VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ PREZENTACE PROJEKTU JSHELTER

WEB PRESENTATION OF THE JSHELTER PROJECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN KRČMA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN BEDNÁŘ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Krčma Jan**
Program: Informační technologie
Název: **Webová prezentace projektu JShelter**
A Web Presentation of JShelter Project
Kategorie: Web

Zadání:

1. Seznamte se s funkcionalitou rozšíření pro webové prohlížeče JShelter (navazující na rozšíření JavaScript Restrictor). Nastudujte problematiku otisku webového prohlížeče a možnosti jeho zneužití pro sledování uživatelů v kontextu projektu JShelter. Seznamte se s útoky realizovanými v prostředí webu, proti kterým nabízí rozšíření obranu (např. skenování lokální sítě, určování identity počítače pomocí odchylky vnitřních hodin).
2. Navrhněte webovou prezentaci, která bude názorně a srozumitelně demonstrovat funkcionalitu rozšíření JShelter. Demonstrujte zejména, jaký vliv má toto rozšíření na soukromí a bezpečnost koncového uživatele.
3. Návrh webové prezentace pro projekt JShelter implementujte.
4. Otestujte vytvořenou webovou prezentaci v aktuálních verzích různých webových prohlížečů (minimálně Google Chrome a Mozilla Firefox).
5. S využitím webové prezentace ověřte z pozice uživatele, zda JShelter splňuje funkcionalitu deklarovanou ve své dokumentaci. Zhodnoťte zejména, jak JShelter ovlivňuje získávání otisku webového prohlížeče a zda dokáže zabránit útokům, které má za cíl mitigovat.
6. Navrhněte možná pokračování.

Literatura:

- SCHWARZ, Michael, LIPP, Moritz a GRUSS, Daniel. *JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks*. Network and Distributed Systems Security (NDSS) Symposium 2018. ISBN 1-891562-49-5.
- BERGMON, John. *Attacking the Internal Network from the Public Internet Using a Browser as a Proxy*. Forcepoint research report. 2019. Dostupné z: https://www.forcepoint.com/sites/default/files/resources/files/report-attacking-internal-network-en_0.pdf
- 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.

Pro udělení zápočtu za první semestr je požadováno:

- Vývoj aplikace bude probíhat iterativně metodologií *Feature-driven development*. Student bude vedoucímu práce předkládat funkční aplikaci ve 2-týdeních vývojových cyklech, každý bude zaměřen na konkrétní bezpečnostní aspekt webové prezentace. Pro udělení zápočtu je nutné demonstrovat alespoň 5 bezpečnostních aspektů, tedy mít rozpracované body 1-4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bednář Martin, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 13. října 2021

Abstrakt

Tato práce se zaobírá tvorbou uživatelsky přívětivé webové prezentace projektu JShelter, ve které jsou vyzvednuty jeho výhody v ohledu bezpečnosti jeho uživatelů při pohybování se na webu tak, aby byla dostatečně zachována jejich anonymita a zároveň i funkčnost navštěvovaných stránek. Především zde bude představeno zabezpečení poskytované rozšířením JShelter proti sledování uživatele na základě odchylky jeho vnitřních hodin a také utvrzení ochrany same-origin policy.

Abstract

This thesis deals with creating user-friendly web presentation of the project JShelter, in which its advantages in terms of the safety of its users when navigating the World Wide Web are highlighted, so that their anonymity and at the same time the functionality of the visited sites are sufficiently preserved. Above all, the result provided by the JShelter browser extension against tracking the user based on the deviation of his internal clock and also the confirmation of the same-origin policy protection will be shown.

Klíčová slova

rozšíření webového prohlížeče, JShelter, bezpečnost, sledování uživatelů, anonymita, webové útoky, webová prezentace, otisk zařízení

Keywords

web browser extension, JShelter, safety, user tracking, anonymity, web attacks, web presentation, fingerprint

Citace

KRČMA, Jan. *Webová prezentace projektu JShelter*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Bednář

Webová prezentace projektu JShelter

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Martina Bednáře. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Krčma
9. května 2022

Poděkování

Rád bych tímto poděkoval Ing. Martinu Bednářovi za jeho ochotu a čas, který mi věnoval vedením této bakalářské práce a za poskytnutí jeho odborné pomoci. Také bych rád poděkoval své rodině, přítelkyni a fence Berthě, kteří při mě při psaní této práce stáli, motivovali mě a vždy mě podpořili. Dále patří mé díky všem průběžným a finálním testerům, bez kterých by práce nemohla vzniknout.

Obsah

Kapitola 1

Úvod

Každý dobrý produkt, v tomto případě rozšíření webového prohlížeče, je nutné nabídnout veřejnosti tak, aby mu rozuměla a pochopila, v čem jsou jeho výhody. V dnešní době, kdy je sice většina lidí v práci s počítačem zdatná, se ale stále najdou tací, kterým je potřeba mnoho věcí vysvětlit a ukázat, jakým způsobem pomáhají.

Každá návštěva škodlivé webové stránky zanechává tzv. otisk webového prohlížeče, dle kterého je útočník schopný identifikovat počítač návštěvníka dané stránky, jelikož samotný prohlížeč si dokáže o zařízení, na kterém běží, zjistit mnoho informací.

JShelter [16] se právě tomuto snaží zabránit statistickou anonymizací některých dat takovým způsobem, aby byl otisk co nejobecnější. Další jeho strategie je zamezení získání otisku tím, že žádaná data vůbec neposkytne a nebo je zašumí nepřesnostmi (tzv. *farbling*). Díky tomu také dokáže bránit proti určitým internetovým útokům zneužívajících funkcí prohlížeče, například skenování lokální sítě nebo určování identity počítače pomocí odchylky vnitřních hodin.

Tato práce je strukturálně rozdělena do několika kapitol.
Ve druhé je popsán otisk webového prohlížeče a jak jej lze získat. Dále jsou zde zmíněny webové aplikace unikátnosti, které uživateli jeho otisk představí a také jsou zde vyzdviženy rizika i výhody s otiskem spojené.
Třetí kapitola se zaobírá internetovými útoky, u kterých je zmíněn jejich mechanismus a způsoby zneužití.
Ve čtvrté je vysvětlený princip funkce projektu *JShelter* pro zabezpečení pohybu na webu a jeho zabránění jednotlivým internetovým útokům.
Následně je v páté kapitole popsán návrh webové prezentace projektu, u kterého jsou vyzdviženy splněné požadavky a jejich přínos.
Šestá kapitola se zaobírá implementací dané webové prezentace. Jsou zde popsány použité technologie a důvod jejich zvolení v závislosti na požadavcích vedoucího, vzhledu stávajících stránek projektu ale i koncových uživatelů.
V poslední kapitole je kladen důraz na testování výsledné webové prezentace s koncovými uživateli různých věkových kategorií a také kontrola poskytované bezpečnosti v rámci internetových útoků.

Kapitola 2

Otisk webového prohlížeče

Tato kapitola je věnována podrobnému popisu otisku webového prohlížeče a jsou v ní vysvětleny způsoby, jak jej lze získat. Dále se věnuje webovým aplikacím, které uživateli zjistí a ukáží jeho otisk, porovnají jej s databází a navrhnou mu způsoby, jak jej lze minimalizovat. V poslední sekci kapitoly jsou rozebrány bezpečnostní rizika a výhody s otiskem spojené.

2.1 Popis otisku webového prohlížeče

Otisk webového prohlížeče je unikátní soubor dat získaných z uživatelského zařízení. Čím více dat je k jeho sestavení použito, tím je otisk přesnější. Můžeme si to představit, jako bychom lidský otisk prstu měli rozdělený na několik desítek částí. Jedna dvě části nejsou dostatečné, abychom daného člověka identifikovali, ale s každou další částí se počet osob, kterým by daný otisk mohl patřit, snižuje. Úplně stejným principem funguje otisk webového prohlížeče. Kdybychom například útočníkovi nějaká data neposkytli, můžeme si to představit jako kdybychom si přes část lidského prstu přelepili černou lepící pásku. Díky ní bychom teoreticky neměli být, pomocí našeho upraveného otisku, identifikovatelní, za předpokladu toho, že bychom neměli dostatečně unikátní zbylé části. Nejdůležitější je, abychom upravili nebo neposkytli co nejvíce dat, za účelem ztížení naši identifikovatelnosti.

Uživatel je nejvíce zranitelný, pokud je jeho otisk natolik unikátní, že je skoro jedinečný napříč celou sítí internet. Pokud má například uživatel zvolený jazyk prohlížeče takový, který se na světě používá málo, je mnohonásobně zranitelnější než uživatel s jazykem běžnějším, třeba angličtinou. Další roli hrají jednotlivé komponenty uživatelova zařízení, jako je například množství operační paměti, software grafické karty, počet jader nebo i operační systém, dále také nastavení prohlížeče, povolení či zakázaní souborů *cookies*, zvolené časové pásmo, nainstalované fonty písma, spustitelnost kódu jazyka *JavaScript* nebo třeba informace ohledně podpory dotykové obrazovky. Toto vše je skrze webový prohlížeč při navštívení zneužité stránky útočníkovi, bez nastavení větší ochrany v prohlížeči či nainstalování podpůrných programů bezpečnosti, přístupné a dostupné ke zneužití.

2.2 Jak lze otisk získat

S vývojem tvorby webových stránek narůstá počet těch, které jsou dynamické. V praxi to znamená, že jich čím dál tím více používá nějaký skriptovací jazyk, aby fungovaly, například až 95 procent obsahuje ten nejrozšířenější, *JavaScript* [23]. Skript napsaný v nějakém z těchto jazyků může po dobu, co je uživatel na dané stránce, provádět výpočty na klientské straně, aniž by si toho návštěvník všiml. Může tímto způsobem například přenést zátěž z webových serverů na zařízení klienta, nebo v horším případě si o uživateli zjistit soukromá data, které pak může použít k vytvoření otisku.

Když se uživatel pokusí navštívit webovou stránku, jeho prohlížeč jí automaticky pošle požadavek s žádostí, aby k ní mohl přistoupit. V průběhu jejího načítaní však na ní umístěné reklamy nebo sledovače mohou způsobit, že prohlížeč bude posílat i další požadavky na jiné stránky třetích stran a to dokonce i v řádech desítek či stovek. V těchto požadavcích se mohou objevit dodatečné informace, jako je ku příkladu uživatelovo časové pásmo, jeho nastavení prohlížeče, typ operačního systému a mnohé další zmíněné výše v podkapitole 2.1. Část těchto informací posílá prohlížeč standardně, jelikož jsou nutné ke správnému zobrazení navštívené stránky. Do této skupiny se řadí rozlišení obrazovky, typ prohlížeče a jeho verze a v neposlední řadě i druh zařízení. Ostatní nezbytné informace si pak musí útočník zjistit sám. Na první pohled se tyto informace mohou zdát nevýznamné a žádným způsobem neohrožující soukromí uživatele, ale za pomoci díky nim vytvořeného otisku webového prohlížeče lze sestavit profil chování, který by mohl zahrnovat informace od politické příslušnosti až po často nakupované produkty. Kromě otisku se k sestavení profilu používají *cookies*, což jsou střípky informací, které si uživatelem navštívené webové stránky uchovávají na jeho zařízení, jako je ku příkladu stav přihlášení nebo obsah nákupního košíku. *Cookies* však mohou být na úkor uživatelské pohodlnosti zablokovány, aby nepředstavovaly sledovací hrozbu. Také mohou být odstraněny a tím by webové stránky přišly o uložené informace a již by je nemohly používat ke sledování.

2.3 Webové aplikace unikátnosti

Ke zjištění unikátnosti a tím pádem i zranitelnosti jednotlivých uživatelů existuje několik webových aplikací, které názorně demonstrují výše zmíněné skutečnosti. Nejvíce známe projekty jsou ku příkladu *Am I Unique*[1] nebo *Cover Your Tracks*[2], která je podrobněji popsána níže.

Projekt *Cover Your Tracks* má 2 cíle. Jednak ukázat a vysvětlit uživatelům, jak je jejich webový prohlížeč unikátní a do jaké míry jej lze identifikovat. Druhý cíl je výzkumný projekt k odhalení nástrojů a technik online sledování a také otestování účinnosti bezpečnostních webových rozšíření, jako je právě projekt *JShelter*.

Tím, že si uživatel spustí test na této stránce, mu poskytne *Cover Your Tracks* informaci o ochraně soukromí poskytované jeho prohlížečem, ale také zároveň pomůže *EFF (Electronic Frontier Foundation)*, což je nadace, která *Cover Your Tracks* vyvíjí, aby použila statistické metody k vyhodnocení schopnosti sledování třetími stranami a nejlepší způsoby, jak se proti nim bránit.

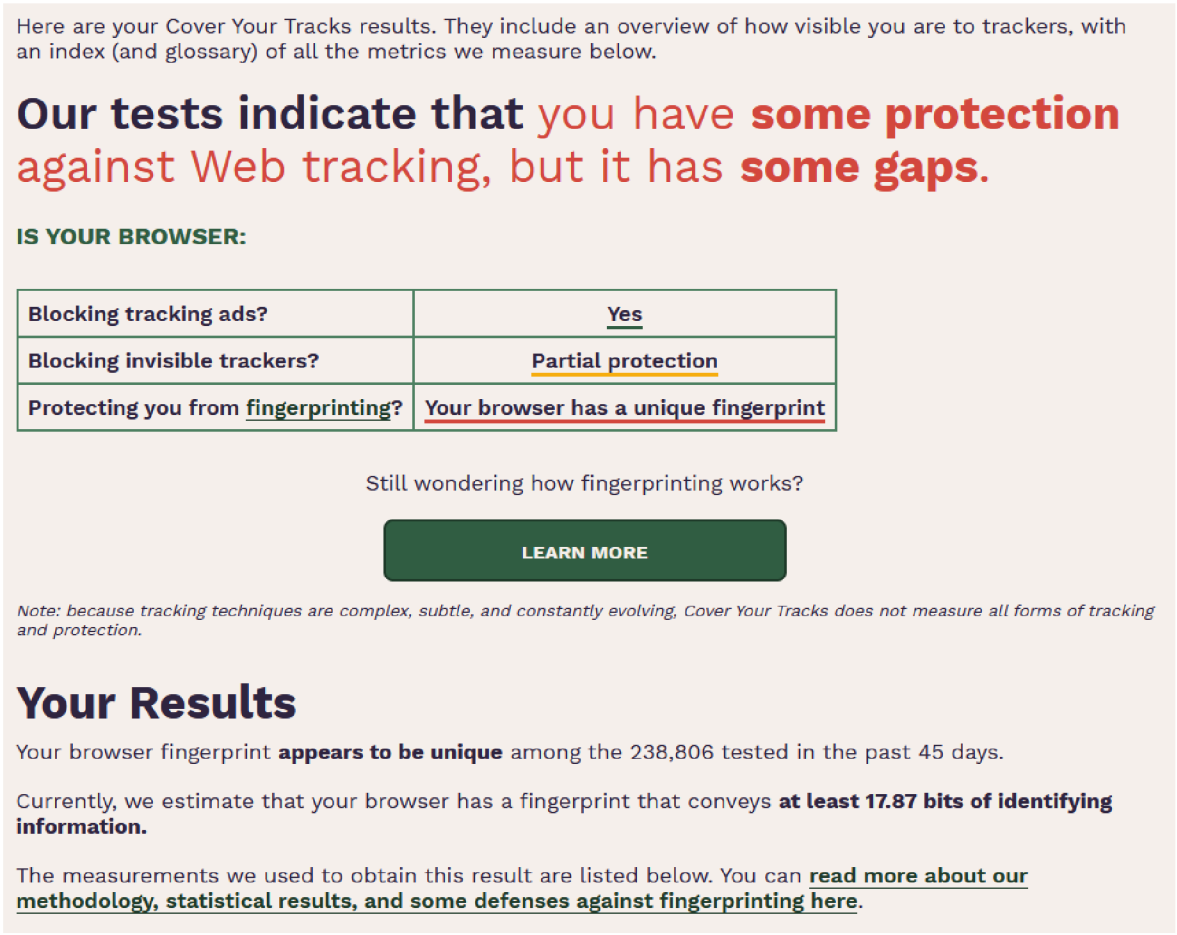
[1] Dostupné na: https://amiunique.org
[2] Dostupné na: https://coveryourtracks.eff.org

Tento projekt začal roku 2010, kdy jej spustila nadace *EFF* pod názvem *Panopticlick*. Byl to vědecký projekt k odhalení unikátnosti jednotlivých webových prohlížečů. Zjišťovány jsou informace ohledně nastavení a verze uživatelských operačních systémů, prohlížečů a jejich rozšíření. Všechny tyto informace jsou následně použity k porovnání s ostatními daty v databázi jiných uživatelů. Poté je pro názornější vyobrazení jednoduchosti identifikace při pohybu v síti internet vygenerováno skóre unikátnosti. Dřívější vyhodnocení statistických výsledků tohoto experimentu bylo zveřejněno v symposiu *(PETS – Privacy Enhancing Technologies Symposium)* [17].

Milióny uživatelů internetu používají rozšíření webových prohlížečů pro zvýšení soukromí, ale i jiné technologie k zablokování sledování, jako jsou například *Disconnect*, *Ghostery* nebo *AdBlock*. Proto byl v roce 2015 tento projekt rozšířen o funkci *tracker blocker testing*. Cílem této funkce je zjištění efektivity těchto rozšíření.

Nejnovější verze byla spuštěna roku 2020 s nynějším názvem *Cover Your Tracks*, která spojuje určení unikátnosti webového prohlížeče i efektivity jeho sledovacích blokátorů. Analyzuje se úroveň zabezpečení proti sledování jednak kontrolou nastavení obrany tím, že jsou simulovány jednotlivé odlišné způsoby sledování a také zjištěním úrovně bezpečnosti v návaznosti na to, jestli se sledování povedlo či ne.

I když jsou bezpečností rozšíření plně funkční, nemusí vždy poskytnou dostatečnou ochranu, jelikož i přes jejich snahu může být otisk webového prohlížeče dostatečně unikátní. *Cover Your Tracks* analyzuje unikátnost uživatelova prohlížeče v porovnání s ostatními, kteří jejich stránku nedávno navštívili. Následně se vygeneruje zpráva ohledně zabezpečení proti sledování i otisku prohlížeče, vizte obrázek 2.1, která slouží uživateli jako zpětná vazba, aby se mohl případně lépe zabezpečit. Anonymní výsledky testování jsou následně použity pro ostatní uživatele, kteří tento test také absolvují, k vytvoření jejich zpráv.



Obrázek 2.1: Zpráva generovaná webovou aplikací *Cover Your Tracks*

K dosažení výsledků používá *Cover Your Tracks* simulované sledovací domény, aby zjistila reakci blokátorů sledování, které má uživatel aktivní.

Určité blokátory (například *Ghostery*) mají jako jejich hlavní spouštěč *URL* parametry, které se shodují s databází známých reklam a sledovacích majáků. Jiné blokátory (například *Disconnect* nebo *AdAway*) mají jako spouštěč určité domény a *Cover Your Tracks* se snaží se k těmto doménám zařadit, aby mohl vyzkoušet jejich účinnost. Další blokátory (jako je například *Privacy Badger*, také vyvíjený *EFF*) používají heuristický přístup, tudíž blokují sledování na základě jejího použití napříč doménami.

Pro zjištění těchto odlišných způsobů obrany projekt *Cover Your Tracks* simuluje sledování, které spustí všechny druhy blokací tím, že generuje požadavky třetích stran jako ku příkladu [5]:

```
https://simulator-sledovani.cz/?akce=sleduj&urlreklamy=3251
https://zaskodne-sledovani.cz/?akce=sleduj&sledovaciserver=1297
https://nesleduj-me-prosim.cz/?akce=sleduj&cislo=987654
```

Aby se potvrdilo zabezpečení heuristickým přístupem, každá z těchto *URL* adres je načítána z domén, které se snaží nastavit *cookies* a uložit si je na jejich serverové straně.

První *URL* simuluje sledování za pomoci viditelné reklamy, která pokud se nezobrazí, tak byl test úspěšný.
Druhá *URL* simuluje neviditelný sledovací maják, který pokud bude zablokovaný, byl test úspěšný.
Poslední *URL* má doménu, která používá *Do Not Track Policy* a pokud budou doménové skripty povoleny, test uspěje.

Pokud se simulovaná reklama nebo maják načtou, ale jejich *cookies* zůstanou zablokované, považuje se tento výsledek jako částečná ochrana, jelikož sice webová stránka jednoduše nezíská unikátní identifikátor, ale stále nezabraňuje sledování za pomocí *IP* adresy nebo jinými způsoby.

Ke změření unikátnosti otisku webového prohlížeče zjišťuje *Cover Your Tracks* následující informace:

- řetězec uživatelského klienta z každého prohlížeče,
- `HTTP ACCEPT` hlavičku odeslanou prohlížečem,
- rozlišení obrazovky,
- hloubku barev,
- časové pásmo systému,
- rozšíření webového prohlížeče jako je *Java* nebo *Flash*,
- *Acrobat* nebo *Quicktime* a jejich verze,
- nainstalované fonty,
- jestli prohlížeč povoluje skripty jazyka *JavaScript*,
- informace o povolení *cookies*,
- hash obrázku generovaném plátnem nebo *WebGL*,
- zda prohlížeč posílá navštíveným stránkám *Do Not Track* hlavičku,
- operační systém a jeho nastavený jazyk
- a zda prohlížeč podporuje dotykovou obrazovku.

Následně za pomoci těchto dat vyhodnotí pro uživatele jeho skóre unikátnosti [7].

2.4 Bezpečnostní rizika a výhody

Tím, že se díky otisku identifikuje právě jedno zařízení, lze následně jeho pohyb sledovat napříč sítí internet [22], a to i v anonymním režimu prohlížeče.

Pokud se identifikace podaří, následný uživatelský profil může být prodán zprostředkovatelům dat, kteří se snaží získat o všech lidech co nejvíce informací. Kombinují online i offline veřejné záznamy a následně si mohou pomocí získaných informací s přesným otiskem uživatelova webového prohlížeče vytvořit podrobnou datovou složku pro kohokoliv. Následně takový profil prodají reklamním společnostem, aby mohly využívat cíleně zaměřených reklam.

Velkou výhodou zneužívání otisku webového prohlížeče a ne sledování za pomocí *cookies* je především tajnost, jelikož aby mohl být uživatel sledován přes *cookies*, musí k tomu dát svolení. Stačí si pouze představit, kolik senzitivních dat je uloženo v uživatelské historii vyhledávání [21]. Pokud si například vyhledá nějaký příznak nemoci, může být tato skutečnost prodána jeho pojišťovně, která může usoudit, že je uživatel náchylný na nemoci srdce a příslušně díky tomu zvýšit cenu jeho pojistky.

Otisk webového prohlížeče nemusí být vždy použit pouze k nekalým účelům, ale například se pomocí něj dají odhalit charakteristiky *botnetů* [3], což je síť nakažených počítačů, které mohou být vzdáleně kontrolovány a nuceny zasílat spam, šířit malware nebo provádět *DDoS* útoky a to i bez svolení vlastníka nakaženého zařízení. Také se pomocí něj banky snaží odhalit krádež identity a bankovních podvodů.

Kapitola 3

Webové útoky

Tato podkapitola se zaobírá podrobným rozepsáním a popsáním jednotlivých internetových útoků. Je zde zmíněn jejich průběh, nutná příprava a cíle, jakých lze při jejich použití dosáhnout.

3.1 Meltdown

Meltdown je hardwarová slabina postihující *IBM* procesory, *Intel x86* mikroprocesory a určité mikroprocesory na bázi *ARM*, která dovolí útočníkovu procesu přistupovat a číst i z paměťového prostoru ostatních procesů, i když k tomu nemá žádné povolení. *Meltdown* ohrožuje široké spektrum systémů, do kterých například spadají zastaralé neaktualizované operační systémy *Windows*, *Linux*, *macOS* ale také *iOS*. Dále byly postiženy servery, většina chytrých a vestavěných zařízení využívající procesory na bázi *ARM*, například tiskárny, mobilní zařízení, televizory a další.

Jako čistě softwarová ochrana proti *Meltdown* bylo použito snížení výkonnosti zařízení o 5 až 30 procent [13]. Toto se dle tvůrců opravy projeví na výkonu zařízení pouze minimálně.

Meltdown zneužívá souběhu, inherentního pro design mnoha moderních procesorů. Tento jev nastává mezi přístupem k paměti a kontrolou oprávnění při čtení a vykonávání instrukce. Společně s útokem na postranní kanál mezipaměti lze umožnit procesu, aby obešel běžné bezpečnostní kontroly jeho oprávnění, jenž mu brání v přístupu k datům náležícím operačnímu systému a i jiných spuštěných procesů. Tudíž se díky této slabině nepovolenému procesu podaří dostat do celé zmapované paměti daného procesu.

Existuje několik hardwarových možností, jak zabránit souběhu a tím také znemožnit *Meltdown*, mezi ně spadá úprava mikrokódu procesoru nebo cesty spuštění.

U některých operačních systémů může *Meltdown* číst i paměť, ke které by se nakažený program vůbec neměl možnost normálně dostat, například k datům kernelu nebo i jiných procesů.

U útoku za pomoci zranitelnosti *Meltdown* nelze zjistit, jestli proběhl nebo právě probíhá.

3.2 Spectre

Spectre je třída bezpečnostních zranitelností, které postihují moderní mikroprocesory, jenž provádějí předpovídání větvení nebo jiné druhy odhadů. Na většině procesorů může spekulativní provádění vycházející ze špatného předpovídání větvení zanechat viditelné nežádoucí následky, které mohou umožnit útočníkům odhalit privátní data uživatele. Ku příkladu pokud by schéma přístupů do paměti vykonané spekulativním prováděním záviselo na privátních datech, následný stav datové mezipaměti by vytvořil boční kanál, skrze který by mohl útočník získat privátní data za pomoci časovaného útoku.

Pro *Spectre* byly vytvořeny 2 identifikátory v rámci *Common Vulnerabilities and Exposures*, *CVE-2017-5753 (bounds check bypass, Spectre-V1, Spectre 1.0)*[1] a také *CVE-2017-5715 (branch target injection, Spectre-V2)*[2]. *JIT engine*, jenž se používá pro chod *JavaScriptu*, byl označen za náchylný, jelikož pomocí něj může jedna webová stránka číst data uložená v paměti prohlížeče jiné stránky nebo i paměť samotnou [9].

Článek popisující *Spectre* rozděluje útok do 4 základních kroků [11].

1. Ukazuje, že predikátní logika předvídání větvení v moderních procesorech může být natrénována ke spolehlivému *hit or miss* založenému na bázi vnitřního chodu útočného programu.

2. Dále popisuje, že rozdíl mezi *cache-hit* a *cache-miss* lze spolehlivě načasovat, tudíž to, co by mělo být jednoduchým rozdílem bez zřejmého využití, může být ve skutečnosti přeměněno na skrytý kanál, který extrahuje informace z vnitřního fungování jiného nesouvisejícího procesu.

3. Článek dále poukazuje na výsledky s pomocí programování orientovaného na návratová data a jeho zneužití společně s dalšími principy za pomoci jednoduchého příkladu programu a fragmentu *JavaScript* zdrojového kódu běžícího v testovacím prostředí prohlížeče. V obou případech se ukázalo, že celý adresový prostor běžícího programu byl čitelný pouze za použití spekulativního provedení podmínkových větví v běžícím kódu generovaných vestavěným kompilerem nebo *JavaScript enginem* již přítomným ve webovém prohlížeči. Základní myšlenkou je prohledat existující kód pro nalezení částí, ve kterých by se spekulativní větvení mohlo snažit přistoupit do adresového prostoru, ke kterému by za normálních okolností neměl program mít přístup, manipulovat procesor do stavu, ve kterém by spekulativní vykonání muselo kontaktovat data. Načasováním vedlejšího efektu toho, že je procesor rychlejší, se naskytá možnost předběžně načíst řádek mezipaměti.

4. Závěr práce se zaobírá zobecněním útoku na jakýkoliv stav procesu oběti, který nemá žádnou funkci. Stručně také pojednává i o tak zcela nezřejmých efektech bez evidentní funkce, jako je latence arbitráže sběrnice.

[1] Dostupné na: https://www.cve.org/CVERecord?id=CVE-2017-5753
[2] Dostupné na: https://www.cve.org/CVERecord?id=CVE-2017-5715

3.2.1 Zneužití na dálku

I když je *Spectre* snadnější vykonat s pomocí kompilovaného jazyka, jako je například *C* nebo *C++* a lokálním spuštěním strojového kódu, lze jej také zneužít tím způsobem, že jeho zdrojový kód umístíme na webovou stránku. Tento postup využívá především jazyku *JavaScript*, který běží lokálně za využití webového prohlížeče. Takto skriptovaný *malware* by v takovém případě měl přístup do celé namapované paměti přiřazené prohlížeči.

Vzdálené zneužití za pomoci *JavaScriptu* je podobné, jako zneužití s využitím strojového kódu. Nejdříve se vyprázdní mezipaměť, poté se špatně vycvičí prediktor větvení a následně se s využitím časovaného čtení kontroluje *hit or miss*.

Jelikož v *JavaScriptu* chybí instrukce `clflush` *(cache-line flush – vyprázdnění mezipaměti)* je nutné pro provedení útoku zvolit jiný přístup. Procesorová jednotka si vždy vybírá z několika automatických čistění mezipaměti a útok spoléhá na to, že dokáže donutit procesor, aby čištění provedl. Zjistilo se, že použití druhého indexu velkého pole, které bylo několik iterací za prvním indexem, vede k použití strategie *LRU (last recently used)*, jenž vyčistí z paměti v poslední době nejméně používané indexy a tím pádem efektivně vyčistí mezipaměť jenom díky tomu, že útok provede několik iterací na velkém poli.

Předvídač větvení by následně byl zmaten iterováním přes velmi obsáhlý objem dat za pomoci bitových operací pro nastavení indexu tak, aby byly dostupné a nakonec by byl použit index, který by nebyl naadresovaný.

Ke zjištění, jestli série čtení paměti vedla ke *cache-hit* nebo *cache-miss*, by byl nutný časovač s vysokou přesností. Webové prohlížeče jako *Chrome*, *Firefox* a *Tor*, který je založený na *Firefoxu*, již za pomoci rozšíření přesnost časovačů snižují a omezují tím *Spectre* v rozhodování úspěšnosti čtení.

3.3 Zneužití prohlížeče jako prostředníka

JavaScript běžící na webové stránce, kterou uživatel navštíví, se obvykle může za pomoci jeho webového prohlížeče, jenž využije jako prostředníka, připojit ke službám, které jsou spuštěny jak na jeho zařízení, tak i k těm, jenž se nachází na jiných počítačích v jeho interní síti. Moderní webové prohlížeče, alespoň ty běžné, této lehce zneužitelné slabině nejsou schopny zabránit, ale spíš jsou jejími největšími nástroji, jak tento útok vykonat. Díky tomu, že mohou posílat požadavky do interní sítě, rozpoznávat v ní jiná zařízení a částečně skenovat porty, mohou také ohrozit spuštěné služby. To vše jen za pomocí škodlivého *JavaScriptu*.

Jelikož se skrze telemetrii [2] nepodařilo objevit webové stránky, které by do interní sítě posílaly požadavky, které by byly neškodné, lze usoudit, že pokud by stránka takový požadavek zaslala, je takové chování vhodné považovat za nebezpečné a snažit se ho omezit, poněvadž vývojáři lokálně běžících programů při zabezpečovaní nepočítají s tím, že by se na ně mohl snažit připojit útočník ze sítě internet. Jeden z tímto způsobem ohrožených programů je například aplikace *Logitech Options*, která otevírá zranitelný *WebSocket server* [2].

Tvůrci webových prohlížečů o této zranitelnosti ví a implementovali
Same-origin Policy (SOP), která sice povoluje zasílání požadavků do interní sítě, ale zabra-

ňuje tomu, aby byl výsledek čitelný pro útočníka, který si jej vyžádal. Ovšem i nečitelný výsledek může pomoci útok vykonat.

Same Origin Policy lze rozdělit do 3 kroků:

1. uživatel přistoupí na veřejnou útočníkovu webovou stránku, na které se nachází *JavaScript*, který zašle *XMLHttpRequest* do interní sítě, se kterou by neměl mít možnost kvůli *SOP* korektně komunikovat,
2. webový prohlížeč požadavek zpracuje a odešle příslušné službě,
3. následně obdrží výsledek, ale již ho nepředá odesílateli.

K provedení útoku je nejdříve potřeba znát *IP* adresu napadaného zařízení. Samotné zařízení bude vždy reagovat na požadavky zaslané na 127.0.0.1, jelikož se jedná o alias *localhost*, takové požadavky by si tím pádem posílal sám sobě. Dále je potřeba zjistit *IP* adresu, pod kterou zařízení komunikuje v rámci interní sítě, abychom mohli cíleně hádat *IP* adresy ostatních zařízení, nacházejících se v této síti. Ke zjištění interní *IP* adresy lze použít například *WebRTC API*, které ale nelze použít na všech kombinacích operačních systémů a webových prohlížečů. Tuto adresu může následně záškodný *JavaScript* odeslat útočníkovi, který ji využije pro provedení útoku.

Předtím, než je otestována otevřenost portů a služeb na nich běžících, je nutné zjistit jaká další zařízení se na interní síti nacházejí. Zjišťování bude probíhat hádáním *IP* adres za pomocí toho, že většina domácích routerů a síťových zařízení menších firem, například *ADSL* modemy, používají k přiřazování *IP* adres prostory 192.168.0.0/24, 192.168.1.0/24 a 10.0.0.0/8, kterou například využívá společnost *O2* u svých domácích modemů. S velkou pravděpodobností se v této podsíti bude uživatel nacházet. Dále je také potřeba zmínit, že levné *DHCP* servery začínají s přiřazováním *IP* adres automaticky od oktetu .100, tím pádem by se mělo prioritněji prohledat například také 192.168.0.100+ nebo 192.168.1.100+. U větších firem se poté objevují adresové prostory jako jsou 172.16.0.0/24 nebo 10.0.0.0/24. Slepé prohledávání by zde bylo neúčinné, tudíž je výhodnější zjistit *IP* adresu uživatele výše zmíněnou metodou za pomoci *WebRTC API* a následně hádat *IP* adresy dalších zařízení v blízkosti té uživatelské.

Jakmile má útočník seznam možných *IP* adres, dalším jeho krokem je kontrola, jestli jsou k těmto adresám opravdu přiřazená zařízení v interní síti. Jelikož *SOP* brání v tom, aby se přímo připojil na nějaký port a zjistil tím, že zařízení existuje, musí se při obdržení odpovědi zaslaného požadavku rozlišovat, jestli v jeho čtení brání *SOP* a jako odpověď se vrátí *reset packet*, nebo zda požadavku vyprší platnost a nastane *timeout*. Poslední možnost by znamenala, že buď zařízení s touto *IP* adresou v interní síti neexistuje, nebo není spuštěné. Protože velké množství portů může být zavřené, nepoužívané nebo blokovány prohlížečem, jelikož nesouvisí s *HTTP*, je vhodné vyzkoušet ty základní jako jsou 80, 443 a 8080. Výsledkem tohoto kroku by byl seznam *IP* adres a jejich portů společně s jejich otevřeností či zavřeností.

Nyní má útočník seznam spuštěných zařízení a jejich možná otevřených portů a zajímají jej na nich spuštěné služby. Díky *SOP* je ale nezjistí snadno, ale jelikož každá služba má svoje logo nebo obrázek ikony, který je uložený v tom stejném adresáři a se stejným jménem v rámci všech zařízení, na kterých byl nainstalován, je schopný poslat požadavek webovému

prohlížeči na jeho zobrazení. Pokud obrázek existuje, proběhne event *onload* a lze usoudit, že se na zařízení tato služba nachází. Pokud ale proběhne event *onerror*, obrázek neexistuje a není potom v tomto případě pravděpodobné, že by se služba na zařízení nacházela. Tímto způsobem si útočník, za pomoci velkého seznamu nejběžnějších služeb a lokací uložení jejich log a ikon, může otestovat jednotlivá zařízení a porty a tím pádem zjistit, jaká zařízení jsou například tiskárny, jaký router je používán nebo jaké služby jsou na zařízeních nainstalované.

Tímto způsobem lze rozšířit otisk webového prohlížeče a snáze identifikovat uživatele a to jenom díky tomu, že přistoupí na jedinou webovou stránku, na které zůstane delší dobu. K udržení jeho pozornosti se například využívá obsah pro dospělé nebo streamování filmů a seriálů, jelikož na takových stránkách zůstává uživatel déle a nezpochybňuje v takové míře jejich bezpečnost. Následně by se útočník mohl pohybovat po interní síti za pomoci *XSS* a změnění původu stránky. Také by byl schopný kompromitovat zranitelné služby.

Detekcí tohoto útoku je kontrola, jestli se *JavaScript* na cizí webové stránce umístěné v síti internet snaží připojit k privátní *IP* adrese, což by se dít správně nemělo. Jakožto obranu lze například změnit rozmezí, ve kterém jsou přiřazeny *IP* adresy. Toto není dostatečně dobrá ochrana, jelikož útočné webové stránky vetšinou skenují veškeré rozsahy privátních adresových prostorů. Větší zabezpečení nabízí procházení internetu za použití *web proxy*.

3.4 Odchylka vnitřních hodin

Typicky se v každém počítači řídí chod jeho hodin pomocí krystalu, který osciluje určitou frekvencí. Jelikož nemůžou být, i přes veškerou snahu jejích výrobců, všechny stejné i na atomární úrovni, vyskytují se u nich odchylky. Tím pádem se hodiny mohou buď opožďovat, nebo předbíhat, úplně stejně, ale v menším měřítku, jako u hodinek analogových. Následný rozdíl je vždy srovnán za pomocí *Network Time Protocol (NTP)* serveru, který je synchronizuje na správný čas. Sice se nejedná o změny v rámci vteřin či minut, ale i minimální posuny času mohou uživatelův počítač odlišit od jiných a lze tento fakt využít k jeho identifikaci nebo ke zlepšení jeho otisku webového prohlížeče.

Odchylka se počítá tím způsobem, že útočník vezme čas uložený v paketu, který od sledovaného uživatele odchytil a odečte jej od svého času. Započítat se ale musí doba potřebná ke zpracování a odeslání paketu z jednoho zařízení a jeho přijetí na druhém, k tomu také ještě čas, který paket strávil na cestě mezi nimi.

Existuje hned několik způsobů, jak tuto odchylku zjistit. Rozlišovány jsou zejména na aktivní a pasivní, ty se následně dělí na krátkodobé či dlouhodobé [15].

Při provádění pasivního útoku útočník pouze monitoruje a zaznamenává časová razítka procházejících paketů, nejčastěji z hlaviček protokolů, převážně *HTTP* nebo *TCP*. Pouhým naslouchánim na sebe útočník neupoutá nechtěnou pozornost a může si nerušeně sbírat data, která následně použije k útoku.

Aktivní útok využívá vsazování paketů sond časových razítek, aby zvětšil oproti pasivnímu přístupu jejich množství. Využívá se k tomu například *ICMP* nebo generování časových razítek za pomocí programovacího jazyku jako je *Java* či *Python*. Tímto způsobem ale útočník

zanechává v síti stopy a je náchylný k jeho odhalení napadeným uživatelem.

Krátkodobé sbírání časových razítek a určování odchylky vnitřních hodin má za cíl co nejrychleji odhalit zařízení, po kterém útočník pátrá.

Oproti tomu dlouhodobé sbírání může brát v potaz mnohem více skutečností, jako například jednotlivé synchronizace hodin zařízení s *NTP* serverem, ale také změny teplot, kvůli kterým se frekvence krystalu mění.

Různé kombinace těchto způsobů se nejvíce hodí pro různé aktivity. Za pomoci dlouhodobého pasivního útoku lze nejlépe zjistit lokaci zařízení a jeho změnu a všechny ním používané *IP* adresy. Dlouhodobý pasivní útok nabízí lepší kvalitu zjištěné odchylky oproti ostatním útokům a využívá se zejména k identifikaci virtuálních PC a deanonymizaci uživatele. Krátkodobě provedený aktivní útok se nejvíce hodí k odchylkové identifikaci při více-faktorovému ověřování [15].

Existuje hned několik způsobů, jak časové razítko od napadeného zařízení obdržet.

Pomocí *Internet Control Message Protocol (ICMP)* se lze dotázat napadeného, aby útočníkovi zaslal jeho přesný čas. Některé systémy jako je například *macOS* ale *ICMP* standardně nepodporují a byla by u nich nutná změna nastavení.

Protokoly *HTTP* a *XMPP* společně s dalšími přidávají časové razítko do aplikační hlavičky odeslaného paketu, ale tato razítka se velmi zaokrouhlují a díky tomu znesnadňují vypočítání odchylky.

TCP posílá časová razítka ve všech vrstvách paketu při navazování spojení klienta se serverem, ze kterých si je útočník pasivně může zjistit. U operačních systémů *Windows* jsou ale tato razítka standardně zakázána a stejně jako u *ICMP* by musela být povolena změnou nastavení.

Zároveň pomocí technologie *AJAX* s využitím jazyka *JavaScript* nebo *Python* může útočník umístit na volně přístupnou webovou stránku kód, který mu bude zjišťovat přesný čas uživatelova zařízení a zároveň mu odesílat přesná časová razítka.

Pomocí experimentů bylo dokázáno, že zjišťování odchylky vnitřních hodin za pomocí jazyku *Python* potřebuje méně paketů, než za použití metody *TCP*, která však zase potřebuje kratší čas [15].

Jelikož více-faktorové ověřování využívá odchylku jako jeden z faktorů, pokud by ji útočník zjistil a napodobil za pomocí zneužití *NTP* serveru, může tento faktor být lehce oklamatelný.

V dnešní době začíná být čím dál tím běžnější, že i když musí mít zařízení pouze jednu `IPv4` adresu, tak adres `IPv6` může mít tolik, kolik si jich zvládne vytvořit. Internetový provoz, který jakákoliv adresa vyprodukuje, musí mít na jeho paketech stejnou odchylku vnitřních hodin a tím pádem lze porovnat jednotlivé vypočítané odchylky a zjistit, které adresy vycházejí z jednoho a toho stejného zařízení. Tímto způsobem se potlačuje anonymita `IPv6`

adres, jelikož různé IPv6 adresy lze spojovat a přiřazovat k jedinému zařízení. Uživatel si tím pádem nemůže měnit adresy a vždy očekávat, že nebude odhalen jejich stejný původ.

Kapitola 4

JShelter

Tato kapitola se zaobírá rozšířením webových prohlížečů, projektem *JShelter*, které si klade za cíl vrátit uživateli kontrolu nad chováním jeho prohlížeče a především se zaměřuje na zesílení soukromí a bezpečnosti.

Stejně tak, jak firewall kontroluje síťové spojení, kontroluje *JShelter API* poskytované webovým prohlížečem. Tím pádem omezuje jeho možnosti v získávání informací a jejich následnému poskytování webovým stránkám. Zajišťuje přídavnou bezpečnostní vrstvu, u které si uživatel může zvolit, na jaké webové stránky se jaká omezení budou aplikovat. Při správném nastavení může tato bezpečnostní vrstva zamezit internetovým útokům jak na samotný prohlížeč, tak i na celé zařízení.

Za nápadem na vytvoření *JShelteru* stál na jeho počátku Ing. Libor Polčák Ph.D. Ze začátku byl projekt pojmenován *JavaScript Restrictor*, následně se v roce 2021 přejmenoval na nynější název a rozrostl se o zahraniční výzkumný tým *FSF (Free Software Foundation*, jmenovitě John Hsieh, Michael McMahon a Ruben Rodriguez. Mezi české a slovenské vývojáře, kteří se na projektu podíleli, patří Ing. Martin Bednář, který pracuje na vývoji a testování tohoto rozšíření v rámci jeho doktorského studia, dále v rámci svých diplomových prací projekt obohatili:

Ing. Zbyněk Červinka, jenž ověřil samotný koncept *JShelteru*, Ing. Martin Timko vyvíjel verze dostupné pro veřejnost až po 0.2.1 a zároveň rozšířil kompatibilitu na prohlížeče *Chrome* a *Opera*. Ing. Pavel Pohner vytvořil *Network Boundary Scanner*, Bc. Peter Horňák přenesl funkcionalitu z projektu *Chrome Zero* a podílel se na opravě chyb. Ing. Matúš Švancár přenesl prostředky proti zisku otisku (takzvaný *farbling*) z webového prohlížeče *Brave*, Ing. Marek Saloň vytvořil detektor získávání otisku a následně jej zdokonalil a v neposlední řadě Ing. Radek Hranický Ph.D. implementoval obalení *API* senzorů.

Projekt se místy inspiruje projektem *ChromeZero* [4] [20] a prohlížečem *Brave*.

V následujících sekcích bude popsána funkčnost *JShelteru* verze 0.8, která je v době psaní této bakalářské práce (16. 3. 2022) tou aktuální nasazenou.

4.1 Strategie proti získávání otisku

JShelter používá výhradně 3 základní a 2 pokročilé technické strategie, kterými se snaží zabránit získání uživatelova otisku webového prohlížeče třetími stranami. Jednotlivé strategie si nejsou výlučné a při obalování určitých *API* jsou, pro co možná největší efekt, kombinovány. U výčtu těch základních níže je vždy pod jejich popisem vyobrazen seznam základních *API*, které se tímto způsobem při doporučeném nastavení *JShelteru*, jenž lze vidět na obrázku 4.2, obalují.

4.1.1 Znepřesnění

První strategií je úprava zaokrouhlováním nebo znepřesněním, v praxi to znamená, že se vezme reálná hodnota a místo toho, aby se dotazujícímu odeslala přesná, zaokrouhlí se. V jakých řádech záleží na nastavení uživatele, je doporučeno zaokrouhlovat nejvýše na 2 desetinná místa, jinak se zvyšuje šance, že by mohlo být zařízení skrze odchylku vnitřních hodin nebo využitím jiných dat identifikováno.

- lokálně zobrazované obrázky
- geolokace

4.1.2 Podvržení

Další strategií, kterou používá *JShelter*, je podvržení. V tomto případě se při dotazu například na *API* ohledně počtu v prohlížeči nainstalovaných zásuvných modelů při doporučeném nastavení vrátí pouze falešné a nejběžnější. Případný útočník by takto získal falešný seznam, který by byl pro otisk nepoužitelný.

- informace o grafické kartě
- nainstalované zásuvné modely prohlížeče
- připojené video a audio vstupní zařízení
- paměť zařízení a informace o *CPU*

4.1.3 Zakázání přístupu

Poslední základní strategií je zakázání přístupu, při čemž buďto požadavek nebude vůbec k dané *API* připuštěn nebo jako odpověď dostane prázdnou množinu tam, kde by nějakou hodnotu určitě očekával. Tímto se šance na zisk otisku nemusí nutně snížit, jelikož takto upravená prázdná množina bude nejspíše velmi výjimečná a objevovat se bude pouze u uživatelů, kteří aplikují nadstandardní bezpečnostní opatření, čehož může útočník využít. Jako obrana proti útokům tato strategie funguje dobře, ale omezuje se ní funkčnost webových stránek a tím pádem také pohodlí uživatele. Je doporučeno toto nastavení používat pouze na těch opravdu nebezpečných stránkách.

- výčet gamepadů
- přístup nebo přenos dat na nespolehlivé servery pomocí majáků
- trvalý identifikátor panelu prohlížeče

4.1.4 Zašumění

Jednou z pokročilých strategií používanou *JShelterem* je takzvané *farblování* (zašumění), název této metody začali používat vývojáři webového prohlížeče *Brave*. Jedná se o minimální, ale poznatelné změny výstupu jednotlivých funkcí prohlížeče, které lze zneužít ke tvorbě otisku. Je kladen důraz na zachování funkčnosti navštívených stránek, proto si nelze činnosti této metody všimnout pouhým okem, jak lze vidět níže na obrázku 4.1, kde se vlevo zobrazuje neupravené plátno, uprostřed upravené a úplně vpravo jsou zvýrazněné změny, které vznikly *farblováním*[1]. Jelikož se změny deterministicky určují v závislosti na jedinečných sezeních, jsou na různých stránkách odlišné a tudíž útočník ztrácí možnost sledovat uživatele při pohybu v síti internet.

Obrázek 4.1: Výsledek *farblování* plátna

Například u požadavku na aktuální čas se desetinná část náhodně upraví. Více informací k tomuto přístupu lze najít v diplomové práci Ing. Matúša Švancary [24], nebo přímo na stránkách projektu *Brave* [6].

- přesnost času
- lokálně generované audio a údaje o zvukové kartě

4.1.5 Detektor získávání otisku

Detektor získávaní otisku (Fingerprint detector) je experimentální strategie, při které se obalování neaplikuje do té doby, dokud se nezjistí podezřelé chování spojitelné se získáváním otisku webového prohlížeče.

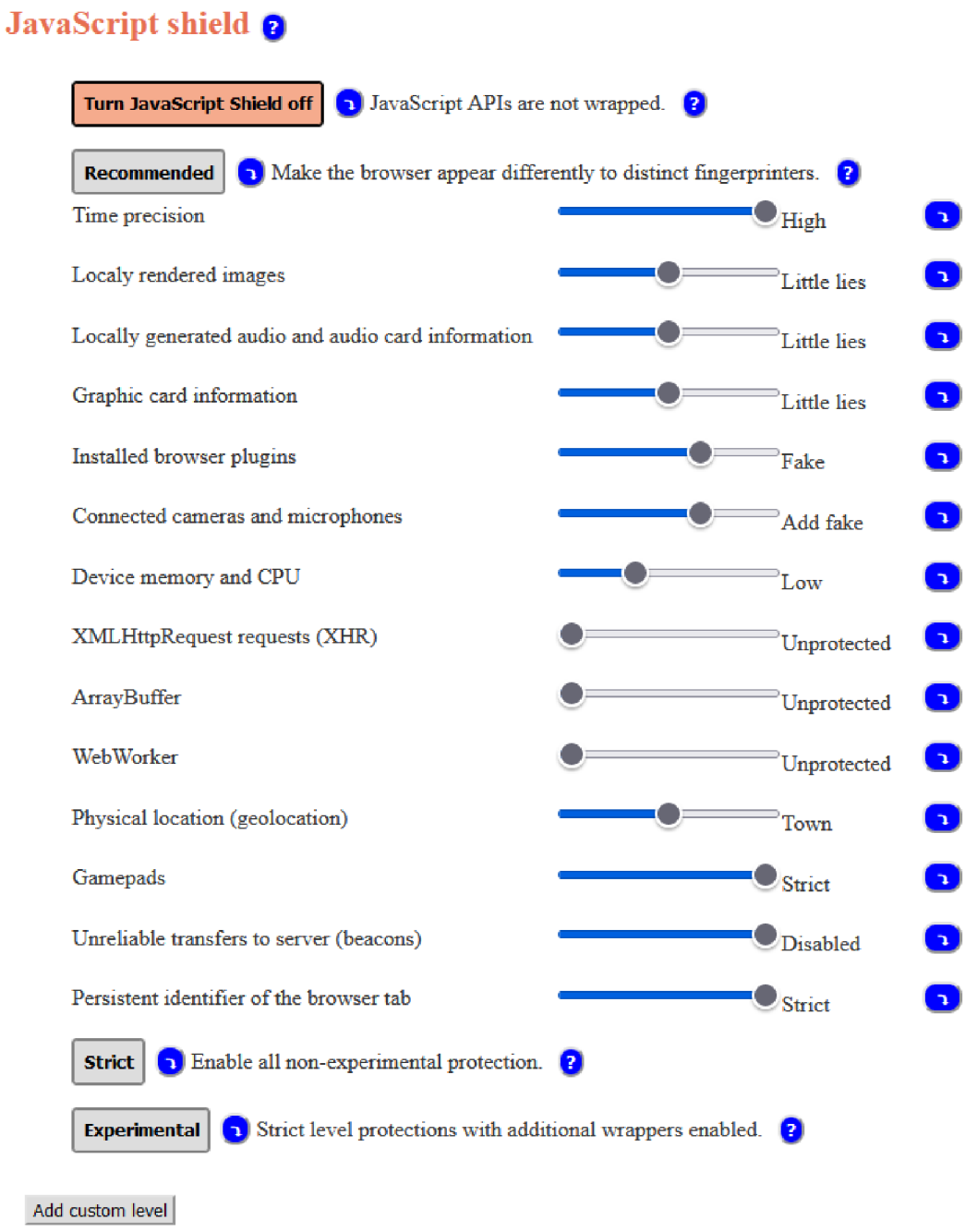
Podle nastavení si uživatel může zvolit, jestli ho na případné pokusy nebo možnosti zisku otisku detektor upozorní pouze změnou barvy ikony rozšíření, nebo i vyskakujícím oznámením, na které když si uživatel klikne, mu vygeneruje podrobnou zprávu důvodů, proč si myslí, že se jedná o nebezpečné chování webové stránky.

Další možností je blokování tímto způsobem nebezpečných stránek tím, že se mezi ní a uživatelem přeruší spojení a následně se vymaže obsah *localStorage* a *sessionStorage* za účelem

[1] Dostupné na: https://pxlsfiddle.com/farbling.html

zničení možných uložených dat. Při nejvyšší možné úrovni zabezpečení se nevymažou pouze výše zmíněná úložiště, ale také všechna ostatní, které daná webová stránka mohla využívat.

Více informací k *detektoru získávaní otisku* je k dostání v diplomové práci Ing. Marka Saloňe [19].



Obrázek 4.2: Doporučené možnosti nastavení ochrany

4.2 Obrana před útoky

Jako obranu před útoky poskytuje *JShelter* dvě opatření, která jsou postupně popsána níže. Nejdříve je zde zmíněná ochrana proti zneužití prohlížeče jako prostředníka a následně zabránění útokům využívajícím předpovídání větvení a zneužívajícím přístupu do mezipamětí.

4.2.1 Network Boundary Shield

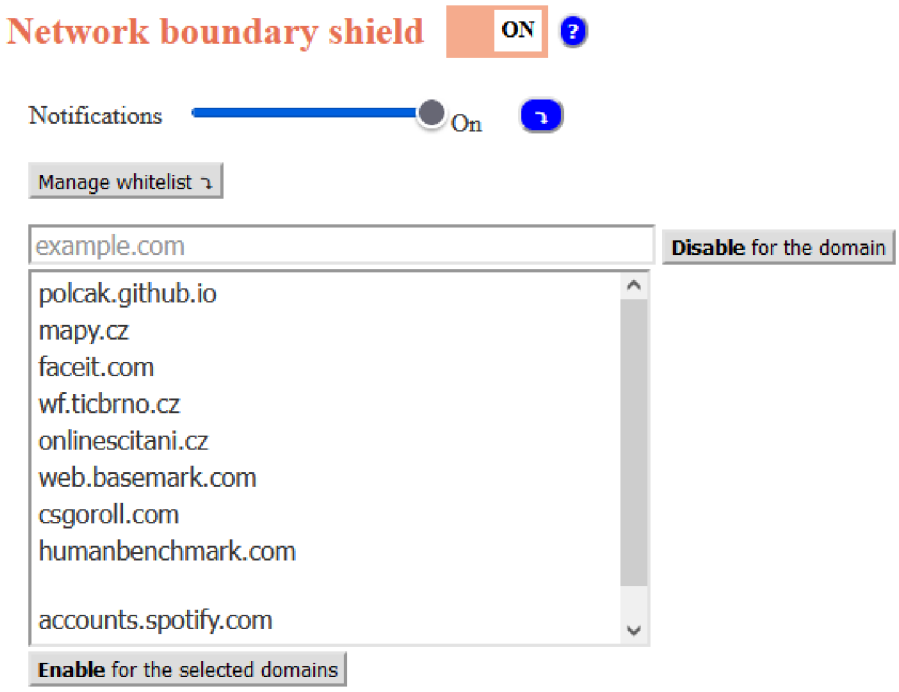
Obranu před útokem, který zneužívá prohlížeč jako prostředníka, zajišťuje *JShelter* pomocí funkce *Network Boundary Shield*, zkráceně *NBS*, jenž slouží jako ochrana proti útokům ze sítě internet směřovaných do sítě lokální, především těm průzkumným [2].

Funkcionalita *NBS* je založena na filtrování *HTTP* požadavků. Používá k tomu blokování `webRequest` *API* k přebrání *HTTP* požadavků tím, že každý požadavek je pozastaven do té doby, než je zanalyzován a případně povolen. Pokud by se zdál podezřelý, je zablokován.

Hlavním cílem *NBS* je předejití a zabránění útoků, jako třeba když veřejná webová stránka pošle požadavek na nějaký soubor nebo zdroj v privátní síti. Takhle může útočník například zjistit, jaký router používáme tím, že si zažádá o zobrazení loga jeho výrobce. *NBS* zjistí, že se webová stránka snaží poslat požadavek směřovaný na *IP* adresu do lokální sítě. Uživatel může povolit určitým stránkám výjimku, jak lze vidět na obrázku 4.3, aby se k lokálním zdrojům dostaly.

NBS zjišťuje k rozlišení prefixů lokálních a veřejných *IP* adres *CSV* soubory poskytnuté organizací *IANA* a podporuje jak `IPv4` tak `IPv6`. *CSV* soubory jsou staženy při sestavení *JShelteru*. Zároveň má také minimální dopad na výkonnost webového prohlížeče, který je ale v každé implementaci odlišný.

Více informací k *Network Boundary Shield* je k dostání v diplomové práci Pavla Pohnera [14].



Obrázek 4.3: Výjimky pro webové stránky u *NBS*

4.2.2 Obalování skupiny TypedArray

Aby *JShelter* předešel vykonání mikroarchitekturálních útoků, jako je například *Meltdown* nebo *Spectre*, je nutné zabránit zneužití paměti a `ArrayBuffer`, který bývá často označován jako bytové pole. Samotný `ArrayBuffer` se nijak neobaluje ani nepozměňuje, ale jelikož jsou k práci s ním potřeba buď objekty typu `TypedArray` nebo `DataView`, které představují reprezentaci binárních dat ve specifických formátech, tak se zabezpečovací změny provádějí právě s nimi.

`TypedArray` je prototyp pro skupinu 9 objektů, z nichž je každý obalen objektem `Proxy`, se stejnými obslužnými funkcemi i referencí na jejich konstruktér. Pokud si uživatel zvolí, že chce své zabezpečení zvýšit, funkce `offsetF()` vygeneruje náhodné vstupy, které každý index namapují na jiné místo v paměti a tím pádem se každý prvek ze vstupu přesune na nový index.

`DataView` poskytuje přistupování k číslům a jejich nastavení na určitém ofsetu v paměti. Jednotlivé indexy jsou stejně jako u `TypedArrays` vypočítávány pomocí funkce `offsetF()`.

Více informací k obalování skupiny TypedArray je k dostání v bakalářské práci Petera Horňáka [8].

4.3 Obalovaná API

V následujícím seznamu jsou popsané jednotlivé kategorie funkcí a metod prohlížeče, které je *JShelter* schopný omezovat [10] a které nebyly zmíněny výše u strategií v kapitole 4.1.

- stav baterie
- *DOM API*
- metody *ECMA*
- metody *HTML*
- `NavigatorPlugins`
- senzory
- virtuální realita
- *WebAudio, WebGL a Web XR*

Kapitola 5

Návrh webové prezentace

V této kapitole je popsaný návrh webové prezentace pro projekt *JShelter*. Jeden z hlavních cílů jejího vytvoření je, aby ji pochopila široká veřejnost, tudíž je nutné, aby byla jednoduchá, srozumitelná a především napsaná v angličtině. Kvůli těmto požadavkům bude nutné následné důrazné testování jak na lajcích v oblasti informačních technologií, tak také na odbornících.

Důvodem tvorby nové prezentace byl fakt, že ta původní[1] nebyla uživatelsky přívětivá. Vypisovala pouze názvy obalovaných metod a jejich výsledek bez jakéhokoliv vysvětlení či objasnění zobrazovaných dat, jak si lze všimnout na obrázku 5.1. Také se kvůli spojení projektu s *Free Software Foundation* vytvořil nový web pro popsání jeho funkčnosti, dokumentace a průběhu vývoje. Stará testovací stránka na novou platformu přenesena nebyla a vývojáři projektu se na ni přestali odkazovat, jelikož vzhledově již nadále nezapadala a kvůli blogovému způsobu vizualizace obsahu se ani jako celek přenést nedala.

Pro přenos by tudíž bylo nutné jednotlivé zobrazované obalované metody rozdělit na několik logicky souvisejících částí a ty následně implementovat jako jednotlivé stránky. Na každou by se následně mělo umístit objasnění na ní zobrazované metody či metod a jakým způsobem pomáhá jejich obalení v otázce zvýšení bezpečnosti při pohybu na webu. Vysvětlení by mělo být jednoduché, stručné ale zároveň dostatečné výstižné, aby návštěvník nemusel číst velké množství textu, ale zároveň dostatečně pochopil význam obalení dané *API*.

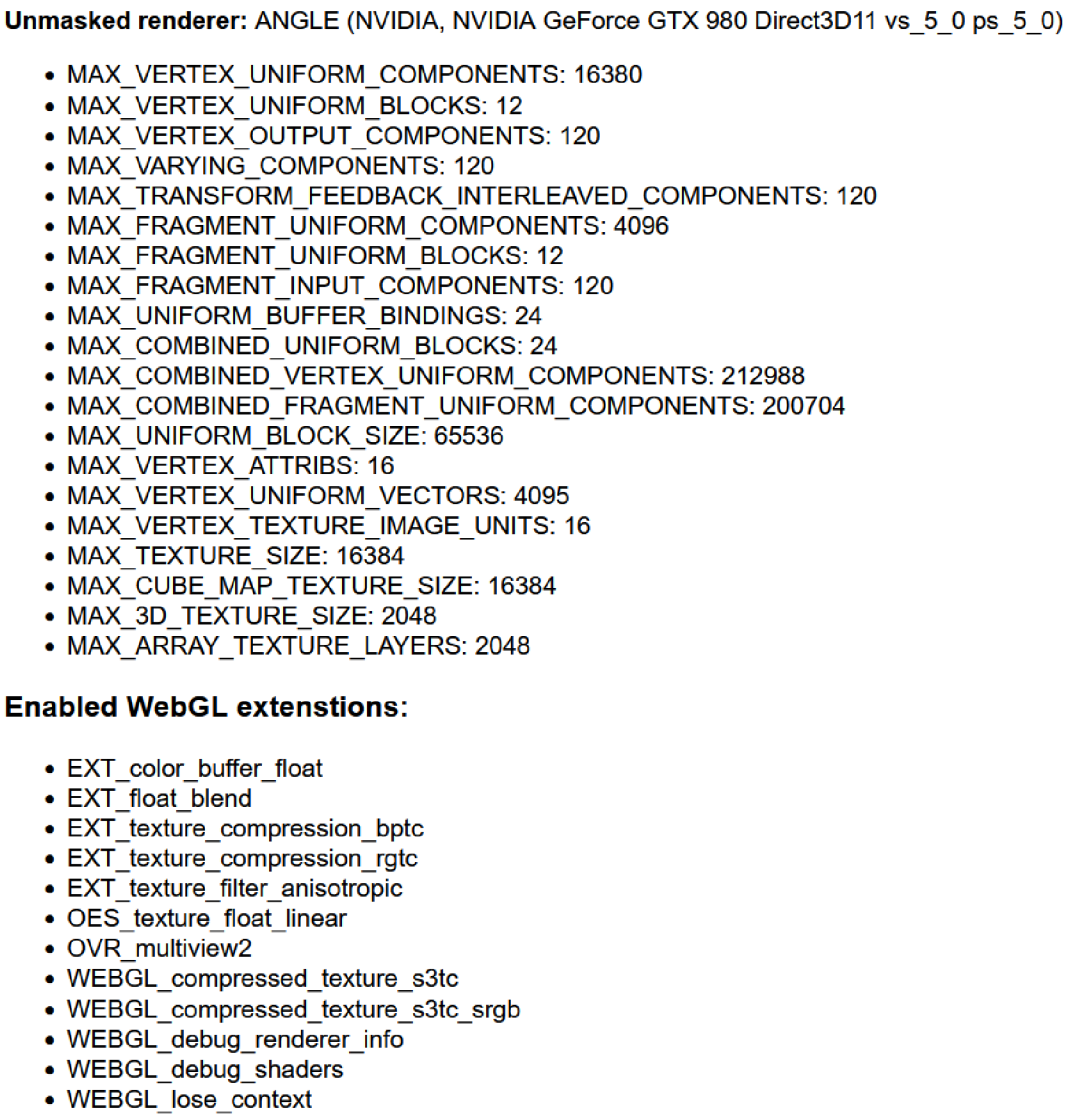
Některé metody však není možné jednoduše vysvětlit nebo je dokonce ani názorně ukázat, tudíž by se musel jejich počet snížit a uživatelům zobrazit jenom ty, u kterých je větší šance, že pro ně budou relevantní v tom smyslu, aby pochopili, že jejich neobalení může působit významnou hrozbu pro jejich bezpečnost a anonymitu.

Dalším nezbytným krokem by bylo provázení uživatele vyzkoušením si funkčnosti *JShelteru* v praxi. Pokud by nebyl vyzván, aby si otestoval u každé *API* různé úrovně zabezpečení, nebylo by jisté, zda by ukázka proběhla tak, jak byla zamýšlena a návštěvník by nemusel správně pochopit důvod obalení a jeho přínos.

V této kapitole je popsáno základní rozhraní, vzhled a princip nové testovací stránky a jejího začlenění do šablony již existujícího webu. Dále je zde zmíněn způsob její navigace

[1] Stará testovací stránka https://polcak.github.io/jsrestrictor/test/test.html

a nastíněna hlavní myšlenka ukázek jednotlivých funkcí společně s její hlavní stránkou, která bude fungovat jako rozcestník. Následně zde budou vysvětleny principy při návrhu ikon funkcí a dalších na stránce se vyskytujících obrázků a v poslední sekci bude popsáno průběžné testování kvality návrhu, které vedlo k jeho zdokonalení a předešlo možným chybám a špatným praktikám, které by jinak byly odhaleny až při finálním testování.



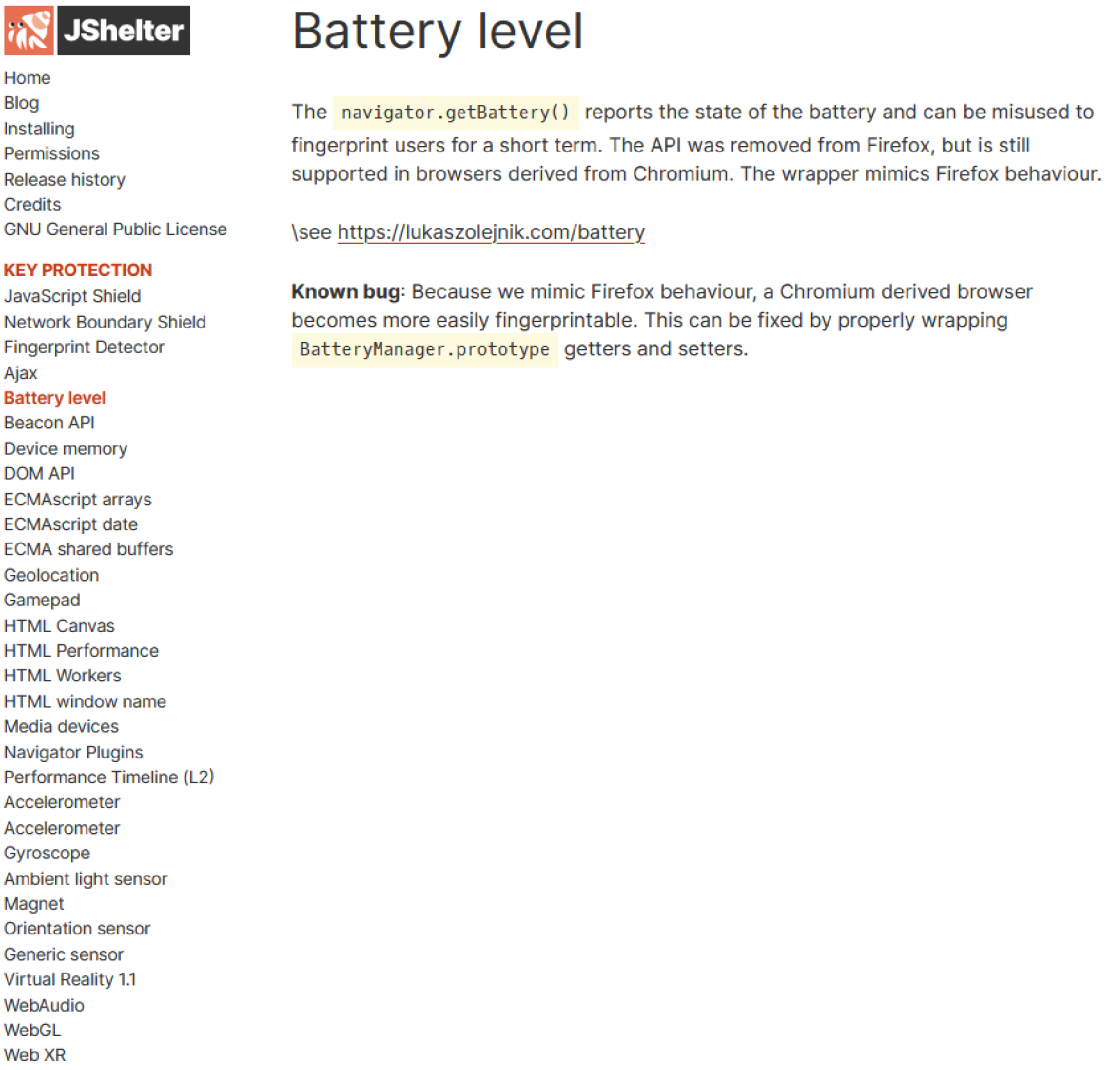
Obrázek 5.1: Vzhled staré testovací stránky

5.1 Základní rozhraní a vzhled

Návrh probíhal formou implementace jedné pokusné stránky a je v něm největší důraz kladen na minimalistický a přehledný vzhled. Jelikož je důležité začlenění vytvořené prezentace do již existujících webových stránek projektu *JShelter*, jejichž vzhled lze vidět na obrázku 5.2, nenachází se zde mnoho možností v rámci barevného schématu. Je nutné dodržet základní tři barvy, bílou, černou a oranžovou, které se v závislosti na světlém či tmavém módu pouze prohodí.

S existující levou částí stránky (navigační menu) je nutno v rámci návrhu počítat, jelikož musí být vždy součástí každé podstránky, kvůli zachování kontinuity vzhledu, ale také aby se uživatelé rychle dostali tam, kam chtějí a nemuseli se přes testovací stránku proklikávat vždy až na její začátek. Spodní část stránky (patička) návrh žádným způsobem neomezuje. Nutností každé stránky je kvůli použitému rámci také její nadpis v horní části.

Také se kvůli blogovému stylu nových webových stránek musí počítat s maximální šířkou každé ukázky tak, aby zůstaly stálé elementy vizuálně podobné a vzhled celé stránky byl pěkný. Toto omezení se nejvíce týká vzdálenosti prvků od krajů obrazovky. Existující webová prezentace má fixně danou vzdálenost levého menu od okraje, tudíž je nutné, aby se minimálně podobná dodržela i na straně pravé. Tato skutečnost bude poměrně omezující, jelikož se tímto pracovní plocha zmenší přibližně na polovinu šířky obrazovky. Vzhled stránky díky tomuto sice bude působit lepším dojmem, avšak funkčnost a jednoduchost začlenění jednotlivých ukázek funkcí do návrhu se tím poměrně ztíží.



Obrázek 5.2: Vzhled nové stránky projektu

5.2 Navigace

Navigace po webové prezentaci bude fungovat na principu klikání na vybranou ikonu či tlačítko, které bude vždy dostatečně výrazné a intuitivní, aby uživateli hned bylo jasné, na co má kliknout a kam se díky tomu dostane.

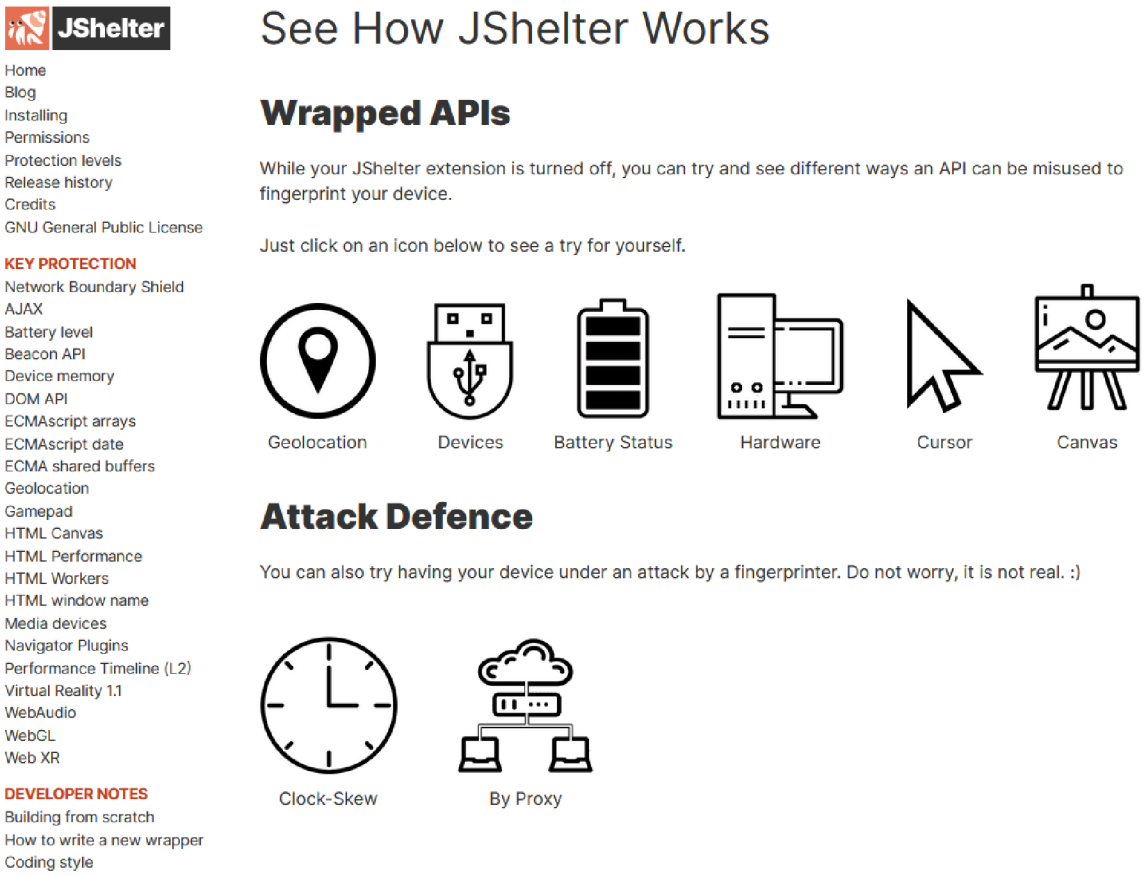
Pro jednoduchost bude na každé stránce funkcí pouze možnost vrátit se zpátky na rozcestník, jelikož jak se ukázalo v průběžném testování, více možností může uživatelům připadnout matoucí a ani vzhledově by nebylo vhodné, kdyby stránka byla předimenzovaná příliš mnoha elementy, které by přímo s danou ukázkou vybrané funkce nesouvisely.

5.3 Hlavní testovací stránka

Úvodní stránka webové prezentace se bude skládat ze 2 částí, avšak jedna z nich bude vždy skrytá, tudíž budou působit dojmem 2. První bude od počátku viditelná a uživatel zde bude vyzván, aby si rozšíření webového prohlížeče *JShelter* nainstaloval, ale ponechal vypnuté aby si nejdříve vyzkoušel, jak se jeho prohlížeč chová standardně, což bude vyobrazeno pro lepší názornost obrázkem. Dále zde bude muset potvrdit, že opravdu není zapnuté, aby se dostal na skrytou část a mohl pokračovat. Pokud uživatel *JShelter* nainstalovaný nemá, budou zde umístěny také ikony jednotlivých webových prohlížečů, které *JShelter* nabízí. Kliknutím na danou ikonu je uživatel přesměrován do obchodu, kde si může rozšíření stáhnout a nainstalovat. Návrh první části je k vidění na obrázku 5.3

Druhá část, vyobrazena na obrázku 5.4, bude fungovat jako rozcestník, ikony funkcí budou rozděleny do 2 skupin. Horní bude představovat obalovaná *API* a spodní obranu proti útokům. U každé skupiny bude krátký popis, který uživatele navede, aby si vyzkoušel, jaké informace o jeho zařízení je prohlížeč si schopný zjistit a také, co jej po kliknutí čeká.

Obrázek 5.3: Vzhled první části hlavní stránky



Obrázek 5.4: Vzhled druhé části hlavní stránky

5.4 Ukázky funkcí

Jednotlivé stránky funkcí se budou vždy držet stejné struktury. Nejdříve na nich bude popsáno, co se díky nim dá zjistit a jakým způsobem se takové informace dají zneužít pro zisk otisku webového prohlížeče. Dále zde budou názorně vypsána data, které jejich *API* zjistila.

Pokud se bude jednat o interaktivní ukázku, bude uživateli popsáno, co přesně má udělat, aby si chování *API* vyzkoušel. Tento přístup se bude kvůli udržení pozornosti upřednostňovat, jelikož pouhé čtení si informací a koukání na výsledky by mohlo vést ke snížení zájmu v ukázkách pokračovat.

Na konec bude uživatel vyzván, aby si zapnul rozšíření *JShelter* a vyzkoušel si jednotlivé úrovně zabezpečení, které budou vypsány níže i s jejich předpokládaným efektem na chování *API*.

5.5 Ikony a obrázky

Vizuální prvky jsou mezi tím prvním, čeho si uživatel při přístupu na webovou stránku všimne. Je proto důležité zvolit takové, aby upoutaly jeho pozornost, ale aby zároveň nebyly příliš rušivé. Návrh počítá s možností, že na stránku budou přistupovat uživatelé s různými vzhledy prohlížeče. Jelikož se šablona, do které se návrh má začlenit, mění v závislosti na světlém či tmavém zobrazení, je nutné tuto skutečnost zohlednit v návrhu ikon a obrázků.

Obrázky s bílým pozadím v tmavém zobrazení působí rušivě, tudíž musíme v závislosti na vzhledu prohlížeče obrázky zaměnit. U ikon návrh počítá s jinou možností a to s tím, že se vytvoří takové, aby i bez záměny mohly být kvalitně a viditelně zobrazeny v jakémkoliv motivu. Tohoto docílíme tím, že okraj ikony necháme černý, střední část zvolíme bílou a vnitřní část bude znovu černá. Tímto způsobem je jisté, že se správně zobrazí v jakémkoliv motivu.

5.6 Průběžné testování kvality návrhu

Průběžné testování kvality je jednou z nejdůležitějších náležitostí při tvorbě návrhu. Dá se díky němu předejít komplikacím předělávání nevyhovujícího finálního produktu, které může být často zdlouhavé a náročné. Také je důležité si vybrat širší spektrum testerů, jelikož něco, co se líbí jednomu, se nemusí líbit druhému nebo dokonce ani většině, tudíž je důležité umět udělat kompromis a nebo zanedbat hlas jednotlivce, pokud je se svým názorem v menšině.

Testování probíhalo na zástupcích 3 věkových kategorií z řad přátel a rodiny. V rozmezí 20 až 30 let byli vybráni tři testeři, další tři od 31 do 60 let a poslední dva ze skupiny 60+ let.

Celý proces probíhal buďto naživo nebo přes e-mailovou korespondenci, kde byly kladeny připomínky ke vzhledu návrhu a následně konzultovány možnosti jejich oprav. Nejčastější problém byla názornost ikon jednotlivých funkcí, což bylo vyřešeno výběrem lepších a zároveň také přidáním jejich popisků. Často se objevily odlišnosti názorů u velikosti mezer mezi jednotlivými komponenty. Zde musel být použit kompromis a byla vybrána vzdálenost aritmeticky vypočítaná ze všech navržených. Zbylé připomínky se většinou týkaly barevného odlišení prvků pro lepší názornost, těmto však nebylo možné, kvůli nutnosti udržet vzhled podobný šabloně, vždy vyhovět.

Celý proces testování se několikrát opakoval do té doby, dokud nebyla většina dostatečně spokojená na to, aby si dokázala představit s výsledným návrhem po nasazení prezentace pracovat a byli si jistí tím, že pro ně žádný jeho komponent nebude rušivý či nevhodný.

Kapitola 6

Implementace

Tato kapitola vysvětluje postupy, techniky a technologie, které byly k implementaci návrhu webové prezentace *JShelteru* použity. Také zde jsou rozebrány problémy, na které jsem při ní narazil.

Na začátku je vysvětlený rámec *Pelican*, který byl kvůli začlenění do již existujících stránek projektu *JShelter* k vytvoření webové prezentace použit, a jsou zde zmíněny jeho výhody a nevýhody. Dále se zde objevuje popis způsobu vytvoření zdrojových souborů, obrázků a ikon. Jejich seznam je k dispozici v příloze A. Následně jsou zde rozepsány jednotlivé webové stránky, u kterých je vždy zobrazen výčet požadovaných souborů, popis představovaných *API* nebo útoků a způsob vysvětlení jejich chování v závislosti na nastavení *JShelteru*.

6.1 Rámec Pelican

Pelican je statický generátor webových stránek, který nepotřebuje žádnou databázi či logiku na straně serveru. Tento projekt udržuje *Justin Mayer* a ostatní členové týmu vývojářů [12].

Zdrojový kód *Pelicanem* vytvořených webových stránek je psán buďto ve *reStructuredText* nebo *Markdownu*, také se do něj dá vložit čistý *HTML* kód, který zůstane nepozměněn. Tím, že je výstup kompletně statický, je snadné takto vytvořené stránky hostovat úplně kdekoliv. Motivy lze upravit pomocí *Jinja* šablon a výsledný obsah lze mít ve více jazycích. Dále také *Pelican* podporuje zvýrazňování syntaxe kódu, importování obsahu z *WordPressu* a *RSS feedu*. Modularita je podporována prostřednictvím začlenění pluginů.

6.2 Použité soubory, obrázky a ikony

Veškeré zdrojové soubory s kódem jsou formátu `.md` a byly vytvořeny. Pro správné fungování musely být uloženy do adresáře `/jsrestrictor-master/website/content/pages`.

Všechny soubory obsahují pouze kód jazyků *JavaScript*, *HTML* a *CSS* styly. Obsahem jsou tudíž skoro stejné jako soubory `.html`, ale obsahují pouze značky od úrovně `<body>` níže. Značky vyšších úrovní doplňuje rámec automaticky. Další odlišností je povinná hlavička pro *Pelican*, ve které se nachází název stránky a datum jejího vytvoření.

Začlenění souborů `.js`, tedy těch, které obsahují zdrojový kód zobrazovaných *API* a předváděných útoků, nebylo v rámci vhodné, jelikož by se musel předělat již existující web, kvůli nainstalování nových zásuvných modulů. Tím pádem je veškerý kód vložen přes značku `<script>`. Obdobně to platí i u kaskádových stylů. Šablona automaticky nahrazuje lokálně uložené `.css` soubory jejím hlavním stylem a změny bylo nutné vynutit vložením stylů přímo do zdrojového souboru pod značkou `<style>`.

Obrázky byly vytvořeny pořízením snímku obrazovky a následném oříznutí zbytečných částí. Všechny jsou uloženy ve složce `/jsrestrictor-master/website/content/images`.

Ikony byly nejdříve získány z internetových stránek, které je poskytovaly zadarmo při dodržení jejich licenčních podmínek[1][2]. Následně jim bylo online odstraněno pozadí[3]. Dále byly upraveny v *MS Malování 3D*, kde jim byla prostřední část vyplněna bíle a tmavé části změněny na černé tak, aby jejich vzhled odpovídal návrhu. Pokud ikona nebyla dostatečně viditelná na obou základních motivech prohlížečů, byly do ní manuálně dodělány nápravné změny.

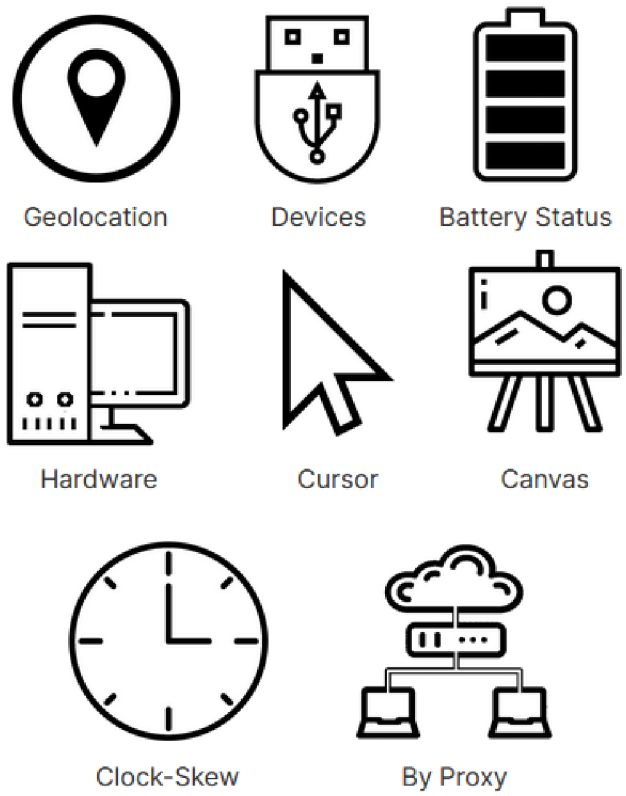
Ikony jsou použity jako navigační prvky při pohybu po webové prezentaci a také jako informativní vizuální prvky, popisující hodnoty získané jednotlivými *API*.

6.3 Hlavní stránka

Hlavní testovací stránka se skládá ze 2 částí. V té první je vloženým obrázkem vyobrazeno nastavení vypnutého *JShelteru*. Základní světlý obrázek se zaměňuje za tmavý na základě uživatelem vybraným vzhledem prohlížeče. Výměna je dosažena za pomoci `srcSet` při splnění podmínky stylu `prefers-color-scheme: dark`. Dále se uživatel dostane stisknutím tlačítka, které viditelný `<div>` první části vymění za skrytý té druhé pomoci funkce `myFunction()`, která pracuje se styly `style.display = inline` a `none`.

Druhá část funguje jako rozcestník pro ukázky jednotlivých *API* a webových útoků. Navigaci zařizují odkazy jako ikony viditelné na obrázku 6.1.

[1] Dostupné na: https://pngtree.com/legal/license-terms
[2] Dostupné na: https://www.pngkey.com
[3] Dostupné na: https://www.remove.bg



Obrázek 6.1: Vzhled jednotlivých použitých navigačních ikon

6.4 Stránky jednotlivých demonstrací

V následujících podsekcích jsou popsána jednotlivá naimplementovaná *API* a webové útoky. Je zde zmíněna jejich funkčnost a nastíněno vysvětlení jejich důležitosti v rámci bezpečného pohybu v síti internet tak, aby jejich význam byl pro uživatele lehce pochopitelný.

U každé z prezentovaných *API* jsou také vyobrazeny ikony, které pro názornost používají, a pokud byl jejich kód alespoň z části převzat, jsou zde zmíněni původní autoři.

6.4.1 Geolokace

V horní části stránky je uživateli zdůrazněno, že jeho poloha může v rukou útočníka představovat nebezpečí. Dále je naveden, aby si *API* geolokace vyzkoušel. To udělá stisknutím tlačítka, které na obrazovku vykreslí za pomoci `<iframe>` mapu ze stránky *Open Street Map* s jeho polohou. Po jejím zobrazením se uživateli nabídne tlačítko, že svoji polohu může aktualizovat, kdyby došlo k její změně. Jelikož zařízení bez GPS modulu není schopno reportovat svou přesnou geolokaci, je tato skutečnost uživateli objasněna.

Ve spodní části stránky se nachází upozornění, že při doporučeném nastavení ochrany *JShelter* obaluje *API* geolokace a vrací znepřesněnou polohu v rámci jednotek kilometrů. Při striktním nastavení se data neposkytují žádná.

Kód byl částečně převzat a upraven v rámci licence *GPL 3.0+* od Martina Timka, Petera Marka a Libora Polčáka.

6.4.2 Výčet zařízení

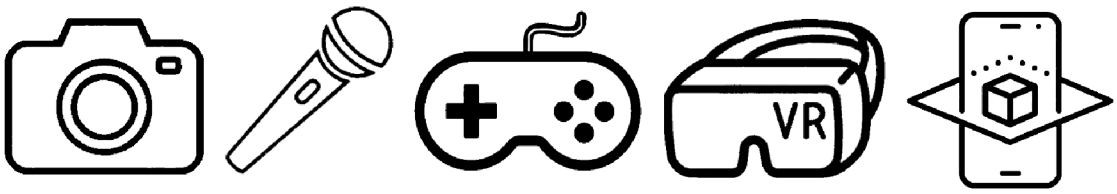
V horní části stránky je uživateli vysvětleno, že seznam zařízení i s jejich identifikací může být jedinečný a díky němu lze snadněji rozpoznat jeho otisk. Mezitím na pozadí proběhne volání metod:

`navigator.mediaDevices.enumerateDevices()` zjistí počet a druh připojených zařízení,
`navigator.getGamepads()` vrátí množství připojených gamepadů,
`navigator.getVRDisplays()` poskytne výčet displejů podporujících virtuální realitu
a `navigator.xr()` zjistí podporu mixované reality.

Při vypisování výsledků volání se vytváří seznam nalezených zařízení pomocí značek `<ul>` a `<li>`, kde místo odrážek jsou použity ikony znázorňující dané položky, které lze vidět na obrázku 6.2. Ikony jsou ale vykresleny i když se žádný prvek nenajde, kromě `navigator.mediaDevices.enumerateDevices()`, a to pouze jednou pod nadpisem sekce každé metody.

Ve spodní části stránky se nachází upozornění, že při doporučeném nastavení ochrany *JShelter* pouze zamíchá vrácený seznam zařízení a při striktním nastavení se nereportují žádná.

Kód byl částečně převzat a upraven v rámci licence *GPL 3.0+* od Libora Polčáka.



Obrázek 6.2: Ikony *API* zařízení (kamera – mikrofon – gamepad – VR – mixovaná realita)

6.4.3 Stav baterie

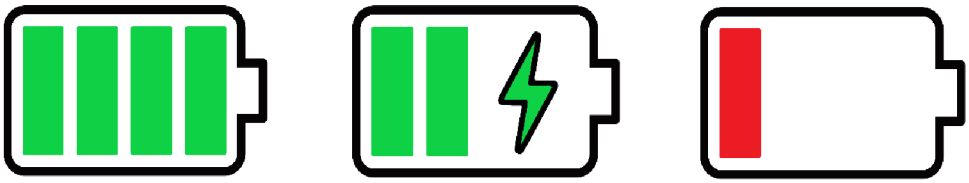
Nejdříve je uživateli vysvětleno, že kombinace získaných dat ohledně baterie jeho zařízení může ulehčit útočníkovi sbíraní jeho otisku a že některé prohlížeče, jako je například *Chrome* tuto *API* blokují standardně.

Při načtení stránky se zavolá metoda `navigator.getBattery()`, která je následně využita pro vypsání stavu nabití baterie, zda se nabijí a kolik času zbývá do jejího vybití. Před jejím použitím se ale zkontroluje, jestli je prohlížečem podporovaná, aby nedošlo k chybovému hlášení snahy o volání neexistující funkce. Pokud vrátí za pomoci `battery.level` úroveň nabití rovnou 1, znamená to, že zařízení baterii nemá a musí být neustále připojeno k síti. *API* při zjištění zbývajícího času do vybití vrací výsledek v sekundách, proto je vhodné jej před zobrazením uživateli převést na minuty. Také zde je potřeba pohlídat, jestli se `battery.dischargingTime` nerovná 0. V tomto případě by to znamenalo, že je

připojené k síti. V poslední řadě se zkontroluje jestli se nabíjí díky `battery.charging`.

Všechny výsledky jsou vizuálně vyobrazeny vedle ikon zobrazujících tyto 3 informace, které lze vidět na obrázku 6.3. V případě, že je *API* blokována, místo hodnot je uvedeno *unknown*.

Na konci stránky je uživatel poučen, že pokud je *JShelter* aktivní, tak se stránka nedozví o baterii nic.



Obrázek 6.3: Ikony *API* baterie (úroveň nabití – stav nabíjení – čas do vybití)

6.4.4 Výkon a hardware

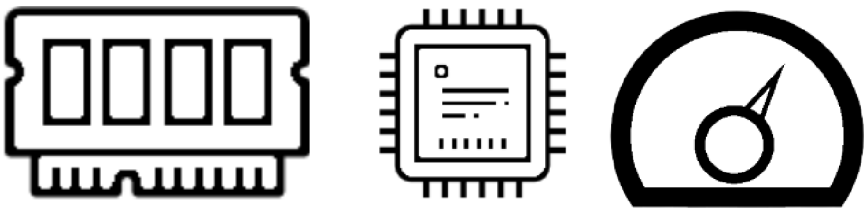
Nejdříve je uživateli vysvětleno, že informace o hardwaru jeho zařízení mohou tvořit výjimečnou kombinaci, která může být použita proti němu.

Při načtení stránky se zavolá funkce `getHW()`, která s pomocí *API* `window.navigator.deviceMemory` a `window.navigator.hardwareConcurrency` zjistí operační paměť a počet logických procesorů zařízení a následně tyto hodnoty vypíše. Také se každých 200 milisekund rekurzivně volá funkce `updatePerformanceLabelEvery()`, která počítá a vypisuje výkon zařízení.

Získané hodnoty jsou ukázány vedle příslušných ikon, které je znázorňují a lze je vidět na obrázku 6.4.

Na konci stránky je uživatel poučen, že pokud je *JShelter* aktivní s doporučeným nastavením, změní se reportované hodnoty paměti i procesorů a znepřesní se výkon. Se striktním nastavením se tyto hodnoty ještě více oddálí skutečnosti.

Kód byl částečně převzat a upraven v rámci licence *GPL 3.0+* od Martina Timka.



Obrázek 6.4: Ikony *API* hardwaru (operační paměť – logické procesory – výkon)

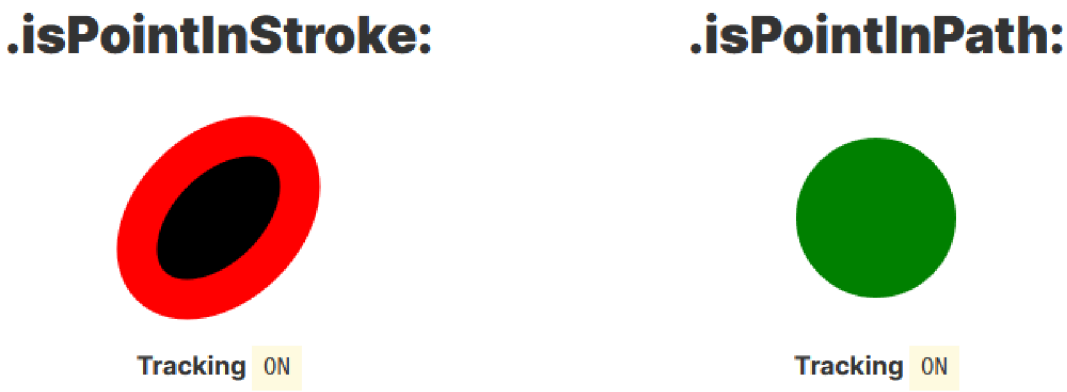
6.4.5 Kurzor myši

Na začátku stránky je popsána *API* sledující pohyb myši a uživatel je upozorněn, že pokud poloha kurzoru je zaznamenávána, může se stát obětí cílených reklam či jiných marketingových aktivit.

Dále je uživatel vyzván, aby přejel kurzorem myši přes dva vykreslené objekty pomoci `canvas.fill()` a `canvas.stroke()`. Funkcemi `isPointInStroke` a `isPointInPath` je následně sledováno, zda se v nich kurzor nachází. Pokud ano, je celý objekt dočasně překreslen z červené barvy na zelenou. Při načtení stránky se provedou kontrolní testy povolení *API*, jejichž výsledek je zobrazen pod každým objektem, což lze vidět na obrázku 6.5.

Na konci stránky je uživatel poučen, že pokud je *JShelter* aktivní s doporučeným nastavením, *API* zůstává neobalená a se striktním je zablokována a tím pádem nefungují různé po najetí rozbalující se prvky.

Kód byl částečně převzat a upraven v rámci licence *GPL 3.0+* od Martina Timka a Matúša Švancára.



Obrázek 6.5: Grafické prvky pro názornost *API* `isPointIn`

6.4.6 Plátno

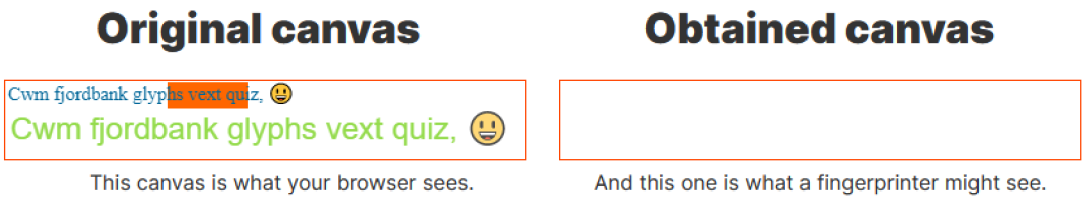
Na začátku stránky je uživateli vysvětleno, že webová stránka může každé jeho kliknutí myší zaznamenat tím, že jej vykreslí do plátna, které může následně zanalyzovat a získat informace o jeho pohybu v rámci dané stránky.

Pod vysvětlením je daná *API* nabídnuta k vyzkoušení. Vytvoří se nové plátno, do kterého může uživatel za pomocí tlačítek vykreslovat funkcemi `writeLineToCanvas()` úsečku, `writeCircleToCanvas()` kružnici, `writeTextToCanvas()` a `writeOutlineTextToCanvas()` text, `writeGradientToCanvas()` gradient a `writeImageToCanvas()` obrázek. Následně je vyzván ke stisknutí tlačítka, které plátno překreslí daty, které z něj šly vyčíst. Také se na stránce objevuje testovací plátno, viditelné na obrázku 6.6, které již data obsahuje a vedle něj další, jenž ukazuje z něj zjistitelná data.

Na konci stránky je uživatel poučen, že *JShelter* obaluje *API* pouze ve striktním nasta-

vení a to tak, že zabraňuje odeslání dat webovým stránkám.

Kód byl částečně převzat a upraven v rámci licence *GPL 3.0+* od Martina Timka a Matúša Švancára.



Obrázek 6.6: Ukázka nepodařeného zisku otisku

6.4.7 Odchylka vnitřních hodin

Implementace ukázky útoku za pomoci odchylky vnitřních hodin se skládá pro větší pochopení a vizuální kvalitu ze dvou stránek. Na té první je uživateli za pomoci názorných snímků obrazovky vysvětleno, jak bude ukázka probíhat. Je zde umístěn krátký popis odchylky a zdůrazněno, že čím striktnější nastavení *JShelteru*, tím je možnost získání odchylky útočníkem nižší až žádná.

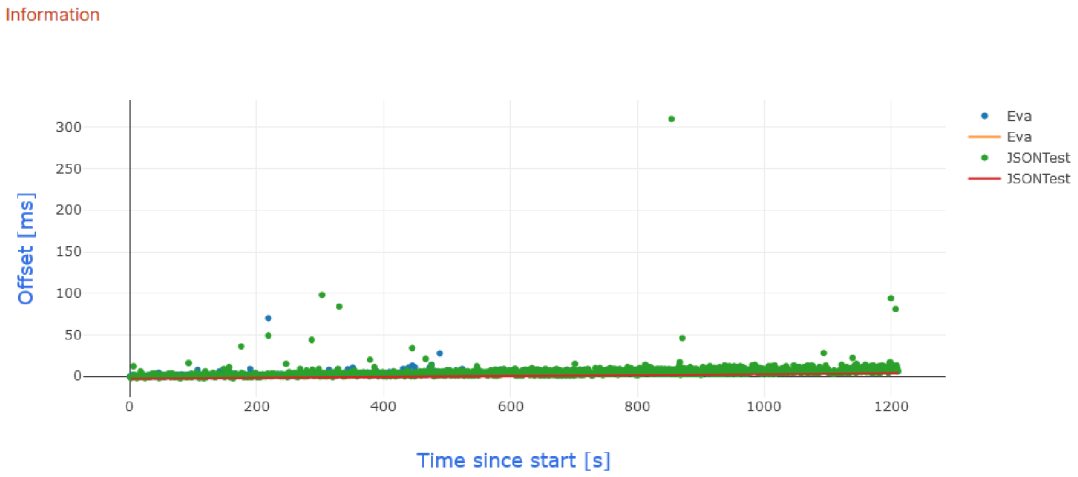
Na druhé stránce se pro kontrolu v horní části zobrazuje objekt `Date`, který se každou milisekundu aktualizuje funkcí `updateClock()`, která je při načtení stránky zavolána funkcí `initClock()`.

Následují výpočty odchylek oproti dvěma serverům. Jakmile je dostatečně odhadnutá, její výsledek se vypíše pod okno, které zobrazuje jednotlivá časová razítka uživatelova zařízení oproti těm získaných se serveru. Finální odchylka se zobrazí na stejném místě jako odhadnutá, ale její hodnota bude kvůli odlišení podtržená. Odchylky se vypočítávají přímo na serveru pana Jireše, proto je hlavní kód uložen mimo adresář webové prezentace na `http://www.stud.fit.vutbr.cz/~xjires02/pcf.js` a `http://www.stud.fit.vutbr.cz/~xjires02/fce.js`.

Interaktivní graf s časovými razítky se začne vykreslovat po 90 sekundách od začátku měření knihovnou *Plotly* s verzí `1.54.1`, který lze vidět na obrázku 6.7.

V poslední části stránky se zobrazují předchozí měření oproti oboum serverům, ale až při druhém úspěšném měření.

Kód byl převzat a upraven v rámci licence *GPL 3.0+* od Martina Timka a Matúša Švancára.



Obrázek 6.7: Graf odchylek vykreslovaný knihovnou *Plotly*

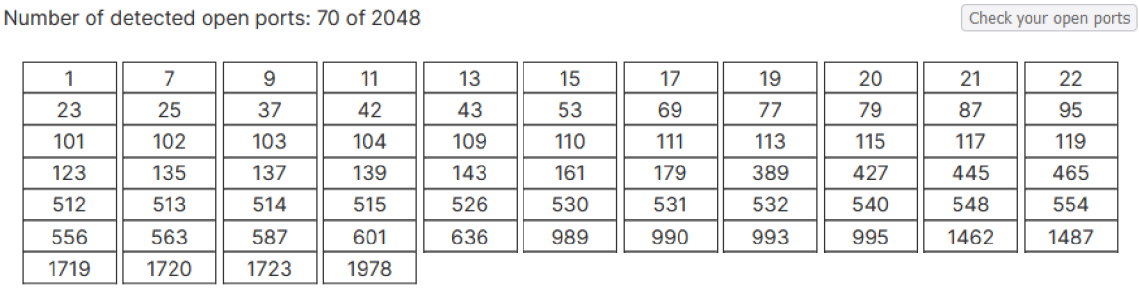
6.4.8 Prohlížeč jako prostředník

Stránka ukázky útoku za zneužití prohlížeče jako prostředníka nejdříve uživateli vysvětlí, že pokud by útočník mohl zjistit, jaká zařízení se v jeho interní síti nacházejí, šlo by o zásadní bezpečnostní hrozbu.

Dále je zde vypsána *IP* adresa jeho routeru jako odkaz, který jej zavede na stránku s nastavením. Seznam nejpoužívanějších adres byl získán online [18]. Pro její zjištění je opakovaně volána funkce `check_router()`, jejímž parametrem je hádaná adresa routeru. Volání se opakuje pro celý seznam předpokládaných adres.

Druhá část ukázky se skládá ze zasílání dotazů na jednotlivé porty běžící na `127.0.0.1 (localhost)`. Pokud je port otevřený, jeho číslo se propíše do dynamické tabulky, jenž lze vidět na obrázku 6.8 , která po dokončení bude plnit funkci seznamu. Dotazy jsou zasílány funkcí `scan()`, jejímž parametrem je číslo portu. Prochází se pouze porty 1 – 2048, kvůli náročnosti na výpočetní a síťovou kapacitu jednotlivých prohlížečů. Mezi jednotlivými kontrolami otevřenosti portů se čeká 50 ms.

Funkce `scan()` posílá požadavky na zobrazení prázdného obrázku na jednotlivé porty daných *IP* adres a nerozlišuje rozdíl mezi `image.onerror` a `image.onload`. Pokud požadavek nevyprší, znamená to, že port dané adresy je otevřený.



Obrázek 6.8: Dynamická tabulka zobrazující otevřené porty

6.5 Integrace do existujících webových stránek

Integrace do již existujících webových stránek mezinárodního projektu *JShelter* probíhala v rámci služby *GitHub*[4]. Nejdříve byl stažen nejaktuálnější repozitář, do kterého byly provedeny příslušné změny, jako je přidání zdrojových souborů a obrázků s ikonami.

Následně byl zadán požadavek na sloučení nově přidaného obsahu s tím stávajícím, takzvaný `pull request`, který čeká na schválení hlavním vývojářem projektu, doktorem Liborem Polčákem. V rámci tohoto procesu jsou zjišťovány nedostatky, které jsou co nejdříve opravovány, aby byl přidaný obsah dle požadavků a mohl být kompletně začleněn.

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

Kapitola 7

Testování

Testování je nedílnou součástí každého projektu, především takového, který bude využívat veřejnost. To, co vývojáři může připadat v pořádku a dostatečně srozumitelné, nemusí veřejnost pochopit. Proto je nutné testovat návrh produktu průběžně při implementaci, aby finální testy neodhalily nedostatky, které by se těžko opravovaly. Výsledný produkt by měl před nasazením projít rukama co největšího počtu lidí, aby bylo jisté, že jeho kvalita splňuje požadavky.

V této kapitole je nejdříve zmíněno testování webové prezentace v prostředí různých internetových prohlížečů. Především je zde kontrolována funkčnost jednotlivých *API*, jelikož ne všechny musí být vývojáři prohlížečů povoleny. Dále následuje představení plánu testování na uživatelích společně s připraveným dotazníkem. V poslední části jsou výsledky testování s jejich celkovým zhodnocením a na základě nich navrhnuty další možnosti vývoje.

7.1 Testování funkčnosti na různých webových prohlížečích

Před uživatelským testováním je nutné funkčnost prezentace vyzkoušet lokálně na různých webových prohlížečích, jelikož některá *API* mohou být blokována. Na takové musí být testeři upozorněni, jelikož jinak by si mohli myslet, že *JShelter* nebo webové stránky fungují špatně a znehodnotilo by to výsledné hodnocení.

Testování probíhalo na těch nejběžnějších prohlížečích jako jsou *Mozilla Firefox*, *Google Chrome*, *Opera* a *Microsoft Edge*. U každého byly vyzkoušeny jednotlivé ukázky a bylo zjišťováno, jestli jejich funkčnost zůstala taková, jaká byla zamýšlena a následně také, zda se vizuální stránka v jejich prostředí nezměnila.

Pro tyto účely byly použity následující verze jednotlivých prohlížečů.

- *Mozilla Firefox* – verze 99.0.1 (64 bitů)
- *Google Chrome* – verze 101.0.4951.54 (64 bitů)
- *Opera* – verze 85 (64 bitů)
- *Microsoft Edge* – verze 101.0.1210.32 (64 bitů)

Kompabilita s různými prohlížeči				
Název *API* nebo ukázky	Firefox	Chrome	Opera	Edge
Geolokace	ANO	ANO	ANO	NE
Zařízení	ANO	ČÁSTEČNĚ	ČÁSTEČNĚ	ČÁSTEČNĚ
Baterie	NE	ANO	ANO	ANO
Hardware	ANO	ANO	ANO	ANO
Kurzor	ANO	ANO	ANO	ANO
Plátno	ANO	ANO	ANO	ANO
Odchylka hodin	ANO	ANO	ANO	ANO
Prohlížeč prostředníkem	ANO	NE	NE	ČÁSTEČNĚ

Webová prezentace byla po celou dobu jejího vývoje testována pouze v prohlížeči *Mozilla Firefox*. Zde byla její funkčnost omezena pouze *API* baterie, jelikož není od verze 52 podporovaná [1].

Prohlížeče *Google Chrome*, *Opera* a *Microsoft Edge* všechny částečně blokují *API* výčtu zařízení. Pokud se webová stránka o jejich zjištění pokusí, dostane pouze počet mikrofonů a kamer bez jejich identifikátorů. Gamepady jsou automaticky reportovány 4 a také bez jejich názvu. Informace o virtuální a mixované realitě zůstávají dostupné.

Geolokace je jako jediná omezována v prohlížeči *Microsoft Edmake ge* a to tím, že se při dotazu vrací poloha upravená v rámci jednotek až desítek kilometrů. Důvodem nepřesnosti je fakt, že pokud nemá zařízení GPS modul, využívá ke zjištění aplikaci *MAPS* zabudovanou v operačním systému *Windows 10*, u které je nutné přesnou polohu nejdříve nastavit [25].

Poslední odlišností zjištěnou při testování funkčnosti bylo blokování odesílání požadavků do lokální sítě při ukázce útoku zneužitím prohlížeče jako prostředníka. Toto se projevovalo, stejně jako u výčtu zařízení, pouze na prohlížečích *Google Chrome*, *Opera* a *Microsoft Edge*, který blokoval jenom nějaké.

7.2 Plán testování na uživatelích

Testování na uživatelích se drželo předem vytyčeného plánu. Nejdříve byl zkopírován aktuální repozitář projektu *JShelter*, do kterého byly nahrány soubory vytvořené webové prezentace.

Následně byl vybraným testerům, většinou z okruhu přátel a rodiny, poslán odkaz na aktualizovaný repozitář nebo zkomprimovaný soubor, zároveň s pokyny, jak si server s prezentací mají spustit na svém zařízení a také, jak se při testování mají chovat a čeho si mají všímat. Jeden z hlavních požadavků na ně byl, aby první část dotazníku vyplňovali průběžně po každé ukázce *API* nebo útoku, aby měli úroveň pochopení ještě čerstvě uloženou v paměti a nemuseli si jednotlivé demonstrace na konci vybavovat. Tímto způsobem byla ještě více podpořena objektivnost testování, jelikož zde nehrála roli paměť.

V rámci plánu bylo vybráno přibližně 35 testerů, kteří měli za úkol oslovit další z řad jejich přátel, aby jich bylo co nejvíce. Celkově se počítalo přibližně s 55 vyplněnými dotazníky za jeden až dva dny. K dispozici byl po celou dobu vývojář webové prezentace

pro zodpovězení případných otázek. Pokud někdo nebyl schopný lokální server s prezentací na svém zařízení rozjet, bylo domluveno testování s osobní účastí, při kterém si ji mohli vyzkoušet na vývojářově zařízení.

7.3 Uživatelský dotazník

Při tvorbě uživatelského dotazníku byl kladen důraz na jednoduchost a nenáročnost na čas. Tudíž se zaměřuje pouze na ty nejdůležitější aspekty spojené s pochopením a vizuálním vzhledem testované prezentace. Vytvářen byl pod domněnkou, že dobrovolní neplacení testeři nebudou chtít vyplňovat zdlouhavé dotazníky. Časový rozsah byl tedy stanoven na pár minut, jelikož i samotné vyzkoušení si jednotlivých funkcí *JShelteru* může být časově náročné. Například demonstrace útoku za pomoci odchylky vnitřních hodin může trvat i 15 minut a pokud by stejný čas mělo trvat i následné vyplňování otázek, mohlo by se stát, že by tester dotazník nevyplnil. Dále se také v dotazníku kvůli délce představení různých funkcí objevuje otázka, zda uživatel nějaké demonstrace nepřeskočil. Tyto by následně nemohly být objektivně hodnoceny a znepřesňovaly by výsledek testování. Celý dotazník byl vytvořen službou *Google Forms*[1] a je napsán v anglickém jazyce, jelikož se v něm nachází i celá webová prezentace a míchat různé jazyky by nebylo vhodné.

Ukázky jednotlivých otázek dotazníku jsou nastíněny níže. Pro lepší srozumitelnost je rozdělen do dvou částí. První se týká pochopení jednotlivých demonstrací a hodnocení jejich vizuální stránky a druhá obecně vyhodnocuje znalosti uživatele v oblasti informačních technologií.

Otázky první části jsou zodpovídány lineární škálou od jedné do pěti. Nižší číslo označuje horší pochopení nebo vzhled dané ukázky. Výjimku tvoří poslední položka, možnými odpověďmi u ní je název jakékoliv demonstrace a lze jich zvolit více nebo žádnou. Otázek je celkem 18 pro ukázky a úvodní stránku (dvě pro každou) a 1 pro jejich přeskočení, níže je pouze obecná ukázka, aby nemusely být všechny vypisovány.

- *Understanding – <API name>*
- *Visual – <API name>*
- *Understanding – <Attack name>*
- *Visual – <Attack name>*
- *What demonstration did you skip?*

Druhá část obsahuje pouze jednu otázku, na kterou lze odpověď v rámci lineární škály od jedné do deseti, kde nižší číslo také znamená horší odpověď.

- Your IT knowledge

7.4 Výsledky testování na uživatelích

V této sekci jsou vypsány výsledky uživatelského testování a pro lepší názornost jsou uváděny v grafech, které znázorňují aritmeticky zprůměrované hodnoty sesbírané z dotazníků.

[1] Dostupné na: https://docs.google.com/forms

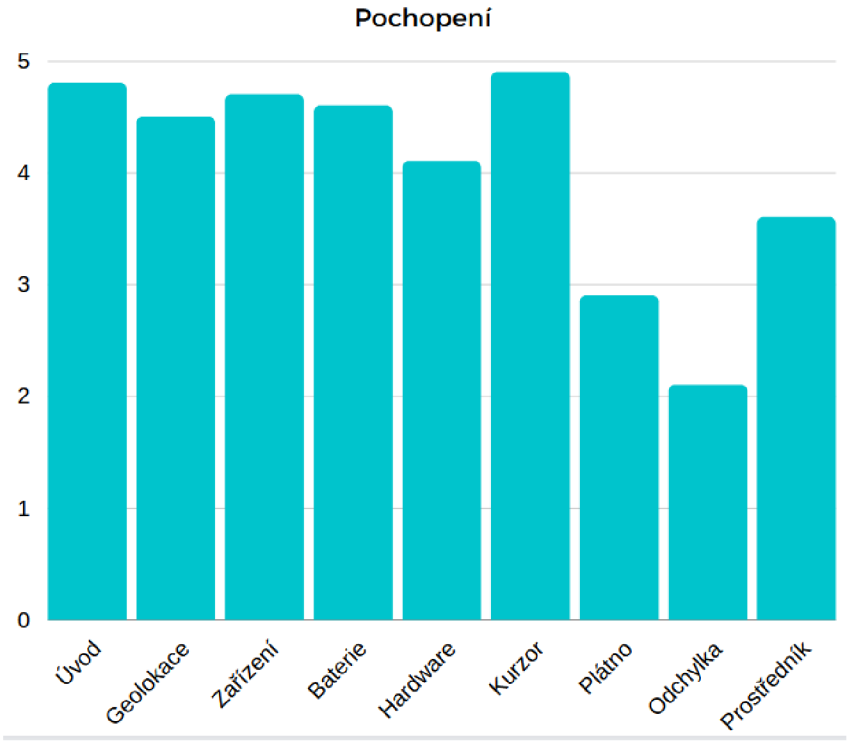
I zde platí, že větší číslo označuje lepší pochopení nebo vzhled. Jediný rozdíl je, že pokud byla ukázka přeskočena, místo hodnoty odpovědi byla započítána nula, aby tento fakt byl zohledněn ve finálním hodnocení. Nejdříve je zde zmíněno pochopení jednotlivých demonstrací či jejich přeskočení, dále jejich vzhled a nakonec zmíněna průměrná úroveň znalosti testerů v oblasti informačních technologií. Testování se zúčastnil nižší počet testerů, než bylo odhadováno, a to 37. Snížení je přičítáno nenasazením prezentace na web.

Jediná ukázka, která byla přeskočena větším počtem lidí, je demonstrace útoku za pomoci odchylky vnitřních hodin zařízení. Toto bylo očekáváno, jelikož je k pochopení časově nejnáročnější a tento fakt se promítl i do hodnocení, které má ze všech nejnižší.

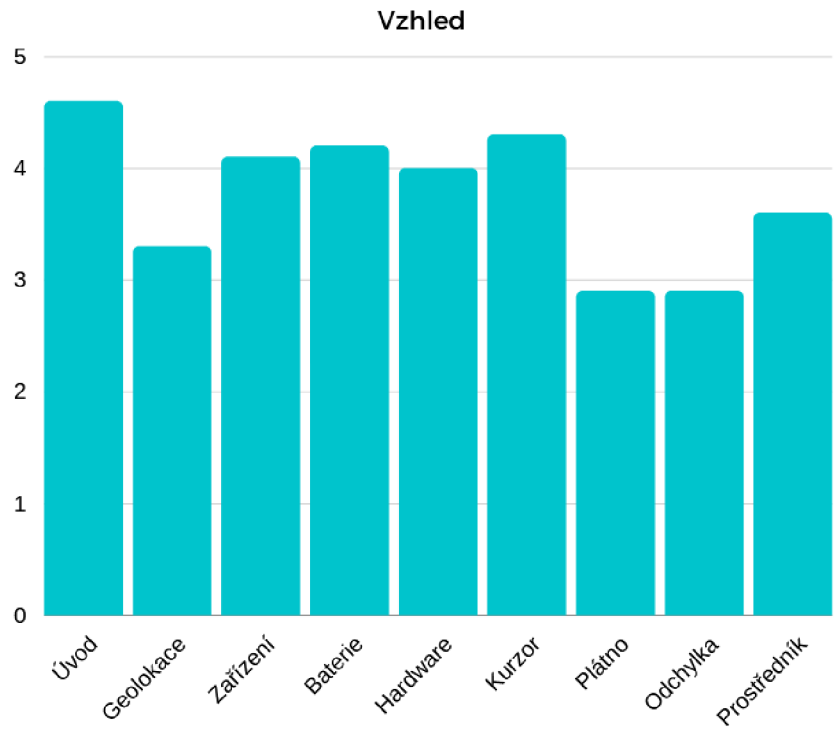
Nejvíce pochopitelné byly naopak ukázky jednotlivých *API* zároveň s úvodní stránkou, čehož si lze všimnout v grafu 7.1, jelikož zde bylo potřeba nejméně vysvětlování a jejich vyzkoušení vyžadovalo buď žádný, nebo pouze minimální čas. Výjimku tvoří *API HTML* plátna, které má hodnocení podstatně nižší.

Co se týče vzhledu, hodnocení se podstatně snížilo a pohybuje se většinou mezi hodnotami 3,0 až 4,0. Nejhůře jsou na tom znovu demonstrace útoků a mezi *API* geolokace, čehož si lze všimnout na obrázku 7.2.

Úroveň uživatelských znalostí v informačních technologiích byla ustanovena na 52 procent.



Obrázek 7.1: Graf pochopení jednotlivých demonstrací



Obrázek 7.2: Graf hodnocení vzhledu jednotlivých demonstrací

7.5 Zhodnocení a další možnosti vývoje

Testování prokázalo, že finální produkt je, i přes uživateli hlášené nedostatky, dostatečně srozumitelný a vzhledově adekvátní na to, aby byl schopen veřejnost uvádět do problematiky otisku webového prohlížeče a zároveň ji pochopitelně vysvětloval funkci webového rozšíření *JShelter*.

Všechny výsledné hodnoty, které byly vyšší než 4,0 byly považovány za vynikající. Nižší hodnoty nutně nemusely indikovat špatné vysvětlení, ale mohly souviset s obtížností danou demonstraci předvést v čase, který byl do ní uživatel ochoten investovat.

U *API HTML* plátna se ukázalo, že stávající implementace není dostatečně pochopitelná a ani vzhledem se nerovná průměru. Zde by bylo potřeba předělat celou ukázku, nejlépe na základě podrobné uživatelské zpětné vazby. Obdobně to platí u demonstrace útoku za pomoci odchylky vnitřních hodin zařízení. Problémem ale nastává fakt, že k jejímu vypočtení je potřeba delší čas, který se bohužel nedá zásadně zkrátit. Jedinou možností by bylo přemluvit uživatele, aby ji nechal spuštěnou na pozadí a následně se k ní vrátil, ale naskytávají se zde obavy, že málokdo bude mít dostatečnou trpělivost, aby potřebný čas vyčkal.

Dále se při testování přišlo na to, že při demonstraci útoku s využitím prohlížeče jako prostředníka došlo ke snížení jeho výkonu, jenž muselo být napraveno aktualizováním stránky nebo jejím zavřením.

Hodnocení vzhledu bylo velmi ovlivněno začleněním ukázek do již existujících webových

stránek a jejich šablony. Pokud by se tento vzhled nemusel dodržet, bylo by zde více prostoru ke zlepšení. Samozřejmě výsledek, který se držel kolem hodnoty 4,0, tedy 80 procent, je dostatečný. U demonstrací, které se blíží k hodnotě 3,0 (geolokace, *HTML* plátno a odchylka), by bylo vhodné přidat lepší vizuální prvky nebo dokonce celou ukázku od základu přepracovat. Obecně by hodnocení prospělo, kdyby se v prezentaci objevovalo více dynamických prvků, například u *API* baterie by se mohly jednotlivé ikony měnit v závislosti na úrovni nabití baterie nebo u geolokace by se mohla do mapy vykreslovat předešlá poloha či dokonce i rádius, ve kterém *JShelter* pozici znepřesňuje.

Úroveň znalostí testerů v oblasti informačních technologií indikuje, že výsledky mohou být zkresleny a nemusely by odpovídat hodnocení učiněným průměrným uživatelem internetu. Na druhou stranu lze tvrdit, že běžný uživatel nebude vyhledávat zabezpečující rozšíření, jelikož si bude myslet, že jej nepotřebuje.

V budoucnu by bylo vhodné provést rozšířenější testování, při kterém by bylo zodpovídáno více otázek, u kterých by se neodpovídalo pouze v rámci lineární škály, ale bylo by vyžadováno slovní ohodnocení každé ukázky. Toto řešení by vedlo k důležité zpětné vazbě, díky které by se prezentace projektu mohla vylepšit a posunout dále. Velkou výhodou by bylo zajištění více testerů, nejlépe v řádech stovek. Toto by zajistilo větší objektivnost výsledků.

Kapitola 8

Závěr

V rámci této bakalářské práce jsem se seznámil s problematikou otisku webového prohlížeče, různými způsoby jeho zneužití, ale také s možnostmi jeho využití pro zvýšení úrovně zabezpečení při pohybu v síti internet. Otisk a všechny jeho náležitosti jsou popsány v kapitole 2 a jeho zneužití bylo jedním z hlavních důvodů vzniku projektu *JShelter*, který bojuje proti jeho získávání webovými stránkami.

V kapitole 3 byly popsány jednotlivé webové útoky, kterým se *JShelter* snaží předejít. *Meltdown* a *Spectre* jsou sice již zastaralé, ale stále jsou nějaká zařízení vůči nim náchylná. Zneužití prohlížeče jako prostředníka společně s útokem za pomoci odchylky vnitřních hodin představují aktuální bezpečnostní hrozby, proti kterým se ukázalo, že *JShelter* dokáže uživatele a jejich zařízení ochránit.

Samotný *JShelter* je podrobněji rozepsán v kapitole 4, ve kterém je popsán jeho cíl a strategie, kterými bojuje za bezpečnost pohybu na webu. Tento popis se zatím nikde v takovém rozsahu neobjevil a může být následně použit pro ostatní studenty, ale i veřejnost, k podrobnějšímu seznámení se s funkčností projektu a jeho hlavní myšlenkou.

V kapitole 5 je zmíněn návrh z obecného hlediska, ale také jeho aplikování při implementaci prezentace. Nejdůležitější prvky jsou zde podrobně popsány a objevují se zde i vizuální elementy, jelikož návrh probíhal implementací jedné prototypové webové stránky, aby byly nedostatky použitých technologií odhaleny co nejdříve a nemusel se při implementaci upravovat. Návrh byl podroben po celou dobu jeho tvorby průběžnému testování, které jej vylepšilo, uvedlo správným směrem a předešlo zbytečným chybám.

V rámci implementace jsem se musel seznámit s rámcem *Pelican*, jelikož v něm jsou vytvořeny již existující webové stránky mezinárodního projektu *JShelter*, do kterých bylo nutné webovou prezentaci jeho funkcí zaimplementovat. Dále jsem musel nastudovat cizí zdrojový kód, který jsem upravil dle svých představ a vytyčených požadavků. Implementované řešení prošlo testováním a bylo shledáno dostatečně vyhovujícím pro veřejnost. Jelikož se jedná o webovou prezentaci projektu *JShelter*, který se postupně vyvíjí a rozrůstá o další funkce, bude stále potřeba ukazovat nová vylepšení a vysvětlovat jejich přínos pro bezpečnost při pohybu na webu. Samotné rozšíření webového prohlížeče tohoto projektu má nainstalováno desítky tisíc uživatelů, tudíž je jeho dosah široký.

Literatura

[1] MDN. *Battery Status API* [online]. Únor 2022 [cit. 2022-05-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Battery_Status_API.

[2] Bergmon, J. *Attacking the Internal Network from the Public Internet Using a Browser as a Proxy* [online]. Forcepoint research report, 2019 [cit. 2022-01-24]. Dostupné z: https://www.forcepoint.com/sites/default/files/resources/files/report-attacking-internal-network-en_0.pdf.

[3] Belcic, I. *What Is a Botnet?* [online]. Říjen 2021 [cit. 2022-01-24]. Dostupné z: https://www.avast.com/c-botnet.

[4] Schwarz, M. *Chrome Zero* [online]. [cit. 2022-01-24]. Dostupné z: https://github.com/IAIK/ChromeZero.

[5] Cover Your Tracks Team. *Feature Policy* [online]. [cit. 2022-04-09]. Dostupné z: https://coveryourtracks.eff.org/about.

[6] Brave Privacy Team. *Fingerprinting Defenses 2.0* [online]. Květen 2020 [cit. 2022-04-09]. Dostupné z: https://brave.com/privacy-updates/4-fingerprinting-defenses-2.0/.

[7] Eckersley, P. *A Primer on Information Theory and Privacy* [online]. Leden 2020 [cit. 2022-04-09]. Dostupné z: https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy.

[8] Horňák, P. *Přenos bezpečnostních opatření z Chrome Zero do JavaScript Restrictor*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22374/.

[9] Williams, C. *Meltdown, Spectre: The password theft bugs at the heart of Intel CPUs* [online]. Leden 2018 [cit. 2022-05-08]. Dostupné z: https://www.theregister.com/2018/01/04/intel_amd_arm_cpu_vulnerability/.

[10] Polčák, L., Bednář, M., Maone, G., Červinka, Z., Timko, M. et al. *Dokumentace JShelter* [online]. [cit. 2022-01-24]. Dostupné z: https://jshelter.org/levels/.

[11] Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M. et al. *Spectre Attacks: Exploiting Speculative Execution*. Leden 2018. Dostupné z: https://www.researchgate.net/publication/322253254_Spectre_Attacks_Exploiting_Speculative_Execution.

[12] Mayer, J. *Pelican Static Site Generator, Powered by Python* [online]. Březen 2022 [cit. 2022-05-08]. Dostupné z: https://blog.getpelican.com/.

[13] Williams, C. *Kernel-memory-leaking Intel processor design flaw forces Linux, Windows redesign* [online]. Leden 2018 [cit. 2022-04-09]. Dostupné z: https://www.theregister.com/2018/01/02/intel_cpu_design_flaw/.

[14] Pohner, P. *Detekce podezřelých síťových požadavků webových stránek*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22377/.

[15] Polčák, L. a Franková, B. Clock-Skew-Based Computer Identification: Traps and Pitfalls. *Journal of Universal Computer Science*. 2015, sv. 21, č. 9, s. 1210–1233. ISSN 0948-6968. Dostupné z: https://www.fit.vut.cz/research/publication/10725.

[16] Polčák, L., Bednář, M., Maone, G., Červinka, Z., Timko, M. et al. *JShelter* [online]. [cit. 2022-01-24]. Dostupné z: https://jshelter.org/.

[17] Pugliese, G., Riess, C., Gassmann, F. a Benenson, Z. *Long-Term Observation on Browser Fingerprinting: Users' Trackability and Perspective* [online]. Privacy Enhancing Technologies Symposium, 2020 [cit. 2022-05-08]. Dostupné z: https://petsymposium.org/2020/files/papers/issue2/popets-2020-0041.pdf.

[18] TechSpot Staff. *A List of Common Default Router IP Addresses* [online]. Prosinec 2017 [cit. 2022-05-03]. Dostupné z: https://www.theregister.com/2018/01/02/intel_cpu_design_flaw/.

[19] Saloň, M. *Detekce metod zjišťujících otisk prohlížeče*. Brno, CZ, 2021. Diplomová práce. Brno University of Technology, Faculty of Information Technology. Dostupné z: https://www.fit.vut.cz/study/thesis/23645/.

[20] Schwarz, M., Lipp, M. a Gruss, D. JavaScript Zero: Real JavaScript and Zero Side-Channel Attacks. In:. Leden 2018. DOI: 10.14722/ndss.2018.23105. Dostupné z: https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_07A-3_Schwarz_paper.pdf.

[21] Schwarz, M., Maurice, C., Gruss, D. a Mangard, S. *Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript*. Leden 2017. 247-267 s. ISBN 978-3-319-70971-0. Dostupné z: https://graz.pure.elsevier.com/en/publications/fantastic-timers-and-where-to-find-them-high-resolution-microarch.

[22] Schwarz, M., Lackner, F. a Gruss, D. JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits. *26th Annual Network and Distributed System Security Symposium, NDSS 2019*. The Internet Society. 2019, [cit. 2022-01-24]. Dostupné z: https://www.ndss-symposium.org/ndss-paper/javascript-template-attacks-automatically-inferring-host-information-for-targeted-exploits/.

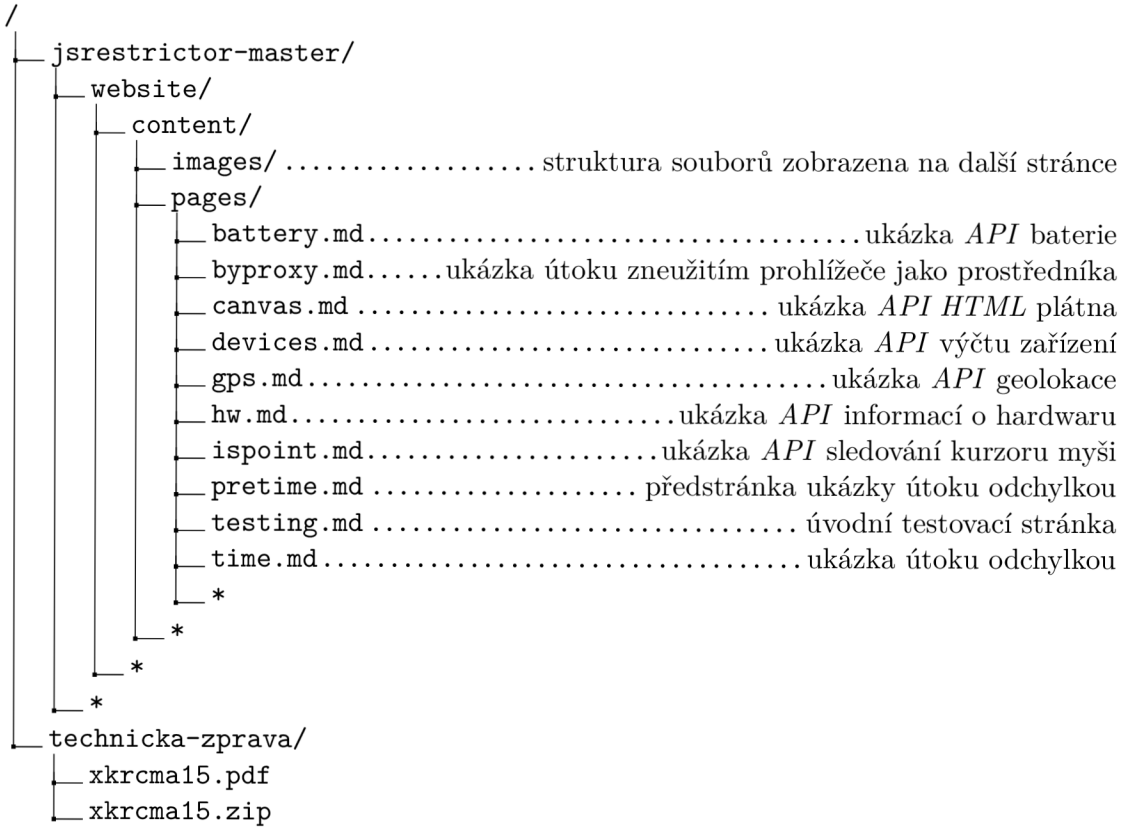
[23] SUCCESS Q. *Historical yearly trends in the usage of client-side programming languages for websites* [online]. 2019 [cit. 2022-01-24]. Dostupné z: https://w3techs.com/technologies/history_overview/client_side_language/all/y.

[24] Švancár, M. *Přenos bezpečnostních opatření z prohlížeče Brave do rozšíření JavaScript Restrictor*. Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23310/.

[25] SkipperGreg. *Geolocation wrong (really wrong) in Edge but correct in Chrome* [online]. Duben 2021 [cit. 2022-05-09]. Dostupné z: https://techcommunity.microsoft.com/t5/discussions/geolocation-wrong-really-wrong-in-edge-but-correct-in-chrome/m-p/2269351/highlight/true#%23M46237.

Příloha A

Obsah přiloženého paměťového média



/jsrestrictor-master/website/content/images/

- bat.png ... ikona baterie
- bat_down.png ... ikona stavu baterie
- camera.png ... ikona kamery
- canvas.png ... ikona *HTML* plátna
- clock.png ... ikona útoku pomocí odchylky hodin
- cpu.png ... ikona procesoru
- cursor2.png ... ikona kurzoru myši
- devices.png ... ikona výčtu zařízení
- dis.png ... ikona vybíjecí se baterie
- firefox.png ... ikona prohlížeče *Firefox*
- gamepad.png ... ikona gamepadu
- graf.png ... obrázek vykresleného grafu
- hw.png ... ikona hardwaru zařízení
- char.png ... ikona nabíjející se baterie
- chrome.png ... ikona prohlížeče *Chrome*
- jshelterOFF.png ... světlý obrázek nastavení *JShelteru*
- jshelterOFFdark.png ... tmavý obrázek nastavení *JShelteru*
- location.png ... ikona geolokace
- memory.png ... ikona paměti zařízení
- micro.png ... ikona mikrofonu
- mr.png ... ikona mixované reality
- opera.png ... ikona prohlížeče *Opera*
- prev.png ... obrázek předchozího měření
- proxy.png ... ikona útoku s využitím prohlížeče jako prostředníka
- return.png ... ikona zpětného tlačítka
- skew.png ... obrázek počítání odchylky
- speed2.png ... ikona výkonu zařízení
- usb2.png ... ikona USB
- vr.png ... ikona virtuální reality
- *