BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

JavaScript Restrictor - webové rozšírenie pre obmedzenie volania JavaScriptu

Martin Timko*

Abstrakt

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

Kľúčové slová: Ochrana súkromia a bezpečnosť užívateľov internetu — JavaScript Restrictor — JavaScript Zero — rozšírenie webového prehliadača — JavaScript — WebExtensions

Priložené materiály: Zdrojové kódy rozšírenia

*xtimko00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Bezpečnosť a súkromie užívateľov pri navštevovaní webových stránok sa dnes stále viac a viac narúša. Až 95% webových stránok využíva dnes JavaScript [1]. Pomocou JavaScriptu je však možné zberať o užívateľoch rôzne osobné a citlivé informácie aj bez vedomia užívateľa, ktoré môžu viesť až k nevyžiadanej identifikácii osôb [2] [3], alebo sa môžu voči nim vykonávať útoky [4]. Je preto potrebné zvýšiť anonymitu a ochranu užívateľov pred zberom dát a útokmi. Avšak pri úplnom zablokovaní JavaScriptu často dochádza k znefunkčeniu webových stránok, čo môže užívateľ vnímať negatívne. Je potrebné hľadať iné cesty, ako blokovanie JavaScriptu, aby užívateľ nemusel obetovať funkčnosť webových stránok na úkor bezpečnosti.

Cieľom tejto práce bolo naštudovať problematiku webových rozšírení, ako aj možnosti vylepšenia bezpečnosti a ochrany osobných údajov používateľov pomocou webových rozšírení. Následne vylepšiť prototyp webového rozšírenia, ktoré vytvoril minulý rok v rámci svojej diplomovej práce Ing. Zbyněk Červinka [5]. Vytvárané rozšírenie bude mať za úlohu chrániť užívateľov pred zberom osobných dát, ako aj zabrániť istým druhom útokov cez JavaScript. Toto rozšírenie zverejniť na obchodoch s rozšíreniami pre príslušné webové prehliadače. Cieľom je, aby si mnou vytvorené rozšírenie mohol ktokoľvek stiahnuť a používať.

Rozšírenie využíva princíp obaľovania JavaScriptových funkcií a objektov (podrobnejšie vysvetlené v sekcii 2.1.2). Vďaka tomuto princípu je možné zvýšiť anonymitu prehliadania webu a zároveň znemožniť niektoré útoky pomocou JavaScriptu na používateľoch.

2. Rozšírenia pre blokovanie JS API

V rámci mojej práce som analyzoval rozšírenia, ktorých cieľom je blokovať niektoré API jazyka JavaScript. Ďalšou nosnou prácou, ktorú som analyzoval okrem práce Zbyňka Červinky [5], je aj práca JavaScript Zero [4].

2.1 JavaScript Zero

JavaScript Zero [4] obsahuje súbor opatrení pre zabránenie útočenia cez JavaScript, ktorý útočníci využívajú pre útoky pomocou škodlivých webových stránok. Demonštráciou tohto modelu je webové rozšírenie nazvané Chrome Zero[1], ktoré autori implementovali. JavaScript Zero má vo svojej podstate podobný cieľ ako moja práca. Teda predstavuje akýsi firewall medzi webovou stránkou a užívateľom a snaží sa zvýšiť bezpečnosť užívateľa.

2.1.1 Analýza útokov

Autori analyzovali mikroarchitekturálne útoky a útoky cez postranné kanály vykonávané pomocou JavaScriptu a práca ukázala, že takmer každý útok cez JavaScript využíva presné načasovanie (anglicky *accurate timing*). Pomocou objektov `Date` a `Performance` v JavaScripte je možné získať časové rozdiely napríklad do zásahov pamäte DRAM či cache s vysokou presnosťou, čím sa tieto objekty môžu využiť pri útokoch a odhaliť utajované informácie [6] alebo identifikovať zariadenie [7].

Ďalej je možné pri útokoch využiť pamäťové adresy pre útoky zamerané na DRAM [8] či viacvláknové procesy a zdieľanie dát. Príkladom je identifikácia užívateľovho vstupu alebo zistenie načítaných webových stránok užívateľa [9]. Rovnako je možné využiť pre útoky aj senzorové API. Ako článok autorov Mehrnezhad a kol. [2] ukázal, pomocou senzorov pre zaznamenávanie pohybov a orientácie bolo možné odvodiť PIN kód obete útoku.

2.1.2 Chrome Zero

Webové rozšírenie Chrome Zero bolo vyvíjané pre webový prehliadač Google Chrome. Tvorcovia chceli vytvoriť rozšírenie, ktoré by pri aplikovaní protiopatrení vzdorovalo a nedalo sa obísť pri záškodnej činnosti, zároveň ale aby toto rozšírenie nemalo markantný vplyv pre užívateľskú spokojnosť pri prehliadaní webu.

Chrome Zero obsahuje 5 úrovní ochrany. Podľa zvolenej úrovne zabezpečenia sa zvolí daná akcia (povoliť pôvodnú funkciu, úplne blokovať funkciu, modifikovať hodnoty, zobraziť povolenie užívateľovi pre odsúhlasenie akcie) pre danú volanú funkciu v skripte (napr. pri volaní `performance.now`).

[1] https://github.com/IAIK/ChromeZero

Hlavnou myšlienkou implementácie je využiť možnosti JavaScriptu pre redefinovanie a teda pre obaľovanie alebo zapuzdrenie pôvodnej implementácie. Týmto spôsobom sa tak funkcia vyjme z globálneho priestoru platnosti a je dostupná iba v obaľovacej funkcii. Takto naberá iba lokálnu platnosť a nie je možné sa k nej dostať z iných častí kódu. Nasledujúci kód predstavuje ukážku zapuzdrenia funkcie `performance.now` so znížením presnosti merania času.

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

Obalená funkcia tak vracia upravené hodnoty a nedovolí tak útočníkovi získať požadované dáta s vysokou presnosťou, čím vie znemožniť útočníkovi presné načasovanie.

Autori napríklad ďalej softvérovo upravili prístupové doby do cache, čím útočník nemôže vyčítať, kedy nastal cache hit alebo cache miss pri útokoch na pamäť cache.

2.1.3 Zhodnotenie JavaScript Zero

Autori práce vytvorili rozšírenie, ktoré chráni užívateľov od rôznych útokov postrannými kanálmi, ako aj pred mikroarchitekturálnymi útokmi. Pri testovaní sa ukázalo, že užívateľ nevedel rozlíšiť, či je Chrome Zero spustený alebo nie, teda dopad spusteného rozšírenia je pre bežného užívateľa nepostrehnuteľný.

Ponuka piatich úrovní zabezpečenia je vhodná pre ľahké a rýchle nastavenie používateľov rozšírenia Chrome Zero. V rozšírení ale chýba nastavenie povolení osobitne pre jednotlivé weby. Vytvorenie vlastných stupňov zabezpečenia užívateľom s možnosťou vlastného nastavenia pre každú funkciu osobitne taktiež nie je možné. Ďalej Chrome Zero je dostupný iba na prehliadačoch Google Chrome a bol vyvíjaný na Chrome tak, že kód nie je jednoducho prenositeľný na iné prehliadače. Naviac toto rozšírenie nie je ponúkané cez oficiálny obchod rozšírení, inštalácia je o to viac sťažená. Dôvodom je, že Google Chrome sa snaží všetky rozšírenia, ktoré nepochádzajú z oficiálneho obchodu z bezpečnostných dôvodov blokovať. Pri testovaní sa ukázalo, že niektoré obaľované API, ako napríklad `BatteryManager`, boli nefunkčné. Rovnako som pri testovaní natrafil pri niektorých weboch[2] na úplné zamrznutie prehliadača. Skutočná funkčnosť rozšírenia tak ostáva otázna.

[2] Napríklad https://jsfiddle.net/

2.2 Prototyp rozšírenia Zbyňka Červinky

Ako som spomínal v úvode, moja práca nadväzuje na diplomovú prácu Ing. Zbyňka Červinky[5], ktorý vytvoril prototyp rozšírenia pre zvýšenie anonymity a ochrany užívateľov. Rozšírenie sa nesnaží úplne blokovať JavaScript, ale technikou zapuzdrenia JavaScriptového kódu (popísanú v sekcii 2.1.2) využívané funkcie iba modifikuje.

2.2.1 Zapuzrdené konštukcie

Konkrétne prototyp rozšírenia obaľuje nasledujúce konštrukcie jazyka JavaScript:

- Objekt `window.Date`,
- funkciu `window.performance.now()`,
- funkciu `window.HTMLCanvasElement.getContext()`,
- funkcia `navigator.geolocation.getCurrentPosition()`,
- objekt `window.XMLHttpRequest`,

Objekt `Date` a fukcia `performance.now` sa dajú využiť pri útokoch spojeným s presným načasovaním ako bolo popísané v sekcii 2.1.1.

HTML elementy, ktoré sú typu Canvas môžu byť používané ako grafické elementy pre kreslenie grafiky alebo animácií. Ako uvádza článok [3], pomocou týchto elementov je ale možné získať napríklad fingerprint grafických kariet a identifikovať tak užívateľa. Blokovaním týchto elementov alebo upravovanie hodnôt týchto elementov je tak možné zamedziť získaniu špecifických popisov pre konkrétne typy grafických kariet.

Keď že `XMLHttpRequest` žiadosti môžu obsahovať napríklad výsledky úspešných útokov, ktoré si útočník posiela, obalením tohto objektu je možné odchytiť tieto žiadosti.

Funkcia `getCurrentPosition()`, vracia aktuálnu polohu zariadenia. Vhodným zapuzdrením funkcie sa znižuje presnosť dát pre lokalizáciu zariadenia, ktoré užívateľ používa.

Nastavenie jednotlivých obalení cez užívateľské rozhranie je nasledujúce. Pre `Date` je možné modifikovať ich návratové hodnoty a zaokrúhľovať ich ich na stotiny, desatiny a celé sekundy, respektíve zaokrúhlenie na desiatky, stovky či tisícky pri funkcii `performance`. Pre HTML elementy typu Canvas a XMLHttpRequest je možné nastaviť úplné blokovanie, alebo dopytovanie užívateľa pri každom pokuse o zápis do Canvasu alebo pokuse o odoslanie XHR žiadosti. Pri geolokačných dátach sa dajú zaokrúhľovať súradnice zariadenia od 0 až po 5 desatinných miest. Pre ďalšie hodnoty ako rýchlosť a smer zariadenia platí to isté ako pri `performance`.

Keď že k invokácii tohto zapuzdreného JavaScriptového kódu dochádza v dobe, kedy ešte prehliadač nezačal spracovávať zdrojový kód načítavajúcej sa webovej stránky, žiaden kód okrem nášho nebude mať prístup k pôvodnej implementácii [5].

Prototyp bol testovaný ako na ukážkovom webe, tak aj na reálnych stránkach, na ktorých bola preukázaná funkčnosť rozšírenia. Nevýhodou ale môže byť stránka s nastavením. Keď že je potrebné pre každú zapuzdrenú funkciu ručne prejsť nastavenie a vybrať si jednu z možností, to je pre bežných užívateľov, ktorí nie sú detailne oboznámení s touto problematikou, dosť problémové. Chýba teda nejaké jednoduchšie nastavenie pre bežného používateľa a rozšírenie je vo forme prototypu. Preto nie je dostupné na oficiálnych stránkach rozšírení a bolo vyvíjané iba pre Mozillu Firefox.

3. Návrh rozšírenia

V tejto sekcii sa pobavíme o návrhu môjho webového rozšírenia, ktoré som nazval JavaScript Restrictor alebo skrátene JSR.

3.1 Návrh funkčnosti

Princíp fungovania doplnku bude zachovaný z prototypu. Zapuzdrené a obalené budú teda funkcie a objekty jazyka JavaScript, ktoré boli popísané v sekcii 2.2. Ďalej sa naviac rozšírenie rozrastie o ďalšie obalené alebo upravované hodnoty. Ide o informácie, ktoré identifikujú webový prehliadač, operačný systém alebo hardvér užívateľa. Menovite sa rozrastie rozšírenie o tieto API objektu `Navigator`, ktoré budú podľa zvolenej úrovne aplikované: `userAgent`, `vendor` čím dokážeme podvrhnúť prehliadač. `platform`, ako aj `appVersion`, a `oscpu`, čím vieme podvrhnúť operačný systém užívateľa. `language`, `languages` nám umožňujú podvrhnúť predvolené jazyky užívateľa, `doNotTrack` indikuje, že užívateľ nechce byť sledovaný. Nakoniec cez `hardwareConcurrency` spolu s `deviceMemory` vieme podvrhnúť hardware užívateľa. Simultánne sa budú spoločne s týmito API modifikovať aj niektoré hlavičky HTTP webRequest. Napríklad Accept-Language alebo User-Agent. Rozšírenie bude vedieť podvrhnúť aj hodnotu Referer hlavičky pre webRequest a `document.referrer`, aby si navštevované stránky mysleli, že k ním pristupujeme priamo a nie cez hyperlinkový odkaz z inej stránky alebo podstránky. Týmto, vieme zvýšiť anonymitu užívateľa, pretože ak sa dané hodnoty operačného systému alebo prehliadača podvrhnú na najpopulárnejšie, ktoré sa využívajú, tým sa užívateľ stane jedným z mnohých podobných, či rovnakých užívateľov.

V rozšírení sa upraví chápanie Canvas elementov oproti prototypu. Keď že pri fingerprintingu pomocou elementov Canvas [3] sa využívajú iné funkcie ako `getContext()`, napr. `HTMLCanvasElement.toDataURL()` alebo `getImageData()`, budú sa tak zamedzovať tieto funkcie. Týmto sa zaručí správne vykresľovanie Canvas elementov na stránkach, ktoré tieto elementy používajú, no zároveň bude rozšírenie chrániť užívateľa pred získaním fingerprintu.

Ďalším rozdielom oproti prototypu je, že rozšírenie nebude ponúkať iba manuálne nastavenie úrovne ochrany užívateľa pred útočníkom či zberom osobných dát.

Používateľ bude mať na výber jednu z troch prednastavených úrovní ochrany. Ďalej bude môcť vybrať aj vlastné nastavenie, ktoré si bude vedieť manuálne nastaviť, presne tak ako v prototype rozšírenia. Nebude chýbať možnosť vypnúť funkcionalitu rozšírenia. Spolu teda bude päť úrovní ochrany užívateľa. Nultá alebo nula, teda žiadna, kedy bude funkcionalita vypnutá. Potom úrovne od 1 do 3, kedy sa bude postupne aplikovať stále väčšie a väčšie modifikovanie a teda zaokrúhľovanie dát funkcií a objektov. Pri maximálnej úrovni 3 bude nastavené maximálne veľké zaokrúhľovanie hodnôt. Napríklad geolokačné dáta úplne nulované alebo budú podvrhnuté informácie o používanom prehliadači. Ako piatu úroveň bude možné nastaviť aj vlastnú úroveň. Teda budú zachované možnosti rozšírenia z prototypu. Jedine tam bude možné nastaviť dopytovanie užívateľa alebo úplné blokovanie XHR žiadostí. Pretože ohľadom týchto žiadostí by bolo vhodné vytvoriť celú heuristiku, ktoré žiadosti blokovať a ktoré nie, a teda v tomto rozšírení to bude dostupné iba pre vyskúšanie si funkčnosti zo strany užívateľov.

Aby sa zvýšila flexibilita rozšírenia, nastavovanie jednotlivých úrovní bude možné aj pre konkrétne domény a subdomény. Bude sa teda môcť vytvárať zoznam domén a k nim priradených úrovní ochrany. Zároveň ale ak nebude nastavená žiadna úroveň pre danú doménu, bude aplikovaná prednastavená úroveň ochrany, ktorú si užívateľ vyberie.

Pre implementáciu rozšírenia JavaScript Restrictor bude použitá technológia WebExtensions[3], ktorá využíva HTML, CSS, JavaScript a rôzne iné potrebné súbory pre rozšírenie, napríklad obrázky.

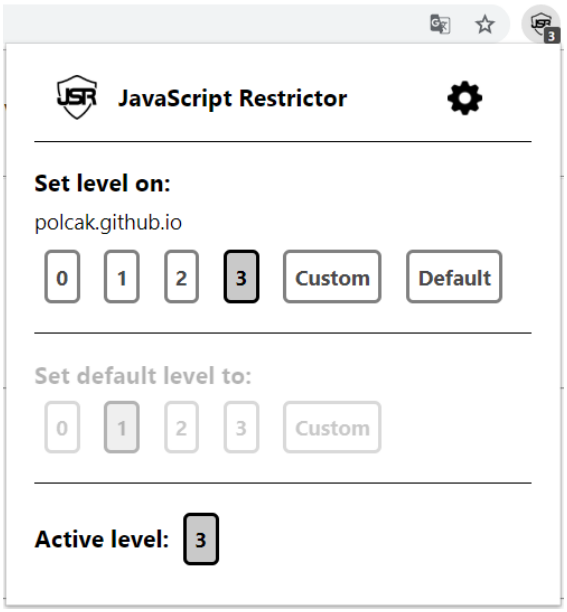
Základná štruktúra súborov rozšírenia ako HTML, CSS a JavaScript súbory bude prevzatá z prototypu a následne sa podľa potrieb budú dané súbory upravovať alebo vytvárať nové.

[3]Viac info na https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

3.2 Návrh užívateľského rozhrania

Návrh užívateľského rozšírenia je dôležitý, aby nebol užívateľ zmätený a ovládanie rozšírenia bolo dostatočne intuitívne. Rozšírenie bude možné nastavovať cez stránku s nastavením alebo cez vyskakovacie okno po kliknutí na ikonku rozšírenia v paneli prehliadača.

Obrázok 1 ukazuje užívateľské rozhranie vyskakovacieho okna po kliknutí na ikonu rozšírenia. Cez neho si užívateľ bude môcť nastaviť úroveň pre aktuálnu doménu, nastaviť predvolenú úroveň v prípade, že nie je vybratá úroveň pre aktuálnu doménu. Rovnako bude možné sa dostať k stránke s nastavením. Pri ikonke rozšírenia v paneli prehliadača bude uvedená aktuálna úroveň ochrany vo forme štítka alebo odznaku (anglicky *badge*).



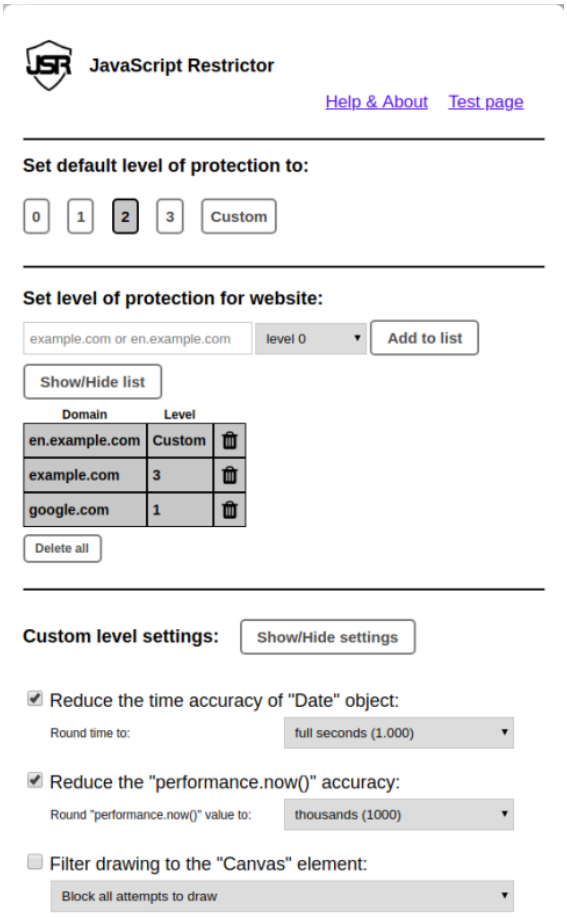
Obrázok 1. Návrh vyskakovacieho okna rozšírenia.

Druhý obrázok 2 nám ukazuje návrh stránky s nastavením, kde bude možné nastaviť predvolenú úroveň, pridať alebo odobrať domény zo zoznamu domén s vybranou úrovňou ochrany. Ďalej sa cez stránku dá nastaviť užívateľská úroveň a pristúpiť k domovskej stránke alebo testovacej stránke, kde si môže užívateľ vyskúšať funkčnosť rozšírenia.

4. Implementácia

V tejto sekcii sa pozrieme, ako bolo rozšírenie vyvíjané pomocou technológie WebExtensions, ktorá ma zaručovať iba minimálne rozdiely v implementácii pre rôzne prehliadače. Ďalej v tejto sekcii spomeniem aj ako bolo rozšírenie publikované na prehliadačoch Mozilla Firefox, Google Chrome a Opera.

Súbor `manifest.json` obsahuje všetky potrebné metadata pre rozšírenie ako meno rozšírenia, špecifikácia skriptov bežiacich pri stránke s nastavením,



Obrázok 2. Návrh stránky s nastavením rozšírenia.

vyskakovacom okne (popup), ale najmä špecifikácia skriptov bežiacich na pozadí webových stránok a pri štarte dokumentov. Chrome a Opera majú rovnaký manifest, Firefox sa líši iba vo formálnych zápisoch niektorých kľúčov manifestu. Ďalej sa rozšírenie delí na súbory background.js, document_start.js, options kódy a popup kódy.

Všetky nastavenia rozšírenia sa ukladajú do pamäte prehliadača s možnosťou synchronizácie medzi zariadeniami v prípade rovnakého účtu užívateľa. Ide o súbor o formáte JSON, ktorý môže slúžiť ako databáza alebo úložisko pre nastavenia.

Súbor backgound.js je skript, ktorý beží nepretržite na pozadí rozšírenia. Obsahuje ale aj akcie pri prvotnom nainštalovaní rozšírenia, teda nastaví sa predvolená úroveň na úroveň ochrany 2 a inicializujú sa nastavenia užívateľskej úrovne ochrany. Následne sa skript stará pri prepínaní kariet prehliadača o prepínanie štítka pri ikonke rozšírenia v paneli prehliadača indikujúceho aktívnu úroveň ochrany.

Skript document_start.js sa stará o funkcionalitu rozšírenia. Teda zistí konkrétnu URL pri načítavaní webovej stránky a na základe toho zisťuje, či sa jedná o doménu, ktorá má špecifické nastavenie ochrany alebo sa má načítať predvolená úroveň ochrany. Následne sa načíta aktuálna úroveň ochrany a začnú sa zapuzdrovať jednotlivé funkcie alebo objekty v kóde s nastaveniami aktuálnej úrovne. Takto sa vložia do webovej stránky všetky obalené funkcie alebo objekty, ktoré vracajú modifikované dáta podľa nastavenia úrovne.

Ďalej skript popup.js, ktorý beží na pozadí vyskakovacieho okna sa stará o interakciu s vyskakovacím oknom. Teda v prípade nastavenia špecifickej úrovne pre danú doménu cez popup, vloží do úložiska rozšírenia novú položku. Ide o dvojicu s doménou a úrovňou k nej priradenej. Pri vrátení úrovne pri doméne na predvolenú úroveň sa položka z úložiska s doménou vymaže. Po zmene nastavenia predvolenej úrovne cez popup sa v úložisku rozšírenia zmení úroveň predvolenej úrovne ochrany. V prípade, že je nastavená úroveň pre koreňovú doménu a nie pre subdoménu, na webe subdomény sa zobrazí v popup okne infomácia, že aktívna úroveň sa načítala na základe koreňovej domény. Všetky zmeny, ktoré sa vykonávajú sa okamžite aplikujú a znázorňujú vhodnou formou pre užívateľa. Teda zmeny sa aplikujú do súboru popup.html, ktorý sa zobrazuje užívateľovi ako vyskakovacie okno.

V podstate je skript options.js, ktorý beží na stránke s nastavením z časti podobný skriptu popup.js. Čo sa týka nastavovaní predvolenej úrovne je funkcionalita rovnaká. Princíp pridávania domén na úložisko prehliadača je rovnaký ako aj mazanie. Rozdielna je forma pridávania. Keď že sa cez stránku nastavenia dá pridať akákoľvek doména zadaná cez textové pole. Stránka s nastavením naviac obsahuje tlačidlo pre úplné vymazanie celého zoznamu domén. Nakoniec sa cez options.js ukladá na úložisko aj nastavenie užívateľskej úrovne (*custom level*). Pri načítaní stránky s nastavením sa prejdú všetky nastavenia uložené v úložisku prehliadača a pomocou pridávaním HTML tried a elementov do options.html sa vytvorí stránka s nastavením, kde je možné vidieť všetky aktuálne hodnoty.

5. Porovnanie s inými rozšíreniami

Potvrdenie fungovania konceptu zapuzdrenia funkcií bolo preukázané v prácach [4] a [5]. V tejto sekcii sa preto budem venovať skôr porovnaniu môjho rozšírenia s Chrome Zero a inými dostupnými rozšíreniami.

5.1 JSR a Chrome Zero

Pre dôvody popísané v sekcii 2.1.3, nebolo JSR spájané s Chrome Zero. Bola by totiž potrebná hlbšia analýza funkčnosti Chrome Zero.

Chrome Zero si dáva za úlohu podobný cieľ ako JSR. Autori Chrome Zero sa zamerali na obalenie a úpravu funkcie `performance.now()`, čo nájdeme

aj v JSR. Ďalej sa zamerali na funkcie spojené s popisom stavu batérie zariadenia a modifikovanie hodnôt získavaných zo senzorov zariadení ako senzor svetla či pohybu. Ďalej sa zamerali na vytváranie náhodných spomalení pri posielaní správ `window.Worker` alebo prístupov do pamäte pre zvýšenie ochrany pred útokmi, čo zatiaľ nie je zapracované do JSR.

Objekty `Date` či geolokačné a funkcie Canvas neobaľujú. V JSR naviac nájdeme obaľovanie a upravovanie hodnôt spojených s informáciou o prehliadači alebo operačného systému, ktoré posielajú prehliadače pre svoju identifikáciu. JSR teda ponúka viac možností pre zamedzenie získavania presných hodnôt o užívateľovi ako Chrome Zero.

5.2 JSR a iné rozšírenia zamerané na ochranu súkromia

Existujúce populárne webové rozšírenia ako Privacy Badger, Adblock Plus alebo napríklad Quick Javascript Switcher je možné stiahnuť z oficiálnych obchodov pre rozšírenia. Z ich testovania vyplýva, že rozšírenia, ktoré automaticky rozhodujú o tom, ktoré prvky sa považujú za nechcené, ako v prípade rozšírenia Privacy Badger, nie vždy rozhodujú správne a blokované prvky môžu znefunkčniť stránku. Časom sa síce očakáva zlepšenie analýzy nechcených sledovacích prvkov, no v tejto chvíli môže byť užívateľ miestami nútený ručne vypínať alebo zapínať ochranné prvky rozšírení. Tým sa narušuje plynulosť prehliadania webu používateľmi rozšírení.

Úplným zablokovaním JavaScriptu v prípade Quick Javascript Switcher sa síce zamedzí funkcia sledovacích prvkov cez JavaScript, ale rovnako sa aj často znefunkčnia webové stránky a niekedy je to až neprijateľné z použiteľného hľadiska pri prezeraní webu. Pri JSR sa snažíme, aby prehliadanie webu bolo bezproblémové a užívateľ nemusel manuálne meniť úrovne ochrany či vypínať rozšírenie. Zároveň, aby užívateľovi plne fungovali navštevované stránky, pričom sa ale zvyšuje úroveň ochrany.

Adblock Plus používa zoznam pravidiel, podľa ktorého sa riadi rozšírenie a vie blokovať nežiadúce sledovacie prvky. Pre blokovanie a filtrovanie nežiadúcich prvkov využíva regulárnych výrazov, na základe ktorých sa dané prvky blokujú. Adblock môže teda blokovať JavaScript, no len na úrovni celého súboru, čo môže viesť k nefunkčnosti častí webových stránok. JSR tento problém rieši blokovaním alebo modifikovaním iba vybraných funkcií či objektov, nie celého súboru.

6. Publikovanie rozšírenia a záver

Pri písaní rozšírenia pomocou Webextensions sa v rámci rozdielnych prehliadačov vieme stretnúť s rozdielnou podporou niektorých volaní API pre dané prehliadače. V našom prípade sa ale nakoniec podarilo kód zjednotiť a teda jediné rozdielne súbory rozšírenia pre rôzne prehliadače sú súbory manifest. JavaScript Restrictor bol publikovaný na oficiálnych stránkach pre rozšírenia a rozšírenie JSR je tak dostupné na stiahnutie a používanie pre prehliadač Mozilla Firefox[4], Google Chrome[5] a pre prehliadač Opera[6].

Zatiaľ sa najviac rozšírenie sťahuje v prípade Opery. Môže to byť spôsobené aj tým, že rozšírení na Operu nie je až tak veľa a ich web s rozšíreniami vie zoradiť rozšírenia od najnovších. Firefox a Chrome radí rozšírenia skôr podľa populárnosti. Teda náhodný užívateľ sa k rozšíreniu dostane podstatne ťažšie. V tejto chvíli sa čaká na spätnú väzbu od užívateľov, aby sme vedeli zapracovať a zmeniť, ak sa v rozšírení nachádzajú nejaké mätúce prvky čomu užívatelia nerozumejú.

Tento článok sa zaoberá problematikou, ako zlepšiť bezpečnosť a zvýšiť anonymitu užívateľov na internete pri prehliadaní webu. V rámci práce bolo navrhnuté a implementované webové rozšírenie JavaScript Restrictor. Pomocou zapuzdrenia a obalenia pôvodnej imlementácie (popísané v sekcii 2.1.2) vybraných JavaScriptových API je tak možné zvýšiť anonymitu a ochranu používateľov. Rozšírenie je dostupné na oficiálnych obchodoch s rozšíreniami prehliadačov Mozilla Firefox, Google Chrome a Opera.

Ďalšia práca a vylepšenie rozšírenia by mohlo spočívať v testovaní a následnom vyladení, ktoré úrovne obsahujú aké zložky ochrany, aby to bolo viac užívateľsky prívetivé. Nepochybne je dnes stále možné nájsť nové možnosti ako užívatelia prichádzajú o svoju anonymitu a postupne tak zavádzať ochranné prvky proti týmto spôsobom identifikácie či útokom voči užívateľom.

Poďakovanie

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

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

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

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

Literatúra

[1] Usage of javascript for websites. https://w3techs.com/technologies/details/cp-javascript/all/all.

[2] Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, and Feng Hao. *TouchSignatures: Identification of User Touch Actions and PINs Based on Mobile Sensor Data via JavaScript*. 2016.

[3] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in html5, 2012.

[4] Michael Schwarz, Moritz Lipp, and Daniel Gruss. *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks*. 2018.

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

[6] Daniel J. Bernstein. *Cache-timing attacks on AES*. 2005.

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

[8] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. *Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript*. 2016.

[9] Pepe Vila and Boris Kopf. *Loophole: Timing Attacks on Shared Event Loops in Chrome*. 2017.