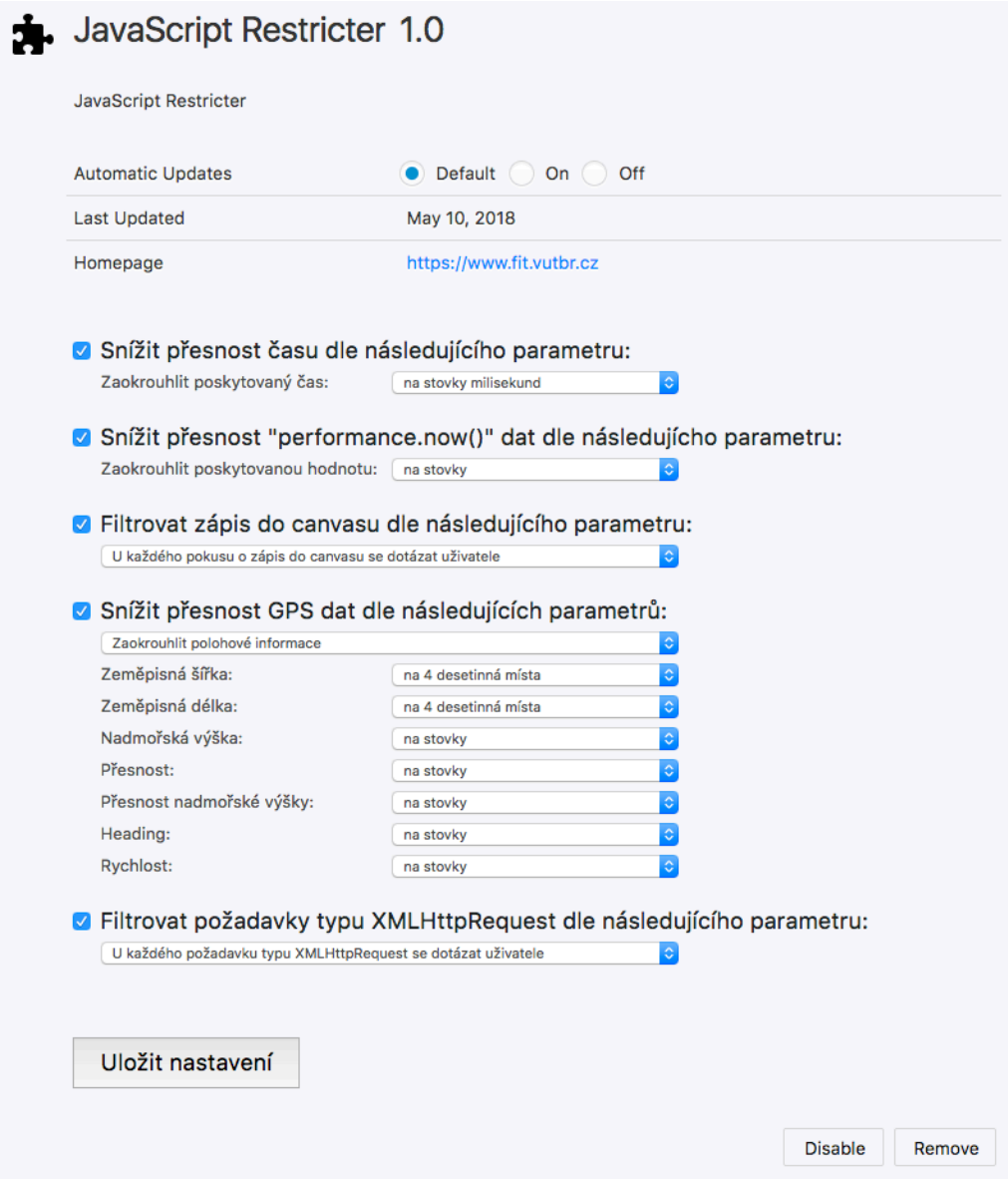
Návrh stránky s nastavením



Obrázek A.1: Celý návrh obsahu stránky s nastavením.

Příloha B

Vzhled stránka s nastavením



Obrázek B.1: skutečný vzhled stránky s nastavením implementovaného doplňku.

Příloha C

Kompletní kód JSON struktury reprezentující nastavení doplňku

```
extension_settings_data: {
  window_date: {
    main_checkbox: value,
    time_round_precision: value
  },
  window_performance_now: {
    main_checkbox: value,
    value_round_precision: value
  },
  window_html_canvas_element: {
    main_checkbox: value,
    type_of_restriction: value
  },
  navigator_geolocation: {
    main_checkbox: value,
    type_of_restriction: value,
    gps_a: value,
    gps_b: value,
    gps_c: value,
    gps_d: value,
    gps_e: value,
    gps_f: value,
    gps_g: value
  },
  window_xmlhttprequest: {
    main_checkbox: value,
    type_of_restriction: value
  }
}
```

Kód C.1: Struktura ve formátu JSON popisující reprezentaci dat nastavení implementovaného doplňku v uložišti technologie WebExtensions.

Příloha D

Kompletní kód pro obalení funkce geolocation.getCurrentPosition()

```
    (
        function() {
            var originalGetCurrentPositionFunction =
navigator.geolocation.getCurrentPosition;
            navigator.geolocation.getCurrentPosition = function(functionName) {
                originalGetCurrentPositionFunction.call(navigator.geolocation,
processOriginalGPSDataObject);
                function processOriginalGPSDataObject(originalPositionObject) {
                    var newLatitude = 0;
                    var newLongitude = 0;
                    var newAltitude = 0;
                    var newAccuracy = 0;
                    var newAltitudeAccuracy = 0;
                    var newHeading = 0;
                    var newSpeed = 0;
                    var newTimestamp = 0;
                    if (originalPositionObject.coords.latitude != null &&
originalPositionObject.coords.latitude != Infinity &&
originalPositionObject.coords.latitude >= 0) {
                        newLatitude =
roundToPrecision(originalPositionObject.coords.latitude, 4);
                    }
                    if (originalPositionObject.coords.longitude != null &&
originalPositionObject.coords.longitude != Infinity &&
originalPositionObject.coords.longitude >= 0) {
                        newLongitude =
roundToPrecision(originalPositionObject.coords.longitude, 4);
                    }
```

```
                if (originalPositionObject.coords.altitude != null &&
originalPositionObject.coords.altitude != Infinity &&
originalPositionObject.coords.altitude >= 0) {
                    newAltitude =
roundToPrecision(originalPositionObject.coords.altitude, -2);
                }
                if (originalPositionObject.coords.accuracy != null &&
originalPositionObject.coords.accuracy != Infinity &&
originalPositionObject.coords.accuracy >= 0) {
                    newAccuracy =
roundToPrecision(originalPositionObject.coords.accuracy, -1);
                }
                if (originalPositionObject.coords.altitudeAccuracy != null &&
originalPositionObject.coords.altitudeAccuracy != Infinity &&
originalPositionObject.coords.altitudeAccuracy >= 0) {
                    newAltitudeAccuracy =
roundToPrecision(originalPositionObject.coords.altitudeAccuracy, -1);
                }
                if (originalPositionObject.coords.heading != null &&
originalPositionObject.coords.heading != Infinity &&
originalPositionObject.coords.heading >= 0) {
                    newHeading =
roundToPrecision(originalPositionObject.coords.heading, -1);
                }
                if (originalPositionObject.coords.speed != null &&
originalPositionObject.coords.speed != Infinity && originalPositionObject.coords.speed
>= 0) {
                    newSpeed = roundToPrecision(originalPositionObject.coords.speed, -
1);
                }
                if (originalPositionObject.timestamp != null &&
originalPositionObject.timestamp != Infinity && originalPositionObject.timestamp >=
0) {
                    newTimestamp = roundToPrecision(originalPositionObject.timestamp,
-3);
                }
                const editedPositionObject = {
                    coords: {
                        latitude: newLatitude,
                        longitude: newLongitude,
                        altitude: newAltitude,
                        accuracy: newAccuracy,
                        altitudeAccuracy: newAltitudeAccuracy,
                        heading: newHeading,
                        speed: newSpeed,
                        __proto__: originalPositionObject.coords.__proto__
```

```
                    },
                    timestamp: newTimestamp,
                    __proto__: originalPositionObject.__proto__
                };
                functionName.call(this, editedPositionObject);
                return true;
            }
            return undefined;
        };
        function roundToPrecision(numberToRound, precision) {
            var moveDecimalDot = Math.pow(10, precision);
            return Math.round(numberToRound * moveDecimalDot) /
moveDecimalDot;
        }
    }
) ();
```

Kód D.1: Kód zajišťující redefinici a obalení funkce navigator.geolocation.getCurrentPosition().

Příloha E

Kompletní kód pro obalení objektu window.XMLHttpRequest

Kompletní kód redefinování objektu window.XMLHttpRequest a obalení jeho původní implementace je zachycen v kódu E.1

```
    (
        function() {
            var blockEverything = false;
            var original_XMLHttpRequest = window.XMLHttpRequest;
            window.XMLHttpRequest = function() {
                var origRequestObj = new original_XMLHttpRequest();
                var original_XMLHttpRequest_open = origRequestObj.open;
                origRequestObj.open = function(requestMethod, requestURL,
requestParameterAsync, requestUsername, requestPassword) {
                    if (blockEverything) {
                        return undefined;
                    }
                    var confirmed = confirm('There is a XMLHttpRequest on URL \"' +
requestURL + '\". Do you want to continue?');
                    if (!confirmed) {
                        return undefined;
                    }
                    if (requestParameterAsync == undefined) {
                        return original_XMLHttpRequest_open.call(origRequestObj,
requestMethod, requestURL);
                    }
                    else if (requestUsername == undefined) {
                        return original_XMLHttpRequest_open.call(origRequestObj,
requestMethod, requestURL, requestParameterAsync);
                    }
                    else if (requestPassword == undefined) {
                        return original_XMLHttpRequest_open.call(origRequestObj,
requestMethod, requestURL, requestParameterAsync, requestUsername);
```

```
                }
                else {
                    return original_XMLHttpRequest_open.call(origRequestObj,
requestMethod, requestURL, requestParameterAsync, requestUsername,
requestPassword);
                }
            };
            return origRequestObj;
        };
    }
) ();
```

Kód E.1: Kód zajišťující redefinici a obalení objektu window.XMLHttpRequest.

Příloha F

Obsah DVD

DVD, které je přiloženo k této diplomové práci, obsahuje samotný doplněk implementovaný v technologii WebExtensions a také testovací ukázkový příklad, na kterém je možné otestovat funkcionalitu implementovaného doplňku. Dále jsou na DVD dostupné podkladové materiály této diplomové práce a soubor README.txt, který obsahuje informace o obsahu média. Detailnější popis testovacího ukázkového příkladu je uveden v sekci 7.2. Obsah přiloženého DVD má tuto strukturu:

```
/
|— demo_example/ ……testovací příklad demonstrující funkcionalitu doplňku
|    |— index.html ……....spouštěcí soubor demonstračního příkladu
|    |— js/ ……..…….....složka obsahující JavaScriptové soubory
|    |    |— canvas.js …… JS kód demonstrující zápis do Canvasu
|    |    |— date.js ……..... JS kód demonstrující práci s objektem window.Date
|    |    |— gps.js ……. JS kód demonstrující práci s polohovými údaji
|    |    |— performance.now.js …. JS kód demonstrující práci s funkcí performance.now()
|    |    |— xmlhttprequest.js ..... JS kód demonstrující práci s XMLHttpRequest dotazy
|    |— style/ ……..……složka obsahující soubory se styly
|    |    |— global.css ……...…..... CSS soubor obsahující veškeré styly
|    |— img/ ……..…… složka obsahující obrázky
|    |    |— demo_image.png ……..... obrázek
|    |— others/ …… složka obsahující ostatní soubory
|    |    |— xmlhttprequest_test_file.html ……..….. testovací soubor pro XMLHttpRequest
|— implemented_extension/ ……..…..…..…..zdrojové kódy implementovaného doplňku
|    |— background.js …….. soubor s JS scripty spouštěný na pozadí
|    |— document_start.js ……..…….. soubor s implementací obalení JS konstrukcí
|    |— manifest.json …….. vstupní bod implementovaného doplňku
|    |— options.css …… soubor s CSS styly určující formát stránky s nastavením
|    |— options.html …… HTML soubor definující stránku s nastavením doplňku
|    |— options.js ……..... JS soubor s kódem potřebným pro realizaci nastavení
|    |— README.md ……...……..... popis doplňku
|— src/ ……..……..…..….. zdrojové kódy této diplomové práce
|— README.rtf ….……popis obsahu DVD a návod na zprovoznění doplňku
```

Příloha G

Zprovoznění implementovaného doplňku

Program je implementován jako doplněk internetového prohlížeče, v této příloze je popsáno zprovoznění doplňku v prohlížeči Mozilla Firefox ve verzi 60.0.

Postup zprovoznění implementovaného doplňku je následující:

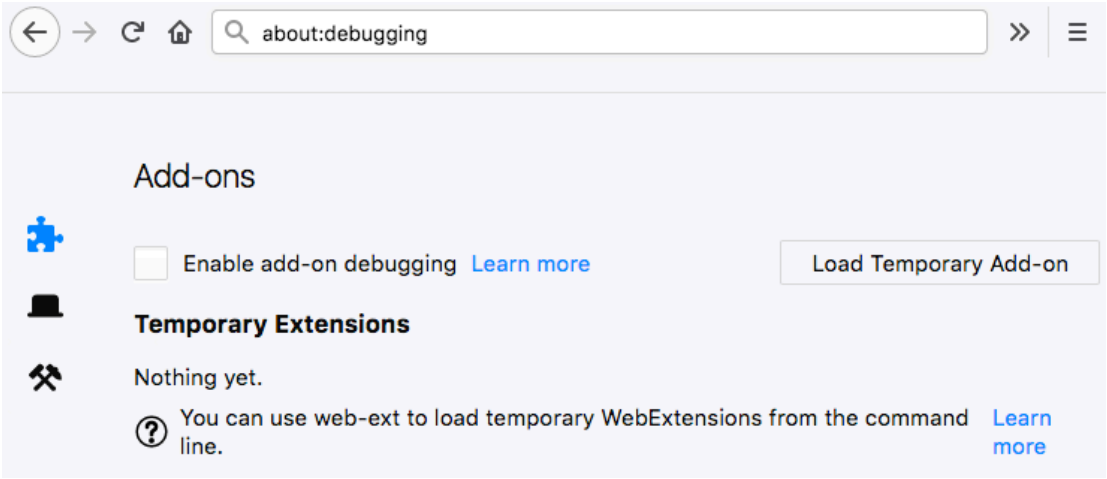
1) Otevřete prohlížeč Mozilla Firefox

2) Přejděte na stránku „about:debugging“ (prostřednictvím zadáním výrazu „about:debugging“ do adresního řádku prohlížeče), zobrazí se okno podobné tomu, jehož výřez je vyobrazen na obrázku G.1

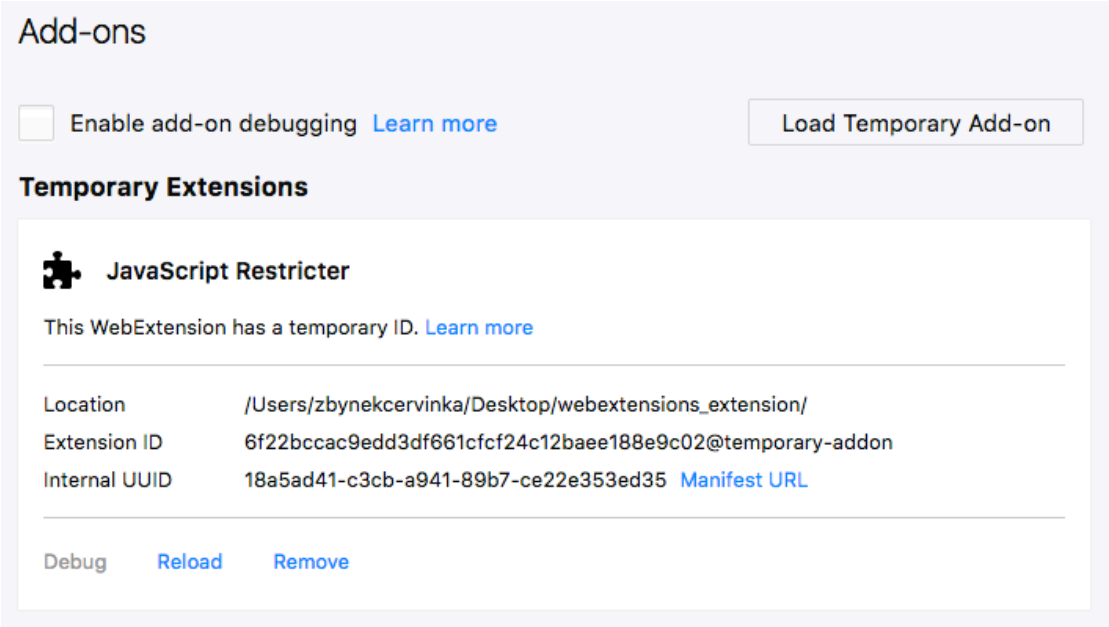
3) V rámci tohoto okna klikněte na tlačítko „Load Temporary Add-on“

4) Vyhledejte soubor manifest.json doplňku, který chcete nainstalovat (doplněk implementovaný v rámci této diplomové práce je umístěn na DVD přiloženém k této diplomové práci, jeho soubor manifest.json se nachází na DVD na adrese */implemented_extension/manifest.json*)

5) Doplněk je nainstalovaný a jeho položka je nyní přítomná mezi položkami ostatních instalovaných rozšíření, obdobně jak je vyobrazeno na obrázku G.2



Obrázek G.1: Výřez obsahu okna webového prohlížeče Mozilla Firefox na adrese „about:debugging“.



Obrázek G.2: Výřez obsahu okna webového prohlížeče Mozilla Firefox na adrese „about:debugging“ po přidání implementovaného doplňku.