Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Digitální otisk webového prohlížeče

Viktor Goldscheider

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Viktor Goldscheider

Informatika

Název práce

Digitální otisk webového prohlížeče

Název anglicky

Digital fingerprint of a web browser

Cíle práce

Bakalářská práce se zabývá zkoumáním metod a principů využívaných pro rozpoznávání jednotlivých zařízení při prohlížení webových stránek (tzv. fingerprinting). Hlavním cílem práce je navržení programového řešení pro identifikaci zařízení.

Dílčí cíle:
- charakterizovat problematiku digitálního otisku zařízení,
- navrhnout řešení schopné takový otisk vytvořit,
- otestovat řešení v praxi a formulovat závěry.

Metodika

Teoretická část bude spočívat v analýze dané problematiky. Informace budou získávány z odborných publikací a dále zpracovávány pro účel vypracování této práce.

Před zahájením práce na praktické části budou zanalyzovány získané teoretické poznatky a již existující programová řešení. Na základě výsledků této analýzy vznikne návrh programového řešení.

V praktické části dojde k samotné implementaci programového řešení pro vytváření digitálních otisků webových prohlížečů. Řešení bude následně testováno v praxi. Na základě dílčích poznatků bude formulován závěr.

Doporučený rozsah práce

30–40 stran

Klíčová slova

digitální otisk, fingerprint, user agent, webový prohlížeč, zabezpečení webu, identifikace

---

Doporučené zdroje informací

Bernardo, V. and Domingos, D. (2016). Web-based Fingerprinting Techniques. In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications – SECRYPT, (ICETE 2016) ISBN 978-989-758-196-0; ISSN 2184-2825, pages 271-282. https://doi.org/10.5220/0005965602710282

Boda, K., Földes, Á.M., Gulyás, G.G., Imre, S. (2012). User Tracking on the Web via Cross-Browser Fingerprinting. In: Laud, P. (eds) Information Security Technology for Applications. NordSec 2011. Lecture Notes in Computer Science, vol 7161. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29615-4_4

Eckersley, P. (2010). How Unique Is Your Web Browser?. In: Atallah, M.J., Hopper, N.J. (eds) Privacy Enhancing Technologies. PETS 2010. Lecture Notes in Computer Science, vol 6205. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14527-8_1

Nikiforakis, Nick & Kapravelos, Alexandros & Joosen, Wouter & Kruegel, Christopher & Piessens, Frank & Vigna, Giovanni. (2013). Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. Proceedings – IEEE Symposium on Security and Privacy. 541-555. https://doi.org/10.1109/SP.2013.43

Tompoariniaina Nampoina Andriamilanto. Leveraging browser fingerprinting for web authentication. Systems and Control [cs.SY]. Université Rennes 1, 2020. English. ⍰NNT: 2020REN1S045⍰. ⍰tel-03150590⍰

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 15. 03. 2023

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Digitální otisk webového prohlížeče" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2023 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Michalovi Stočesovi, Ph.D. za vstřícný přístup a poskytnutá doporučení v průběhu zpracování této práce.

Digitální otisk webového prohlížeče

Abstrakt

Cílem této práce je na základě charakteristik webových prohlížečů a zařízení uživatelů navrhnout řešení, které by bylo schopno vytvořit otisk těchto prohlížečů a zařízení, který pak může být použit pro následnou identifikaci těchto uživatelů. Návrh řešení vychází z teoretických poznatků, které byly autorem práce získány z dostupných zdrojů. Teoretická část práce se zabývá problematikou otisku webových prohlížečů, příčinami jejich vzniku, klíčovými charakteristikami těchto otisků a dále metodami, které umožňují jejich získání. Teoretické poznatky byly aplikovány při zpracování praktické části, ve které bylo vyvinuto programové řešení pro sběr otisků webových prohlížečů. Programové řešení bylo nasazeno do praxe a s jeho pomocí byla získána data o charakteristikách prohlížečů a zařízení uživatelů. Tato data byla dále zpracována za účelem zpětného vyhodnocení výkonnosti navrhnutého programového řešení.

Klíčová slova: digitální otisk, fingerprint, user agent, webový prohlížeč, zabezpečení webu, identifikace, sběr dat, vývoj software

Digitální otisk webového prohlížeče

Abstract

The aim of this work is to propose a solution that would be able to create a fingerprint based on the characteristics of users' web browser and their device, which can then be used for their subsequent identification. The proposed solution is based on theoretical knowledge obtained by the author of the work from available sources. The theoretical part of the work deals with the issue of fingerprints of web browsers, the causes of their creation, key characteristics of these fingerprints and methods that enable them to be obtained. Theoretical knowledge was applied during the processing of the practical part, in which a software solution for collecting fingerprints of web browsers was developed. The software solution was put into practice and data on the characteristics of users' browsers and devices was obtained with its help. This data was further processed in order to retrospectively evaluate the performance of the proposed software solution.

Keywords: digital fingerprint, fingerprint, user agent, web browser, web security, identification, data collection, software development

Obsah

1 Úvod

Problém identifikace osob trápí lidstvo už od počátku vzniku moderních civilizací. Při zamýšlení každý najde situace v každodenním životě, u kterých je nanejvýš praktické, aby došlo k identifikaci zájmových stran. Pod těmito situacemi není nutné hledat nic složitého, stačí začít u konverzace mezi dvěma osobami, které se při zahájení konverzace obvykle představí, a tím dojde ke výměně informací z jejich profilů identit.

V historii byly tyto profily identit poměrně neobsáhlé. I z dnešního pohledu běžné údaje jako jsou např. křestní jména nebo příjmení nebyly v historii samozřejmostí. Vzhledem k jednoduššímu pojetí života v té době byly takové údaje v mnoha případech nepotřebné. Situaci lze předvést na příkladě vesnic. Ve tehdejších vesnicích žilo jen malé množství osob, ve kterém se dalo jednoduše předejít duplicitním údajům, případně šlo jedince upřesnit podle jeho specifických charakteristik např. pomocí bydliště ve vesnici.

Postupem času se nároky na identifikaci zvyšovaly. Počet žijících obyvatel se rapidně zvyšoval a svět se díky novým technologiím stával decentralizovanější.

Stejným způsobem lze nahlédnout na problematiku identifikace v prostředí internetu. S postupným přibýváním služeb a rostoucí komercializací internetu se zvyšovaly nároky na identifikací uživatelů. Postupy, které pokročilejší identifikaci umožňují jsou popsány v této práci.

2 Cíl práce a metodika

Cíl práce

Tato práce se zabývá zkoumáním metod a principů využívaných pro rozpoznávání jednotlivých zařízení při prohlížení webových stránek (tzv. fingerprinting). Hlavním cílem práce je navržení programového řešení pro identifikaci zařízení.

Dílčí cíle:

- charakterizovat problematiku digitálního otisku zařízení,
- navrhnout řešení schopné takový otisk vytvořit,
- otestovat řešení v praxi a formulovat závěry.

Metodika

Teoretická část bude spočívat v analýze dané problematiky. Informace budou získávány z odborných publikací a dále zpracovávány pro účel vypracování této práce.

Před zahájením práce na praktické části budou zanalyzovány získané teoretické poznatky a již existující programová řešení. Na základě výsledků této analýzy vznikne návrh programového řešení.

V praktické části dojde k samotné implementaci programového řešení pro vytváření digitálních otisků webových prohlížečů. Řešení bude následně testováno v praxi. Na základě dílčích poznatků bude formulován závěr.

3 Teoretická východiska

3.1 Co je to otisk

Termín otisk je mezi lidmi všeobecně známý. Většina z nich se s otisky setkává prostřednictvím literatury, filmů, seriálů a jiných typů zábavy, které jsou nejčastěji k lidem distribuované prostřednictvím nějakého datového nosiče.

Otisk je obecně chápán jako unikátní vlastnost objektu, která námi zkoumaný objekt jednoznačně odlišuje od ostatních objektů. V kriminalistice se otisky; nejčastěji prstů, méně často pak dlaní nebo nohou; používají k identifikaci pachatelů trestných činů. Věda, která zkoumá, objevuje a navrhuje postupy pro identifikaci osob na základě otisků, se nazývá daktyloskopie. [1]

Další relevantní vědou, která zkoumá a vyhodnocuje všechny charakteristiky živých organismů, které využívá pro jejich jednoznačnou identifikaci, je biometrie, která se na rozdíl od daktyloskopie nezaměřuje jen na otisky. Biometrie využívá nezaměnitelné znaky jako jsou např. otisk prstu, charakteristiky oční duhovky, DNA apod. [2]

3.2 Otisk v oblasti informačních technologií

Výše popsané principy identifikace živých organismů lze aplikovat na touto prací zkoumanou problematiku a přenést je do oblasti informačních technologií, konktrétně do oblastí internetu a internetových stránek.

Uživatelé internetu přistupují k internetovým stránkám prostřednictvím prohlížečů, které jsou vyvíjeny různými organizacemi. Tyto prohlížeče jsou spouštěny na operačních systémech, které jsou rovněž nesourodé a jsou vyvíjeny v různých podmínkách. Operační systémy jsou provozovány na hardwarech s různorodými funkcemi a dalšími fyzickými vlastnostmi.

Vzhledem k množství výrobců a vývojářů jednotlivých produktů a služeb, které uživatelé využívají, není možné zajistit stejné chování napříč produkty [3]. Z teoretického hlediska by zajištění shody bylo možné v případě, že by se jednotlivé firmy dohodly na výrobě identických produktů a služeb. Tato podmínka ovšem odporuje základním principům podnikání, dle kterých chce každá organizace prodávající nějaké produkty mít lepší produkty

než ostatní firmy, což přispívá k vyšším prodejům a poražení konkurence. Tento fakt organizace nutí k vývoji produktů s novými funkcemi, vlastnostmi a jinými charakteristikami, které jejich produkty odlišují od ostatních.

Kombinace individuálních charakteristik uživatelem použitého hardwaru, operačního systému, programů nainstalovaných v systému a v neposlední řadě samotného webového prohlížeče vyústí v souhrn těchto charakteristik, který pak definuje prostředí jednotlivých uživatelů (tj. uživatelské prostředí).

Otisk prohlížeče lze označit za jednoznačný identifikátor, který odlišuje konkrétní prohlížeč od ostatních prohlížečů ve zkoumaném celku. Cílem stran používajících tyto principy je v ideálním případě opakovaná a v čase neměnná identifikace jednotlivých uživatelů.

Mezi autory panuje shoda, že identifikátor, který umožňuje dosažení takového cíle, vzniká právě díky předešle popsaným vlastnostem uživatelského prostředí.

Autoři, kteří se zabývali touto problematikou, definují otisk v kontextu webového prohlížeče následovně:

„Koncept vytváření otisku zařízení je založen na předpokladu, že každé elektronické zařízení má množinu fyzických a/nebo logických vlastností, které lze zachytit a použít k odlišení od zbytku celku.“ [4]

nebo

„Vytváření otisku zařízení je založeno na předpokladu, že žádná dvě zařízení nejsou stejná, a že lze tuto vlastnost použít k profilování těchto zařízení pomocí produkovaných vzorců chování, a to za předpokladu, že se tyto vzorce v průběhu času opakují.“ [4]

nebo

„Vytváření otisku prohlížečů je metoda sledování webových prohlížečů, která na rozdíl od tradičních sledovacích metod, jako jsou IP adresy a soubory cookies, využívá informace o nastavení prohlížeče, které prohlížeč poskytuje webovým stránkám.“ [5]

Termín „otisk“ byl zvolen kvůli shodě využití s otisky prstů, kdy oba principy poskytují metody k jednoznačné identifikaci zkoumaných objektů. Kromě v této práci popisovaných

otisků prohlížečů se otisky používají i v dalších odvětvích digitálních technologií např. při forenzní analýze fotografií, při které se zjišťuje, jakým zařízením byly fotografie vytvořeny, případně kde byly uloženy. Důležitým zdrojem informací jsou metadata souboru, které lze chápat jako unikátní kombinaci charakteristik zkoumané fotografie. [6]

Literatura často zaměňuje termíny *vytváření otisku zařízení* a *vytváření otisku prohlížeče*. Někteří autoři tyto dva pojmy nerozlišují a používají je synonymně. Někteří autoři [7] [8] považují za součást otisku prohlížeče jen charakteristiky, které pocházejí ze samotného prohlížeče, další autoři [5] [9] i charakteristiky z prostředí počítače uživatele (operační systém, systémová nastavení, uživatelská data a hardware). Tyto charakteristiky lze získat s použitím přídavných modulů, které umožňují rozšířit pravomoci prohlížeče na úroveň, kterou by bylo možné dosáhnout při maximálním využití pravomocí přihlášeného uživatelského účtu [9]. Autoři se neshodují na podmínce konzistence otisku v čase. Někteří zaujímají postoj, že otisk je prakticky použitelný jen v případě, že v průběhu času nedochází k jeho změnám. [5]

Zásadní vlastností metody identifikace pomocí vytváření otisku webového prohlížeče je její obtížná detekce a následná blokace. Na rozdíl od souborů cookies je povědomost o existenci této technologie relativně nízká. Dalším faktem také je, že celý proces probíhá v programové úrovni a neinformovaný uživatel bude jen obtížně sledovat průběh tohoto procesu. U cookies je možné proces sledovat pomocí např. rozšíření prohlížeče nebo developerské konzole. Prohlížeče nenabízejí způsoby, pomocí kterých by bylo možné zachytit pokusy o přístup k citlivým charakteristikám. Uživatel, který si uvědomuje existenci sledovacích prostředků, aktivně pečuje o své soukromí na internetu a praktikuje best-practices jako je blokování cookies třetích stran, je proti většině metodám požívaných při procesu vytváření otisku naprosto bezbranný. [5] [10]

Protiintuitivně, pokud si takový uživatel nainstaluje nějaký z dostupných blokátorů scriptů a reklam (např. AdBlock nebo uBlock), využije specializovaného prohlížeče na ochranu soukromí (např. Brave), zapne požadavek Do Not Track nebo sám modifikuje vlastnosti prohlížeče, vytváří těmito kroky další unikátní charakteristiky, které zvyšují přesnost sledování. Každá unikátní nebo neobvyklá vlastnost je pro vytváření otisku neocenitelným atributem a snižuje pravděpodobnost výskytu prohlížeče se stejným otiskem. [9] [11]

Samotný sběr charakteristik a následné vytváření otisku se zařazuje do šedé zóny a záleží na jednotlivých využitích autory programů, zda jsou jejich záměry legitimní nebo neetické, případně nelegální. Tvorba otisku probíhá zcela pasivně a v případě legitimních úmyslů je výhodou, že uživatele svou existencí neobtěžuje. V případě neetických nebo nelegálních úmyslů jsou uživatelé v podstatě bezbranní, naopak průběh takových praktik není uživateli zviditelněn a uživatel tak může žít ve falešném pocitu bezpečí. [11] [4]

3.3 Vývoj vzniku

3.3.1 Cookies

3.3.1.1 Vznik souborů cookies

HTTP je zkratka pro Hypertext Transfer Protocol. HTTP umožnuje a řídí komunikaci mezi webovými servery a webovými klienty jako jsou webové prohlížeče. Je nepostradatelnou technologií pro provoz webových služeb a v praxi se využívá při vývoji webových stránek, služeb a mnoha dalších aplikací. [12]

Při implementaci protokolu HTTP vývojáři řešili problematiku uchovávání stavových proměnných. Posílání informací přes HTTP protokol je bezstavové, což je považováno za výhodu při budování fyzických sítí a vývoji aplikační a komunikační logiky. Bezstavovost je vlastnost, která sděluje, že popisovaný systém neuchovává žádné informace o předešle provedených transakcích. [13]

Většina webových aplikací ke svému fungování vyžaduje udržení stavu relace. Do této množiny patří jakákoliv webová aplikace, která mezi jednotlivými požadavky vyžaduje znalost faktu, zda se jedná o stejného uživatele jako v předešlých požadavcích. Příkladem z praxe může být např. jakákoliv webová stránka s možností přihlášení nebo online obchod. Bez uchování stavu není možné udržet informace o přihlášení uživatele nebo o přidání položek do košíku v uvedeném příkladu online obchodu. Takové stránky by bez uchování stavů nemohly fungovat a byly by omezeny jen na doručování statického obsahu bez možnosti hlubší interakce s uživatelem. [14]

Problém stavovosti vyřešil vznik souborů cookies, které byly definovány v RFC (Request for Comments) standardech [15] a postupem času byly měněny a rozšiřovány. Jsou jednou z páteřních technologií, které umožňují fungování webových stránek. [16]

Cookies jsou bloky dat, které se při každém požadavku vyměňují mezi webovým serverem a uživatelem. Posílají se jako součást HTTP požadavku, konkrétně jeho hlavičky. Tvorba a správa souborů cookies je obvykle v režii serveru. Po zaslání požadavku klientem server v hlavičce odpovědi uvede jednotlivé cookies, které chce, aby si uživatelův prohlížeč nastavil. Prohlížeč si po přijetí odpovědi cookies nastaví. Při dalším požadavku klient odesílá předešle nastavené cookies. Alternativně je možné cookies vytvořit a nastavit i bez interakce se vzdáleným serverem. Cookies se pak generují přímo v prohlížeči, příkladem může být vygenerování alfanumerického uživatelského identifikátoru, který se pak používá při další komunikaci. [16]

Jednotlivé cookies mají své názvy a hodnoty. Server může při nastavování specifikovat expiraci nebo konkrétní doménu a URL adresu, pro které má být daná cookie nastavena. Většina prohlížečů omezuje délku názvu a hodnot cookies a stanovuje maximální možný počet uložených souborů pro každou doménu. V praxi jsou hodnotami souborů cookies nejčastěji kratší alfanumerické řetězce. [16]

3.3.1.2 Cookies jsou považovány za hrozbu

Cookies, jak již bylo zmíněno, jsou nepostradatelnou technologií dnešního internetu, ale i přes tento fakt jsou cookies mezi laickými uživateli považovány za něco nepatřičného a neetického. Příčinou tohoto stavu je boj mezi velkými technologickými společnostmi a uživateli jejich produktů, který vyústil v medializaci a legislativní změny. [17] [18]

Největším obchodním odvětvím na internetu je prodej reklamy koncovým uživatelům [19]. Většina sociálních sítí (např. Facebook nebo Twitter) a internetových vyhledavačů (např. Google nebo Bing) jsou bezplatné – tedy alespoň na první pohled. Oblíbené webové služby mají možnost doručovat libovolný obsah velkému množství uživatelů. Samotná přítomnost uživatelů a jejich data lze označit za zpeněžitelné komodity. Ty následně nakupují osoby, které se snaží o zviditelnění svých produktů. Společně vytvářejí trh s reklamou a daty, který se odhaduje na vysoké stovky miliard dolarů ročně. [20] [21]

Základem úspěšné reklamní kampaně je doručení reklamního obsahu cílové skupině. Důležitým úkolem reklamních společností je zařadit uživatele s vysokou přesností do cílových skupin. Digitální prostor umožňuje uživatelům pohyb s velkou mírou anonymity. Tato vlastnost internetu se těší oblibě uživatelů a každý pokus o její omezení vede k odporu

z jejich strany. Poskytovatelé služeb v praxi narážejí na nezájem uživatelů o vyplňování osobních údajů. Tento nezájem roste s mírou intimnosti požadovaných údajů. Přemrštěné shromažďování údajů je jedním z parametrů, který může uživatele odradit ještě před použitím samotné služby. [22] [23]

Reklamní společnosti pro každého uživatele vytvářejí jeho behaviorální profil, který není tvořen fyzickou identitou uživatele (jako jméno nebo vzhled), ale především jeho chováním na internetu. Služby uživatele sledují v průběhu času, ukládají data o prováděných interakcích se službou, a tyto údaje pak algoritmicky zpracovávají. Na základě historie vyhledávání, navštívených odkazů, času strávených na webových stránkách a dalších metrik jsou reklamní společnosti schopny vytvořit vzorce chování sledovaného uživatele, zařadit jej do cílových skupin a následně mu doručovat personalizované reklamní nabídky. [24]

Proces tvorby behaviorálního profilu uživatele z počátku využíval především soubory cookies, které umožňovaly sledování ze strany inzerentů. Po odhalení těchto praktik odbornou veřejností se cookies staly problematickými v očích uživatelů, kteří se obávali jejich zneužití, jelikož jsou technickým řešení, pomocí kterého sledování probíhá. [9]

3.3.1.3 Cílené mazání cookies

Účastníci sporu diskutovali, zda je použití cookies pro reklamní účely legitimní a zda je v pořádku, že jsou informace shromažďovány i bez výslovného souhlasu uživatele. V některých zemích se postupem času objevily regulace, které zákonem omezují využití cookies a v některých případech zavádí nutnost obdržení souhlasu uživatele pro jejich použití. [18]

Po medializaci problematiky souborů cookies se uživatelé začali bránit průběžným promazáváním souborů cookies, instalací různých rozšíření webových prohlížečů, jako jsou AdBlock nebo uBlock, která blokují požadavky na servery služeb třetích stran, a použitím inkognito režimu webových prohlížečů, který mezi jednotlivými relacemi neukládá data. [25] [11]

3.3.2 Vývoj nových prostředků pro sledování uživatelů

Míra efektivity souborů cookies se kvůli těmto opatřením snížila a reklamní společnosti byly donuceny k vývoji alternativních způsobů sledování uživatelů. Jedním z takových způsobů jsou tzv. supercookies (také známe jako zombie cookies a evercookies). Stejně jako u souborů cookies se jedná o soubory, které nesou informace o stavových proměnných webových stránek. Zásadním rozdílem je způsob, jakým dochází k ukládání těchto souborů. Na rozdíl od cookies, u kterých jsou postupy pro přenos a ukládaní standardizované (a tím pádem předvídatelné a blokovatelné), má každá supercookie svůj vlastní systém přenosu a ukládání v prohlížeči uživatele. [7] [17]

Hlavní předností supercookies je schopnost odolávat pokusům uživatele o smazání, tím, že se jejich informace ukládají do několika různých lokací. Při smazání v jedné lokaci se supercookie načte z jiné a dojde ke znovunastavení ve všech lokacích. V praxi se pro ukládání používají technologie jako např. HTML5 Session Storage, HTML5 Local Storage nebo HTML5 Databáze (Web SQL). [7] [17]

Ukládaní supercookies provádí program, který je při navštívení stránky stažen a spuštěn. Předávání supercookies mezi klientem a serverem probíhá pomocí jejich začlenění do HTTP požadavku, konkrétně do hlavičky jako HTTP header nebo jako součást těla (body). Při opakovaném navštívení stránky se proces opakuje s tím, že program může přistupovat k již nastaveným hodnotám a znovu je použít. [26]

Supercookies poskytují jenom částečné řešení, jelikož i u nich může dojít ke kompletnímu smazání. Vývoj způsobů sledování pokračoval a výsledkem byly inovativní metody, které lze zařadit do kategorie metod vytváření otisků webových prohlížečů. První zmínky o takových metodách v odborných publikacích lze pozorovat kolem roku 2009, ale je možné, že k využití takových technologií docházelo již dříve. [7]

3.4 Proces vytváření otisku

Proces začíná ve chvíli přijetí požadavku uživatele serverem. Dochází k zisku informací ve dvou oblastech: HTTP hlavička a internetové připojení mezi uživatelem a serverem. [7]

Hlavička na server dochází jako součást HTTP požadavku prohlížeče uživatele. Hlavička obsahuje pole a příslušné hodnoty těchto polí. Lze sledovat vlastnosti jako pořadí, existenci jednotlivých hlaviček a porovnávat jejich hodnoty. Prohlížeče v jednotlivých hlavičkách odesílají různě hodnoty. [10] [11]

Accept HTTP hlavička: Informuje webový server o možnostech webového prohlížeče zpracovávat data. Specifikuje typy dat, které prohlížeč umí zpracovat. [4]

Accept-Language HTTP hlavička: Informuje Webový server o jazyku systému uživatele. Server data využívá pro vybrání vhodné jazykové varianty stránky. [4]

User-Agent HTTP hlavička: Informuje webový server o specifikacích prohlížeče, jeho verzi a operačním systému počítače. Jedná se o jeden z nejznámějších identifikátorů prohlížečů. [4]

Pořadí HTTP hlaviček: Pořadí hlaviček je podle specifikace HTTP protokolu irelevantní. Vzhledem k této vlastnosti nebylo nutné vytvářet standard, který by určoval pořadí hlaviček. Každý prohlížeč si pořadí hlaviček zajišťuje svým způsobem. Pořadí je zachované i po přijetí požadavku serverem a může být analyzováno.

Do-not-track HTTP hlavička: Jedná se o nepovinnou HTTP hlavičku. Vznikla jako reakce na odpor uživatelů proti sledování na internetu. Po vyslovení zájmu o nesledování uživatelem se tato hlavička připojí ke ostatním hlavičkám a posílá se v požadavku na webový server. Server by měl po přijetí této hlavičky přestat uživatele sledovat. Respektování hlavičky není nikým kontrolované a v praxi ji většina webových stránek ignoruje. Uživatelé jsou i přes toto nastavení dále sledování. Přítomnost vysloveného zájmu o nesledování lze detekovat pomocí JavaScriptu po přístupu k hodnotě atributu `*navigator.doNottrack*`. [4]

Webový server následně vrací obsah stránky, na které je umístěn program pro vytváření otisku.

Program se po stažení v prohlížeči uživatele spouští. Při průběhu programu se sbírají informace z vybraných částí prostředí webového prohlížeče uživatele a po dokončení programu se výsledky odesílají na webový server pro další zpracování.

V praxi se nejčastěji setkáme s těmito způsoby odesílání výsledků: na sub-/doménu navštívené stránky a na doménu třetí strany (nejčastěji specializovaná služba poskytující software pro vytváření otisku).

Na sub-/doménu *https://[tracking.]navstivena-stranka.cz/posli/otisk*
Na doménu třetí strany *https://jsem-tracking-sluzba.cz/posli/otisk*

Problém prvního způsobu je jeho obtížná blokace. Vlastníkům stránky tento způsob umožňuje efektivně skrývat programovou část tvorby otisku a znemožnit efektivní blokaci. [9]

Serverová část programu přijatá data zpracuje a generuje identifikátor pro daná data. Tento identifikátor se pak používá podle potřeb vlastníka stránky.

3.5 Způsoby využití

Otisk se využívá pro tyto účely: identifikace konkrétního uživatele a identifikace použitého prohlížeče.

Sledování na webu: Jedná se o sběr blíže neurčených dat. Může se jednat o analytická data, která podávají informace o návycích a aktivitě uživatele. Pro společnosti je výhodné tato data sledovat dlouhodobě. Tato data mohou být použita libovolným způsobem. V praxi se jedná např. o použití prodejci a reklamními společnostmi pro vyváření personalizované reklamy. [27]

Využití pro autentifikaci: Autentifikace má za úkol ověřit pravost uživatelem udané identity. V praxi se lze setkat s identifikačními faktory jako jsou hesla, jednorázové kódy, elektronické podpisy apod. Využití otisku webových prohlížečů spočívá v přidání dalšího faktoru identifikace k dalším jako jsou již zmíněné. Cílem je zvýšit účinnost ostatních faktorů a zabránit zneužití dat uživatele. [28]

Ochrana proti zneužití citlivých dat: Využívá otisk webového prohlížeče většinou jako jeden z faktorů utvářející risk skóre daného uživatele. Toto skóre se nejčastěji využívá v oblasti internetového nakupování a plateb. V případě, že skóre klesne pod kritickou úroveň, může obchodník platbu od takového uživatele preventivně odmítnout, jelikož by se jinak vystavil statisticky vyššímu nebezpečí. [29]

Ochrana proti automatizaci: Zaměřuje se na detekci a potlačení automatizovaných programů. Tyto programy se označují termínem bot (zkratka od robot). Většina takových programů je vytvořena s pomocí nástrojů, které umožňují ovládaní přes prostředí programovacího jazyka. Tyto nástroje umožňují připojení k prohlížeči a implementují funkce jako prohlížení stránek, spouštění JavaScript programů nebo uživatelské vstupy (vkládání textu nebo klikání myší). Úpravami atributů prostředí prohlížeče boti vytvářejí prostor pro detekci, kterého lze využít při vytváření otisku a následně botovi znemožnit přístup ke stránce. [30]

Detekce anomálií: Může upozornit na zařízení s neobvyklým chováním. Příkladem může být desktopový prohlížeč, který přistupuje na mobilní verzi stránek. [29]

Personalizace stránek: Stránky mohou mít zájem dodávat různým prohlížečům odlišné verze stránky např. kvůli technickým limitacím zařízení uživatele. Dále mohou stránky využít otisk pro stanovení jazykové verze stránky, a to bez znalosti IP adresy uživatele. [31]

Komunikace s uživatelem: Stránky mohou mít zájem uživatele informovat o určitých skutečnostech. Jedná se např. o použití zastaralé verze prohlížeče nebo operačního systému nebo o absenci zásadních funkcí, bez kterých stránka nemůže správně fungovat.

Forenzní analýza: Využívá otisk webových prohlížečů jako jednu z metod při sledování útočníků. Otisk může být použit k identifikaci pachatelů kybernetických útoků.

Využití při kybernetických útocích: Otisky prohlížečů usnadňují provádění cílených útoků na zranitelné skupiny uživatelů. Díky získaným informacím je útočník schopen rychleji a efektivněji provádět útoky na technologie, které konkrétní uživatel využívá. Mezi cenné informace patří např. povědomí o typu operačního systému a jeho verze. [8]

3.6 Atributy a metody

Při vývoji programů, jejichž úkolem je vytvoření otisku prohlížeče, se klade důraz na výběr charakteristik a testů, které budou používány k reprezentaci testovaného prohlížeče. Pro zjednodušení se používají termíny ***atribut*** a ***metoda***.

Atributy označují konkrétní vlastnost, která může nabývat různých hodnot (např analogicky u lidí: atribut – barva vlasů, hodnoty – [černá, blond atd.]).

Metody označují procesy, které jsou prováděny za účelem získání atributů a jejich hodnot. Některé atributy v prostředí prohlížeče existují i bez předchozího vytvoření, jiné se vytvářejí za běhu programu pomocí metod.

Dalším důležitým termínem je ***fingerprinter***, který označuje vývojáře a/nebo provozovatele a/nebo vlastníka programu, který otisky vytváří.

Výběr sbíraných atributů a vykonávaných metod přímo ovlivňuje výkonnost celého programu. ***Výkonnost*** je kvalitativní vlastnost takového programu, která pojednává o míře schopnosti programu rozlišit jednotlivé uživatele a jejich prohlížeče, a o konzistenci výsledků v čase.

Vývojáři programů výkonnost zlepšují pomocí optimalizací jako je výběr atributů s vysokým počtem distinktivních hodnot nebo vývojem nových a inovativních metod. Pro výkonnost jednotlivých metod se používá označení ***entropie***, která udává míru nahodilosti výsledků. Ideální je situace, ve které metody vrací zcela unikátní výsledky, které umožňují jednoznačnou identifikaci jednoho prohlížeče mezi ostatními. [7]

3.6.1 Atributy

Jednotlivé atributy lze rozdělit do skupin na základě různých kritérií. Jedno z možných členění nabízí rozdělení atributů na základě způsobu vytváření a přístupu k jejich hodnotám. Toto členění definuje dva základní typy atributů: ***fixní*** a ***dynamické***. Pro fixní atributy dále stanovuje dvě podkategorie: pasivní a aktivní. [28]

3.6.1.1 Fixní atributy

Fixní atributy jsou tvořeny souhrnem vlastností uživatelského prostředí, nikoliv jeho chováním. Uživatelská prostředí jsou tvořena kombinací technologií, které konkrétní uživatel používá. Mezi tyto technologie řadíme samotný prohlížeč; nainstalované zásuvné moduly, rozšíření a další software; operační systém; interní a externí hardware; a internetové připojení. Dohromady definují jedinečný soubor atributů, k jejichž hodnotám lze přistupovat. [28]

Za **pasivní fixní atributy** označujeme takové, k jejichž zisku dochází i bez explicitního vyžádání. To jsou takové, které jsou na servery fingerprintera odesílány jako součást běžné komunikace, a tedy nedochází ke spouštění programu v prohlížeči uživatele. Obvykle se jedná o atributy protokolů využívaných při komunikaci uživatele se serverem, nejčastěji se jedná o protokol HTTP. Během komunikace dochází k výměně informací prostřednictvím technologií jako je IP adresa, HTTP hlavička, TCP (Transmission Control Protocol) nebo TLS (Transport Layer Security). [28]

Aktivní fixní atributy jsou na druhé straně takové, jejichž hodnoty si fingerprinter záměrně vyžaduje. Sběr hodnot takových atributů probíhá prostřednictvím programu, který se spouští po navštívení stránky uživatelem. Tyto programy jsou většinou napsány v jazyce JavaScript, který je nativně podporován nejpoužívanějšími prohlížeči. Dále může docházet k použití jiných programovacích jazyků jako WebAssembly nebo ke spuštění kódu prostřednictvím zásuvných modulů (např. Flash, Java nebo Silverlight), které jsou nainstalované na počítači uživatele. [28]

3.6.1.2 Dynamické atributy

Dynamické atributy jsou tvořeny jak souhrnem vlastností uživatelského prostředí, tak zároveň specifickým chováním takového prostředí. Jejich tvorba je vyvolána aktivací služeb prohlížeče, na jejímž počátku dojde k předání vstupních dat, která jsou fingerprinterem vytvořena takovým způsobem, aby v prohlížeči vyvolala specifické chování. Mezi fingerprintery využívané služby řadíme především takové, které pracují s audiovizuálními daty. Výsledky těchto služeb jsou ovlivňovány proměnnými na všech úrovních uživatelského prostředí (software, hardware atd.). Příkladem můžou být prostředí s odlišnými grafickými kartami, které i přes naprostou shodnost jejich zbytku vykazují

naprosto odlišné chování a výsledky. Dynamické atributy jsou v průměru považovány za atributy s nejvyšší mírou entropie. [28]

3.6.1.3 Další dělení atributů

Jiný způsob dělení atributů definuje systém, který atributy rozčleňuje pomocí místa jejich původu. Mezi autory nepanuje shoda na jednotlivých kategoriích atributů, nicméně je lze vymezit přibližně následovně: atributy prohlížeče a jeho chování; atributy operačního systému; atributy hardwaru; a atributy sítové komunikace. [8] [4]

3.6.2 Metody

3.6.2.1 JavaScript a jeho objekty

JavaScript je nejrozšířenějším skriptovacím jazykem pro vytváření interaktivních webových aplikací a přidávání dynamických funkcí na webové stránky. Je podporován všemi hlavními webovými prohlížeči, včetně Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge a dalších. Zmíněné prohlížeče obsahují vestavěný JavaScript engine, který jim umožňuje interpretovat a spouštět kód programů napsaných v JavaScriptu bez jakýchkoliv přídavných modulů. [32] [33]

Document Object Model (DOM) je programovací rozhraní pro HTML a XML dokumenty. CSS Object Model (CSSOM) je programovací rozhraní pro manipulaci s CSS. Tyto rozhraní poskytují přístup k objektům ***Navigator*** a ***Screen***, které byly přidávány do webových prohlížečů během jejich největšího růstu za účelem poskytování informací o prohlížeči a o zařízení uživatele vývojářům webových stránek. Takové informace vývojářům umožňují tvorbu dynamických a interaktivních aplikací. Implementace těchto objektů nalezneme ve všech moderních webových prohlížečích. K objektům a jejich atributům se přistupuje pomocí jazyka JavaScript. [34] [35]

Objekt *Navigator* obsahuje podrobné informace o prohlížeči; prostředí, ve kterém se prohlížeč nachází; funkcích, které prohlížeč podporuje, a dalších různých charakteristikách. Mezi jednotlivými typy prohlížečů se mění jak jednotlivé hodnoty atributů, tak i samotné atributy. **Objekt *Screen*** poskytuje informace o obrazovce zařízení uživatele, ty jsou ovlivňovány nastavením v operačním systému a parametry monitorů nebo obrazovek připojených k zařízení uživatele. [36] [37]

Tabulka 1 Atributy objektu Navigator v prohlížeči Google Chrome a jejich popis

appVersion	Informace o verzi prohlížeče.
cookieEnabled	Stanovuje, zda je v prohlížeči povolené ukládání souborů cookies.
deviceMemory	Množství operační paměti v systému. Hodnoty jsou kvůli soukromí uživatelů zaokrouhlovány, dále dochází k odstranění extrémů mimo rozmezí 0.25-8 GB.
hardwareConcurrency	Množství logických jader procesoru, které jsou prohlížeči k dispozici.
language	Hlavní uživatelem preferovaný jazyk. Ten se obvykle shoduje s jazykem uživatelského rozhraní prohlížeče.
languages	Všechny uživatelem preferované jazyky.
maxTouchPoints	Maximální počet souběžných bodů dotyku zařízení.
pdfViewerEnabled	Uvádí, zda je prohlížeč schopen zobrazit soubory PDF.
platform	Platforma systému, na které je prohlížeč spuštěn.
productSub	Číslo sestavení (build) prohlížeče.
userAgent	User-Agent prohlížeče včetně názvu prohlížeče, verze a operačního systému.
vendor	Jméno organizace, která prohlížeč vyvinula.

Tabulka 2 Atributy objektu Screen v prohlížeči Google chrome a jejich popis

availHeight	Dostupná výška v pixelech pro obsah webové stránky
availLeft	Vrací první dostupný pixel z leva relativně k umístění prohlížeče
availTop	Pozice prvního pixelu shora, který lze použít pro obsah webové stránky
availWidth	Dostupná šířka v pixelech pro obsah webové stránky
colorDepth	Barevná hloubka
height	Výška obrazovky
orientation	Orientace zařízení
width	Šířka obrazovky

Metoda 1.

Za základní metodu je považováno procházení jednotlivých atributů zmíněných objektů a jejich hodnot. Dostačuje k vytvoření minimálního otisku webového prohlížeče. K zamyšlení je fakt, že hodnoty atributů jsou volně přístupné, s tím že jejich sběr je naprosto triviální. Tyto atributy jsou často předmětem záměrné změny ze strany uživatelů. [9] [11] [4]

Metoda 2.

Metoda vychází z toho, že kromě atributů objektů lze zkoumat chování samotných objektů jakožto celků. Při enumeraci atributů objektů pomocí JavaScriptu bylo pozorováno jedinečné chování mezi jednotlivými rodinami, typy a verzemi prohlížečů. Prohlížeče a jejich verze bylo možné mezi ostatními rozpoznat pomocí specifických vlastností objektů *Navigator* a *Screen*. Kromě chybějících a přebývajících atributů se jednotlivé prohlížeče lišily i v pořadí, ve kterém byly jednotlivé atributy iterovány. [9]

Metoda 3.

Další metoda se zaměřuje na porovnání rozdílů mezi objekty *Navigator* a *Screen* a obyčejnými JavaScriptovými objekty. Objekty testuje pomocí systematického mazání, měnění a přidávání atributů a hodnot atributů těchto objektů. Po provedení operací zjišťuje, k jakým změnám došlo na testovaných objektech. Jednotlivé prohlížeče v tomto směru vykazují rozdílné chování, některé např. hodnoty smažou, jiné příkazy ignorují. [9]

Metoda 4.

Dále je možné nad zkoumanými objekty zavolat metodu `.*toString*`, která vrací textový řetězec reprezentující daný objekt. Nejčastěji se používá při převodu objektů na primitivní datové typy (např. ze seznamu na textový řetězec). Voláním této metody nad různými atributy v objektech *Navigator* a *Screen* bylo možné pozorovat proměnlivost výsledků mezi prohlížeči. Pro stejné atributy těchto objektů jsou v jednotlivých prohlížečích použity rozdílné datové typy (např. *[object PluginArray]* a *[object DOMPluginArray]*). Dále se lišilo formátování vracených hodnot. [9]

3.6.2.2 Textové fonty

Textové fonty a jejich chování

Skupina metod detekce nainstalovaných fontů využívá rozdílů mezi jednotlivými operačními systémy, jejich verzemi a jazykovou lokalizací systému. Webové prohlížeče při zobrazování webových stránek standardně používají fonty nainstalované v operačním systému. Vývojáři stránek můžou použít i vlastní nenainstalované fonty, které se stahují při načtení stránky, ale ty pro tuto metodu nejsou relevantní.

Metody během testování sledují chování prohlížeče, které prozrazuje, zda je testovaný font v systému nainstalovaný, či nikoliv. Při použití nenainstalovaného fontu prohlížeč tento font nahradí defaultní alternativou pro daný systém, což je stav, který je možné detekovat. Metody využívají vlastností a chování CSS a nástrojů k úpravě html dokumentů. [38] [10] [9]

Metoda 1.

Jedna z metod začíná přípravou seznamu fontů, jejichž přítomnost v systému uživatelů má program testovat. Dále je nutné provést výzkum chování prohlížečů. Zjišťuje se, které fonty se používají jako fall-back varianta v případě nepřítomnosti testovaného fontu. Nejčastěji se jedná o fonty z rodin fontů *sans* nebo *serif*.

Při testování dochází k vytvoření neviditelného *iframe* elementu, do kterého je vkládán text s předdefinovanou velikostí ve zvoleném fontu. Doporučuje se použít nevyšší možnou velikost textu, jelikož v takovém případě dochází ke zdůraznění specifických vlastností fontů, což následně usnadňuje porovnávání shodnosti. Velikost výsledného textu na obrazovce je změřena a hodnota v pixelech uložena pro další zpracování. Během testování se kromě existujících fontů použije i nesmyslný a tím pádem neexistující font, který stanoví rozměry fall-back fontu (jelikož dojde k zaručenému použití). Při zpracování výsledků se zjišťuje, které z testovaných fontů mají rozdílnou velikost textu s fall-back fontem. Takové fonty jsou na systému nainstalovány, jelikož nedošlo k využití defaultního fall-back fontu. Opakem jsou pak fonty, u kterých je velikost výsledného text totožná s fall-back fontem – takový font v systému chybí. [9]

Metoda 2.

Jako vstup jsou při testování používány textové řetězce, které jsou vytvořeny takovým způsobem, který při testování zdůrazní rozdíly mezi prostředími. Další z metod tohoto typu rozšiřuje alfanumerické textové řetězce o využití znaků emoji (také známy jako smajlíky). Vzhled emoji není standardizován a každý systém, případně i jeho distribuce, si implementaci a vizuální podobu stanovuje svým vlastním způsobem. Metoda ukazuje, že na základě tohoto principu lze rozlišit jednotlivé operační systémy, jejich verze a v některých případech i výrobce zařízení. To potvrzuje v praxi, kde od sebe rozlišila mobilní zařízení používající operační systém Android od rozdílných výrobců (např. Samsung, LG nebo HTC) nebo jednotlivých verzí systémů (např. Android 4.4 nebo Android 5.0). [10]

Další metody

Alternativní metody k získávání výsledků používají místo JavaScriptu zásuvné moduly jako Adobe Flash nebo Java. Hlavní předností využití zásuvných modulů je možnost získat kompletní seznam nainstalovaných fontů. Výše popsané metody jsou založené na JavaScriptu a pracují s předdefinovaným seznamem fontů. Tento způsob nese riziko, že některé z méně používaných fontů nemusí být v seznamu zahrnuty, a tak nedojde k jejich nalezení. Další výhodou zásuvných modulů je, že seznam je z rozhraní obvykle vracen v seřazené podobě. Pořadí položek seznamu je mezi prostředími proměnlivé, což přispívá ke zvýšení entropie atributu. [7] [9]

3.6.2.3 CSS

CSS jako alternativa k JavaScripu

Metody níže popsané využívají chování technologií, které jsou používány při vývoji webových stránek, konkrétně ty, které jsou definovány standardem CSS3. Záměrem autorů bylo vytvořit metodu nezávislou na použití JavaScriptu. Pokusy o vytváření otisků pomocí metod, které jsou závislé na funkcích poskytovaných jazykem JavaScript, lze ze strany uživatele zablokovat pomocí vypnutí JavaScriptu, které nicméně není pravděpodobné vzhledem k jeho popularitě při vývoji webových stránek, které by se po vypnutí staly nepoužitelné. [39]

Metoda 1.

Součástí každého webového prohlížeče je vykreslovací jádro, které zajišťuje převod kódu stránky do vizuální podoby. Implementace vykreslovacích jader mezi prohlížeči vykazují odlišné chování. Metoda stanovuje sérii testů, které jsou schopny takové rozdíly detekovat.

Testy pomocí příkazů s různými CSS selektory, atributy a jejich hodnotami zjišťují, zda je vykreslovací jádro prohlížeče schopné takové příkazy vykonat, či nikoliv. Příkaz kromě testovacího případu obsahuje další příkaz, který pro své vykonání stahuje data ze serveru. Takovými daty jsou nejčastěji obrázky nebo fonty vyvolané např. příkazem pro nastavení pozadí elementu. O úspěšném vykonání příkazu se fingerprinter dozvídá po přijetí požadavku uživatele na jeho serveru. V opačném případě není příkaz vykonán a data ze serverů nejsou vyžádána. Tyto testy jsou vhodné pro zisk informací jako je typ prohlížeče nebo jeho verze. [39] [40]

Metoda 2.

Pro sběr specifičtějších informací metoda využívá chování funkce Media queries. Media queries se využívají při vývoji webových stránek s responzivním designem. Na základě stavu zařízení uživatele lze nastavovat jednotlivé atributy CSS, mezi atributy stavu se řadí např. velikost obrazovky, velikost okna, orientace zařízení nebo poměr mezi hardwarovým a CSS rozlišením apod. Responsivní webová stránka s implementovanými Media queries se přizpůsobuje zařízení uživatele a např. při zmenšení okna prohlížeče nebo při rotaci zařízení překreslí stránku takovým způsobem, aby nedošlo k snížení pohodlí uživatele při prohlížení webové stránky. [39]

Testování pomocí *Media queries* probíhá totožným způsobem jako v případě testování s CSS selektory. Tvůrci metody prezentují fungování *Media queries* na příkladu, ve kterém po definování jednotlivých testů bylo možné zjistit rozlišení obrazovky uživatele. Stejně jako u příkladu s CSS selektory probíhá výměna těchto informací prostřednictvím dotazů na server fingerprintera. Další testy se zaměřovaly na specifické vlastnosti některých prohlížečů. Prohlížeč Firefox umožnoval přistup k informacím o dalších systémových proměnných jako např. nastavení barev uživatelského rozhraní. [39]

Metoda 3.

Další test byl pomocí stejných principů schopen odhalit přítomnost nainstalovaných fontů. Zjišťování probíhá pomocí funkce `@*font-face*`, která umožňuje importovat libovolné fonty,

které nejsou součástí operačního systému a při načtení stránky se stahují ze serveru. Funkce umožňuje pomocí speciálního argumentu použít lokální font a definovat fall-back alternativu v podobě libovolného fontu. Stejně jako u ostatních testů dochází k provedení dotazu na server po vykonání testovacího případu, v tomto případě se požadavky na server fingerprintera provádí při absenci lokálního fontu. [39]

3.6.2.4 Rozšíření webových prohlížečů

Rozšíření pro webové prohlížeče jsou software, pomocí kterého si uživatelé prohlížeč přizpůsobují svým individuálním potřebám. Po nainstalování dojde k rozšíření funkcí prohlížeče o nové služby. Prohlížeče jako Google Chrome nebo Mozilla Firefox poskytují centralizované platformy, přes které dochází k distribuci rozšíření od vývojářů k uživatelům. Průběh instalace je přirovnatelný průběhu instalace aplikace na mobilním zařízeních přes App Store nebo Google Play. [41] [42]

Uživatelské rozhraní rozšíření je implementováno v podobě webové stránky, tedy je využíváno stejných technologií jako u běžných webových stránek. Rozšíření se skládá ze souborů s aplikačním kódem a dalších doplňkových souborů, které jsou potřebné pro správné fungování rozhraní, může se jednat např. o obrázky nebo CSS soubory. Rozhraní je uživatelům zpřístupněno na unikátní URL adrese, která byla rozšíření při instalaci přidělena. Po načtení této adresy v prohlížecí se rozhraní zobrazí jako klasická webová stránka a uživateli je s ní umožněna interakce. [41] [42]

Jak už bylo v předchozím odstavci poznamenáno, rozšíření se skládají ze souborů, které jsou přístupné v prostředí prohlížeče pomocí URL adresy. Některá rozšíření přidávají do globálního objektu prohlížeče své vlastní objekty, které na webových stránkách umožňují komunikaci s rozšířením. Metoda detekce nainstalovaných rozšíření začíná sběrem informací o testovaných rozšířeních, mezi které řadíme např. URL adresu rozhraní rozšíření, názvy jednotlivých souborů a objekty, které rozšíření vytvářejí. Pro každé rozšíření jsou stanoveny testové případy, které umožňují jeho jednoznačnou identifikaci. [43]

Metoda

Metoda následně provádí jednotlivé testové případy. Příklad lze uvést na testu výskytu URL adresy. Mějme rozšíření „roz321“, jehož rozhraní je dostupné na URL adrese „adresa123“. Metoda vyšle požadavek na otevření webové stránky s URL adresou „adresa123“. V

případě, že požadavek vrátí kladnou odpověď, tedy HTML kód rozhraní rozšíření, se test vyhodnocuje kladně a rozšíření „roz321“ lze považovat za nainstalované.

Některá rozšíření záměrně neodpovídají na požadavky bez autentizačních parametrů, pomocí kterých zabraňují neautorizovanému přístupu k těmto souborům, nicméně bylo zjištěno, že takové požadavky trvají měřitelně delší dobu, a i přes selhání požadavku lze takový požadavek vyhodnotit jako provedený a tím potvrdit existenci instalace rozšíření v prohlížeči. [43]

3.6.2.5 Canvas a WebGL

Princip fungování Canvas elementu

Canvas element je HTML prvek dostupný ve všech moderních prohlížečích, který poskytuje prostor pro vykreslování grafiky na webových stránkách. Umí pracovat i s textem, který se stylizuje obdobně jako u CSS. Po inicializování se s rozhraním vytvořeného elementu pracuje prostřednictvím JavaScriptu. Rozhraní dále zpřístupňuje technologii WebGL, která umožnuje vykreslování trojrozměrné grafiky za použití hardwarové akcelerace. [44] [45]

Unikátní charakteristiky jednotlivých uživatelských prostředí mají za následek rozdílné vykreslování zadaných vstupů. Tyto vstupy jsou často alfanumerického charakteru s tím, že výběr konkrétního fontu textu a jeho velikosti probíhá systematicky za účelem zvýšení entropie výsledků. [46] [47] [10]

Alternativou k vykreslování textu je vykreslování plnohodnotné grafiky, která se nejčastěji sestává z barev, čar, křivek, dvourozměrných nebo třírozměrných obrazců, textur a dalších grafických prvků. Zvyšování komplexity vykreslované grafiky zvyšuje pravděpodobnost, že se rozdíly mezi uživatelskými prostředími pozorovatelně projeví. Autoři metody přisuzují tvorbu rozdílů mezi prostředími především faktorům jako jsou operační systém, verze prohlížeče, grafická karta, nainstalované fonty, antialiasing a sub-pixelové vykreslování. Použití hardwarové akcelerace při vykreslování některých grafických scén přímo přispívá k vytváření rozdílů. [46] [47]

Metoda

Proces testování pomocí této metody začíná výběrem vykreslovaných dat. Při spuštění programu v prohlížeči uživatele dojde k vytvoření Canvas elementu s předdefinovanými vstupními parametry. Tento element obvykle nebývá uživatelům zobrazen, tudíž celý test

probíhá na pozadí stránky bez uživatelova vědomí. Po skončení vykreslování se k výsledkům přistupuje pomocí rozhraní vytvořeného elementu. Výsledkem je obrázek zadaných vstupních parametrů, který rozhraní vrací v zakódované podobě pomocí kódování Base64. Vzhledem k značnému rozsahu velikosti výsledku je přenos po sítí nepraktický a doporučuje se vytvoření hashe výsledku, který pak může být použit pro další zpracování.

Součástí metody je shromažďování atributů WebGL, které jsou volně přístupné přes rozhraní Canvas elementu. Hodnoty těchto atributů nesou vysoce konkrétní informace o grafické kartě uživatelova systému a jejích ovladačích. Autoři popisují postupy pro získání informací jako je název modelu grafické karty, název výrobce grafické karty nebo výpis technologií podporovaných grafickou kartou. [46]

3.6.2.6 IP adresa

IP adresa jako jednoznačný identifikátor

IP adresa je jedinečným identifikátorem, který se přiděluje všem zařízením připojených k počítačové sítí, která ke komunikaci využívá Internetový Protokol. IP adresa zajišťuje identifikaci zařízení nebo síťového rozhraní a udává informace o umístění zařízení v síti. V praxi se využívají verze IPv4 a IPv6.

Při přístupu uživatele na webové stránky dojde k navázání spojení mezi uživatelem a serverem a společně s dalšími daty dojde k předání informací o IP adrese uživatele. IP adresy jsou zároveň dostatečně distinktivní, což umožňuje s vysokou spolehlivostí sledovat jednotlivé uživatele. Z tohoto důvodu jsou jedním z nejstarších atributů, které fingerprinteři zpracovávají. [48]

Obrana uživatelů

Uživatelé, kteří z různých důvodu nechtějí informace o své IP adrese sdílet, v praxi používají anonymizační řešení jako jsou VPN nebo Proxy servery. Taková řešení se v praxi používají při obcházení cenzury a dalšího blokování služeb na internetu ze strany internetových poskytovatelů, dále také pro obcházení požadavků ze strany webových stránek na geolokaci uživatele. Uživatel tak může získat přístup k v jeho zemi nepřístupným webovým stránkám nebo využít rozdílného nacenění pro jednotlivé země konkrétní internetové služby (např. Netflix).

Mezi uživatelem a anonymizačním řešením vzniká sítový tunel, kterým v zašifrované podobě prochází veškeré požadavky uživatele. Požadavky jsou tímto tunelem předávány a anonymizační řešení po přijetí tyto požadavky provede a odpovědi následně pošle zpět uživateli. Uživatel v takovém případě vystupuje pod IP adresou anonymizačního řešení, respektive jeho serveru, který vykonává uživatelovy požadavky. [49]

Uživatele používající anonymizační řešení nelze bez dodatečných informací sledovat, jelikož používají cizí IP adresy, které se navíc s různou intenzitou v čase mění. Uživatel tak teoreticky může při každém požadavku vystupovat jako nový uživatel. Existují metody, které umožňují rozpoznat uživatele, který zakrývá svou IP adresu pomocí anonymizačních řešení, případně dokonce odhalují jeho reálnou IP adresu a rozšiřují ji o další atributy. [50]

Metoda

Technologie WebRTC je dostupná ve všech moderních prohlížečích. Poskytuje metody pro vytvoření peer-to-peer komunikace mezi prohlížeči, pomocí které lze mezi uživateli přenášet různá data. Navazování připojení probíhá pomocí prostředníka (serveru), pomocí kterého si klienti vymění informace o jejich umístění v sítí. Následně dojde k vytvoření přímého připojení mezi klienty. Prohlížeče neposkytují prostředky pro sledovaní a blokaci sítové aktivity takto vytvořené komunikace, s tím, že ji nezobrazují ani ve vývojářské konzoli prohlížeče. [51] [50]

Metoda využívá právě faktu, že se klient musí pro vytvoření spojení jednoznačně lokalizovat v sítí. Při načtení webové stránky se spustí program, který zahájí vytváření spojení pomocí WebRTC, při němž dochází ke sběru věřených a soukromých IP adres uživatelova zařízení, tyto informace můžou být následně libovolně využity. [50]

Výsledkem je znalost uživatelovy reálné IP adresy. Fingerprinter po získání těchto výsledků snadno ověří, zda se uživatelova reálná IP adresa shoduje s IP adresou, ze které byly výsledky obdrženy. Uživatel, jehož IP adresy se neshodují, se s nejvyšší pravděpodobností snaží ukrýt svou pravou identitu.

4 Praktická část

Cílem této práce je návrh a sestrojení řešení pro vytváření otisků webových prohlížečů uživatelů. Dalším cílem je sběr dat pomocí vyvinutého řešení v praxi a následné zhodnocení shromážděných dat.

Praktickou část lze rozdělit na dvě základní části. Při zpracovávání obou částí bylo vycházeno z poznatků získaných při vypracování teoretické části, které byly doplněny o relevantní informace potřebné pro vypracování praktické části a o osobní názory a zkušenosti autora této práce, na jejichž základě došlo k průběžnému plnění těchto cílů.

První část praktické části je zaměřena na teoretický návrh a následnou konstrukci algoritmu pro vytváření otisku webového prohlížeče. Před začátkem prací došlo k systematickému rozdělení vývoje algoritmu na jednotlivé části: návrh architektury řešení, výběr zkoumaných atributů a samotný vývoj řešení. Výsledkem této části je kompletní algoritmus připravený k použití v praxi.

Druhá část popisuje nasazení zkonstruovaného algoritmu do praxe za účelem získání dat uživatelů, kteří byli vystaveni algoritmu při prohlížení webových stránek, na kterých se algoritmus nacházel. Získaná data jsou tvořena hodnotami jednotlivých atributů, které byly v prostředí prohlížeče získávány pomocí metod popsaných v kapitole 4.2 Výběr zkoumaných atributů.

Data byla dále zpracovávána a byly z nich vytvořeny otisky webového prohlížeče jednotlivých uživatelů. Všechny takto získané otisky lze považovat za mezivýsledek druhé části, který byl dále zpracován a kvantitativně a kvalitativně zhodnocen, aby byly získány charakteristiky těchto otisků. Na základě těchto charakteristik byla dále zhodnocena výkonnost samotného algoritmu a byla stanovena doporučení pro případný další vývoj.

4.1 Návrh architektury řešení

Na začátku návrhu došlo ke stanovení požadavků na fungování algoritmu a dalších částí řešení. Vycházelo se z toho, že má dojít ke sběru dat v prohlížeči uživatele a následně k odeslání těchto dat z prohlížeče uživatele takovým způsobem, aby je bylo možné pro účely této práce dále zpracovávat.

Dále byla stanovena podmínka, že musí dojít k implementaci dodatečných faktorů identifikace jednotlivých uživatelů. Tuto podmínku bylo nutné splnit, aby mohlo dojít ke správnému zpracování otisků webových prohlížečů. Pro poskytnutí směrodatného zhodnocení algoritmu je nutné vědět, které otisky byly vygenerovány stejnými uživateli. Duplicitní otisky se musí odstranit, což má vliv na četnost výskytu jednotlivých otisků a ta pak na hodnocení výkonnosti algoritmu jako celku. Pro řešení tohoto problému byli uživatelé označování pomocí kombinace IP adresy a unikátního identifikátoru uživatele, který se uloží do *localStorage* prohlížeče, ve kterém je nižší pravděpodobnost jeho smazaní.

Na základě uvedených požadavků bylo řešení rozděleno na tři základní části: **frontendová**, **backendová** a **databázová**. Pro každou část byla provedena analýza dostupných technologií, na základě které byly stanoveny technologie, které byly použity při vývoji nebo zajištění fungování dané části.

4.1.1 Návrh frontendové části

Frontendová část zajišťuje sběr dat ve webovém prohlížeči uživatele a jejich odesílání na backendovou část. Uživatel proces sběru dat zahájí návštěvou webové stránky, kde je nasazen algoritmus. Po načtení HTML kódu stránky prohlížeč uživatele stáhne zdrojový kód algoritmu ze serverů vlastníka webové stránky, který je následně spuštěn. Algoritmus po spuštění pomocí vytvořené metody tvorby atributů přistupuje k hodnotám vybraných atributů. Po skončení sběru hodnot algoritmus odesílá výsledky společně s vytvořeným identifikátorem uživatele ve formátu JSON pomocí HTTP POST požadavku.

Zvolená technologie

Pro tvorbu samotného algoritmu pro sběr hodnot a atributů prostředí webového prohlížeče uživatele byl použit programovací jazyk **JavaScript**.

Hlavním důvodem pro použití **JavaScriptu** je jeho již zmíněná nativní podpora všemi hlavními webovými prohlížeči, která garantuje fungování algoritmu napříč celým spektrem webových prohlížečů. Za další výhody lze označit přívětivou syntaxi a jednoduchost použití.

4.1.2 Návrh backendové části

Backendová část zajišťuje příjem dat od frontendové části a ukládání získaných dat do databáze. Je tvořena serverem a aplikační logikou, která je schopna požadavky s daty uživatelů zpracovat. Po přijetí požadavku od prohlížeče uživatele jsou data prostřednictvím programového rozhraní uložena do databáze. Kromě samotných dat z prohlížeče uživatele je ukládána IP adresa, ze které byl požadavek obdržen, a hlavička požadavku.

Zvolené technologie

Backendová část byla vytvořena pomocí technologií **Node.js** a **frameworku Express**, které společně vytvářejí přístupové cesty, na které jsou zasílány požadavky od frontendové části, a aplikační logiku, která data v obdržených požadavcích zpracuje a uloží do databáze, pro kterou byla vybrána technologie **MongoDB**. Důvodem pro zvolení Node.js a frameworku Express je jejich oblíbenost mezi programátory, což se promítá v množství dostupných knihoven a dalších rozšíření [52]. Další výhodou je využití stejného programovacího jazyka jako při vývoji frontendové části, tj. JavaScript. Databáze MongoDB byla vybrána z důvodu podpory flexibilní datové struktury, která je vhodná pro potřeby řešení.

Jako hosting byla zvolena služba **Elastic Beanstalk od společnosti AWS**. Důvody pro zvolení této služby byly především možnost automatického škálování aplikace, jednoduchost použití a předchozí dobrá zkušenost autora této práce s touto službou.

- **Nodej.js** je multiplatformní běhové prostředí programovacího jazyka JavaScript. Vývojářům umožňuje spouštět zdrojový kód napsaný v JavaScriptu. Obsahuje vestavěné moduly pro práci se systémem souborů, sítí a dalšími funkcemi souvisejícími s operačním systémem počítače. Byl vybudován na základech V8 JavaScript enginu od společnosti Google, který se používá v prohlížečích Google Chrome.

  Na rozdíl od JavaScriptu, který se používá hlavně pro vývoj webových stránek a aplikací, se Node.js zaměřuje na programování na straně serveru, při kterém se vyžaduje podmínka vysokého výkonu a škálovatelnosti aplikací. [53] [54]

- **Express** je populární framework pro Node.js, který se používá při vývoji backendové části webových aplikací. Poskytuje služby a funkce umožňující flexibilní a minimalistický přistup k vývoji webových aplikací. Mezi tyto funkce lze zařadit funkce template, routing nebo middleware, které vývojářům umožňují zpracovávat jakékoliv typy HTTP požadavků a následně na ně libovolně odpovídat. Jako příklad z praxe lze uvést tvorbu rozhraní, která vykonávají požadavky uživatelů webových aplikací, nebo skriptování na straně serveru. [54]

- **AWS Elastic Beanstalk** je služba od společnosti AWS, která je největším poskytovatelem cloudových služeb na světě. Využívá cloudového modelu „platforma jako služba“, ve kterém se vývojáři zabývají jen vývojem samotné aplikace. Infrastruktura (fyzické připojení a servery), administrace a škálování serveru, instalace softwaru a běh samotné aplikace jsou spravovány výhradně službou Elastic Beanstalk. Aplikace se na servery nasazuje po nahrání kódu pomocí správcovské konzole AWS.

  Vývojářům služba Elastic Beanstalk dále poskytuje sadu nástrojů, které usnadňují monitoring a management aplikace. Jednou z hlavních výhod této služby je schopnost reakce na zvýšení poptávky po službě, čímž je aplikace automaticky škálována prostřednictvím přidávání nebo případně odebírání serverů. [55]

4.1.3 Návrh databázové části

Databázová část zajišťuje uchování dat v čase. Hraje důležitou roli při zpracovávání obdržených výsledků, které jsou před samotným prováděním zpracování z databáze staženy.

Databázová část byla vyřešena pomocí cloudové databáze **MongoDB**. Databáze byla zvolena primárně z důvodu nestrukturované podoby sbíraných dat. MongoDB stanovuje jednoduchou datovou strukturu a principy ukládání výsledků v podobě dokumentů na bázi formátu JSON, což zjednodušuje zpracování výsledků. Dalším důvodem pro zvolení byla přítomnost moderního rozhraní a knihoven, které umožňují příjemné ovládání a rychlosti nasazení tohoto řešení.

- **MongoDB** je dokumentově orientovaný databázový systém používaný pro ukládání nestrukturovaných a polostrukturovaných dat. Jeho hlavními přednostmi jsou vysoká škálovatelnost, flexibilita a efektivnost při zpracovávání požadavků. Patří do skupiny databázových systémů založených na modelu NoSQL, který na rozdíl od tradičních relačních databází není závislý na použití tabulek, řádků a sloupců. Veškerá data se ukládají v podobě dokumentů, které svou strukturou připomínají formát JSON. [56]

4.2 Výběr zkoumaných atributů

Kvalita algoritmu pro vytváření otisku webového prohlížeče nespočívá v kvalitě napsaného kódu nebo v rychlosti programovacího jazyka. Dostačující kód v tomto případě je takový, který funguje s tím, že jeho vykonání trvá přiměřenou dobu. Nicméně, na druhé straně je velmi podstatné, které atributy jsou zkoumány a jakými způsoby.

Správný výběr atributů je naprostým základem pro efektivní fungování algoritmu. Kombinace hodnot jednotlivých atributů shromážděných algoritmem vytváří otisk webového prohlížeče uživatele. Ať už je důvod pro použití algoritmu jakýkoliv, vždy je v zájmu fingerprinterů dostáhnout co nejvyšší entropie hodnot výsledných otisků, jelikož ta přímo udává míru výkonnosti takového algoritmu. Důsledky jejich výběrů jsou tedy přímo pozorovatelné pomocí kvalitativních a kvantitativních charakteristik otisku.

Výkonnost atributů nelze předem předvídat, což ztěžuje vývoj algoritmů pro tvorbu otisků. Autorem práce je navržena strategie výběru atributů, podle níž jedna část atributů vychází z prací ostatních autorů, které v různé míře (většinou malé) informují o výsledcích získaných

z jednotlivých atributů, a u druhé jsou použity představivost a úsudek autora této práce, který na základě průzkumu objektů v prohlížeči a dalších proměnných, pokud možno objektivně, vybere jednotlivé atributy.

Při zpracování se postupovalo podle zvolené strategie a byl vytvořen seznam vhodných atributů a metod k získání jejich hodnot. Společně s výběrem došlo u každého atributu ke stanovení postupů, které objasňuje důvody pro zařazení atributu.

Podstatnou část vybraných atributů představují fixní atributy různého druhu, které se vyskytují v objektech prohlížeče *Screen*, *Navigator*, *Window* a *Performance*. Použití některých z těchto objektů lze odůvodnit poznatky získanými při vypracování teoretické práce. Napříč autory, kteří se zabývají obdobným vývojem, jsou tyto objekty považovány za bohatý zdroj atributů nesoucí hodnoty o zařízení uživatele. Poznatky autorů byly po bližším zkoumaní objektů webových prohlížečů autorem této práce rozšířeny o nové atributy vyskytující se také napříč objekty *Screen*, *Navigator* a dále o objekty *Window* a *Performance* společně s jejich atributy.

4.2.1 Atributy objektu Navigator

Tabulka 3 Seznam vybraných atributů z objektu Navigator

Název atributu
navigator.userAgent
navigator.platform
navigator.language
navigator.languages
navigator.doNotTrack
navigator.cookieEnabled
navigator.hardwareConcurrency
navigator.deviceMemory
navigator.maxTouchPoints
performance.jsHeapSizeLimit

Výše uvedené atributy objektu *Navigator* poskytují nejdůležitější informace o zařízení uživatele s tím, že se očekává vysoká proměnlivost jejich hodnot mezi prostředími uživatelů.

Jako atributy s nejvyšší pravděpodobností vhodnosti pro použití při vytváření otisku prohlížeče byly stanoveny atributy *navigator.userAgent*, *navigator.language*, *navigator.languages*, *navigator.hardwareConcurrency* a *navigator.maxTouchPoints*.

Na základě získaných poznatků a dalšího zkoumání bylo zjištěno, že hodnoty atributu *userAgent* jsou vysoce proměnlivé. Proměnlivost lze ukázat na příkladu mobilních zařízení, u kterých se do hodnoty atributu *userAgent* propisují informace o výrobci zařízení, modelové číslo, nastavení jazykové lokalizace a další podobné vysoce specifické údaje.

Atributy *navigator.language* a *navigator.languages* byly vybrány pro množství kombinací možných hodnot vzhledem k vysokému počtu existujících lingvistických jazyků.

Kromě atributů objektu *Navigator* je do této skupiny přidán atribut *jsHeapSizeLimit* z objektu *Performance*, jehož hodnota určuje maximální možnou velikost operační paměti, kterou si může prohlížeč od operačního systému vyžádat.

Zdrojový kód 1 – Přístup k hodnotám objektu Navigator

```
result.navigator = {};
result.navigator.userAgent = navigator.userAgent;
result.navigator.platform = navigator.platform;
result.navigator.language = navigator.language;
result.navigator.languages = navigator.languages;
result.navigator.doNotTrack = navigator.doNotTrack;
result.navigator.cookieEnabled = navigator.cookieEnabled;
result.navigator.hardwareConcurrency = navigator.hardwareConcurrency;
result.navigator.deviceMemory = navigator.deviceMemory;
result.navigator.maxTouchPoints = navigator.maxTouchPoints;
result.performance = {};
result.performance.jsHeapSizeLimit = window.performance &&
window.performance.memory ? window.performance.memory.jsHeapSizeLimit :
null;
```

4.2.2 Atributy objektu Screen

Tabulka 4 Seznam vybraných atributů z objektu Screen

Název atributu
screen.availWidth
screen.availHeight
screen.availLeft
screen.availTop
screen.width
screen.height
screen.colorDepth
screen.pixelDepth

Výše uvedené atributy objektu *Screen* poskytují veškeré relevantní informace o obrazovce zařízení uživatele. Předpokládá se, že vzhledem k množství zařízení na trhu a dalších uživatelských konfigurací je vysoce pravděpodobné, že se tyto hodnoty budou mezi prostředími měnit.

Zdrojový kód 2 – Přístup k hodnotám objektu Screen

```
result.screen = {};
result.screen.availWidth = screen.availWidth;
result.screen.availHeight = screen.availHeight;
result.screen.availLeft = screen.availLeft;
result.screen.availTop = screen.availTop;
result.screen.width = screen.width;
result.screen.height = screen.height;
result.screen.colorDepth = screen.colorDepth;
result.screen.pixelDepth = screen.pixelDepth;
```

4.2.3 Atributy objektu Window

Tabulka 5 Seznam vybraných atributů objektu Window

Název atributu
window.screenX
window.screenY
window.outerHeight
window.outerWidth
window.innerWidth
window.innerHeight
window.devicePixelRatio

Výše uvedené atributy objektu *Window* poskytují informace o vlastnostech okně prohlížeče.

Výchozí hodnoty těchto atributů jsou stanoveny vlastnostmi obrazovky zařízení uživatele, nicméně očekává se, že uživatel bude při práci s oknem prohlížeče různými způsoby manipulovat, především měnit jeho velikost.

Stanovený předpoklad očekává vysokou míru proměnlivostí hodnot těchto atributů jak mezi prostředími uživatelů, tak i v prostředí konkrétních uživatelů. Výhodou těchto atributů je jejich vysoká míra entropie, jelikož existuje velké množství možných konfigurací velikosti okna prohlížeče a jeho umístění na obrazovce zařízení uživatele. Zásadní nevýhodou těchto atributů však je nestabilita jejich hodnot, což z nich dělá nástroj především pro krátkodobou identifikaci.

Hodnoty *window.screenX* a *window.screenY* například udávají informace o pozici okna na obrazovce, resp. jeho relativní vzdálenost od levého horního rohu v pixelech. Hodnoty atributů *window.outerHeight*, *window.outerWidth* reprezentují velikost okna prohlížeče včetně ovládacích prvků (pole pro URL adresu, záložky apod.), naopak *window.innerWidth* a *window.innerHeight* reprezentují velikost okna bez ovládacích prvků.

Zdrojový kód 3 – Přístup k hodnotám objektu Window

```
result.window = {};
result.window.screenX = window.screenX;
result.window.screenY = window.screenY;
result.window.outerHeight = window.outerHeight;
result.window.outerWidth = window.outerWidth;
result.window.innerWidth = window.innerWidth;
result.window.innerHeight = window.innerHeight;
result.window.devicePixelRatio = window.devicePixelRatio;
```

4.2.4 Další atributy

Zbytek vybraných atributů je tvořen jak dynamickými, tak i fixními atributy. Mezi atributy neexistuje stejnorodý způsob získávání nebo původ, který by je podle nějakého klíče spojoval. Velká část této skupiny atributů nese informace o jazykové lokalizaci prostředí uživatele, které v kombinaci s jazykovými preferencemi prohlížeče uživatele získaných v objektu *Navigator* slouží jako prostředek pro geolokaci uživatele a dále zvyšují entropii otisku.

Časové pásmo

Objekt *Date* reprezentuje libovolnou chvíli v čase. Při zavolání přes konstruktor dojde k vytvoření objektu a vrácení textového řetězce, který obsahuje informace o čase vytvoření včetně časového pásma. Příklad výsledku: Mon Mar 05 2023 03:08:13 GMT+0100 (Central European Standard Time). Důležitá je část s časovým pásmem a dále pojmenování časového pásma. Toto pojmenování je zobrazováno v jazyce webového prohlížeče uživatele.

Dále dochází ke stanovení časového pásma uživatele pomocí funkce *getTimezoneOffset*, která udává rozdíl v minutách od časového pásma UTC. Předpokládá se vysoká míra entropie hodnot vzhledem k počtu časových pásem.

Zdrojový kód 4 – Časové pásmo

```
let date = new Date;
result.tests.dateString = date;
result.tests.timezone = date.getTimezoneOffset();
```

Grafická karta a její ovladače

Důležitými atributy jsou informace o grafické kartě a jejích ovladačích. Předpokládá se použití různých grafických karet nebo čipů v zařízeních uživatelů. Na základě toho je tato skupina atributů považována za relevantní pro potřeby vytváření otisku webového prohlížeče.

Funkce pro sběr dat začíná vytvořením elementu canvas pomocí funkce *document.createElement*. Přes kontext canvas elementu se následně přistupuje k rozhraní technologie WebGL. Dochází k ověření úspěšného průběhu příkazů a začíná sběr atributů z prostředí technologie WebGL, kterými jsou název modelu a výrobce grafické karty, přítomná rozšíření WebGL a jejich počet.

Příklad výsledku 1 – Grafická karta a její ovladače

```
{
    "extensionList": [
        "ANGLE_instanced_arrays",
        "EXT_blend_minmax",
        "EXT_color_buffer_half_float",
        .
        .
    ],
    "extensionLength": 29,
    "vendor": "Google Inc. (NVIDIA)",
    "renderer": "ANGLE (NVIDIA, NVIDIA GeForce GTX 980 Ti Direct3D11 vs_5_0
ps_5_0, D3D11)"
}
```

Zdrojový kód 5 – Grafická karta a její ovladače

```
function gpuInfo() {
    let canvas = document.createElement("canvas");
    let canvasWebGlContext = canvas.getContext("webgl");

    let results = {};

    if (canvasWebGlContext) {
        let extensions = canvasWebGlContext.getSupportedExtensions();
        if (extensions) {
            results.extensionList = extensions;
            results.extensionLength = extensions.length;

            if (extensions.includes("WEBGL_debug_renderer_info")) {
                results.vendor =
canvasWebGlContext.getParameter(canvasWebGlContext.getExtension("WEBGL_debug
_renderer_info").UNMASKED_VENDOR_WEBGL);
                results.renderer =
canvasWebGlContext.getParameter(canvasWebGlContext.getExtension("WEBGL_debug
_renderer_info").UNMASKED_RENDERER_WEBGL);
            };
        };
    };

    return results;
};

result.tests.gpu = gpuInfo();
```

Hlasy syntezátoru řeči

Další metoda získává informace o v systému nainstalovaných hlasech syntezátorů řeči, které jsou dostupné v naprosté většině prohlížečů. Syntezátory řeči se používají pro tvorbu umělé lidské řeči, s tím, že operační systémy disponují různou nainstalovanou nabídkou hlasů, pomocí kterých dochází k vytvoření řeči. To, které hlasy budou nainstalovány, je závislé na jazykové lokalizaci systému uživatele. Jedná se tak o další zdroj informací pro určení geolokace uživatele.

Po spuštění metody dochází ke kontrole, že je technologie prohlížečem uživatele podporována. Hlasy jsou do prohlížeče načítány asynchronně při načtení webové stránky. Z tohoto důvodu dochází k vytvoření posluchače událostí *speechSynthesis.onvoiceschanged*, který je prohlížečem po načtení hlasů zavolán, což zajišťuje, že se část kódu algoritmu, která k hlasům přistupuje, spustí ve správnou chvíli. Pokud by nedošlo k vykonání tohoto kroku, hlasy by algoritmem nemusely být detekovány. Seznam s obdrženými hlasy je následně procházen a pro každý hlas jsou shromážděn jeho název a jazyková lokalizace.

Příklad výsledku 2 – Hlasy syntezátoru řeči

```
result = "Microsoft Jakub - Czech (Czech Republic)_cs-CZ;Microsoft David - English (United States)_en-US;Microsoft Mark - English (United States)_en-US;Microsoft Zira - English (United States)_en-US;Google Deutsch_de-DE;Google US English_en-US;Google UK English Female_en-GB;Google UK English Male_en-GB;Google español_es-ES;Google español de Estados Unidos_es-US;Google français_fr-FR;Google हिन्दी_hi-IN;Google Bahasa Indonesia_id-ID;Google italiano_it-IT;Google 日本語_ja-JP;Google 한국의_ko-KR;Google Nederlands_nl-NL;Google polski_pl-PL;Google português do Brasil_pt-BR;Google русский_ru-RU;Google 普通话（中国大陆）_zh-CN;Google 粵語（香港）_zh-HK;Google 國語（臺灣）_zh-TW;"
```

Zdrojový kód 6 – Hlasy syntezátoru řeči

```
function getSpeechSynthesisVoices() {
    return new Promise(
        function (resolve, reject) {
            try {
                let timeout = setTimeout(() => {
                    resolve(null);
                }, 500);

                if (window.speechSynthesis) {
                    window.speechSynthesis.onvoiceschanged = function () {
                        let result = "";

                        if (window.speechSynthesis) {
                            let voices = window.speechSynthesis.getVoices();
                            if (voices.length > 0) {
                                for (let counter = 0; counter <
voices.length; counter++) {
                                    result += voices[counter].voiceURI + "_"
+ voices[counter].lang + ";";
                                };
                            };
                        } else {
                            result = null;
                        };

                        clearTimeout(timeout);
                        resolve(result);
                    };
                };
            } catch {
                resolve(null);
            };
        }
    );
};
```

4.3 Vývoj jednotlivých částí řešení

Došlo k vývoji každé z částí, které vznikly rozděleném řešení v kapitole 4.1 Návrh architektury řešení. V průběhu vývoje jednotlivých částí byly použity technologie a principy definované ve zmíněné kapitole. Při vývoji byly respektovány požadavky na jednotlivé části řešení, které byly rovněž specifikovány v uvedené kapitole. Tyto požadavky byly v průběhu vývoje do jednotlivých části zapracovány. Vyvinuté částí dohromady tvoří celek, který umožňuje sběr hodnot atributů z prostředí webových prohlížečů uživatelů, jejich odeslání na vzdálený server a následné zpracování a uložení do databáze. Kompletní zdrojový kód je k dispozici v přílohách této práce.

Kapitola Návrh architektury řešení definuje tři části řešení: **frontendová** část, **backendová** část a **databázová** část. Následuje přehled průběhu vývoje každé z těchto částí:

4.3.1 Frontendová část

Na počátku vývoje byla vytvořena základní struktura programu, která se sestává z hlavní funkce obsahující veškeré příkazy, které mají být vykonány. Po spuštění programu dochází k automatickému spuštění této funkce, jejíž pomocí dochází ke sběru informací o webovém prohlížeči uživatele a odeslání dat na backendovou část.

První podmínkou po spuštění obsahu hlavní funkce je přítomnost uložiště *localStorage* ve webovém prohlížeči uživatele. Tato podmínka ověřuje, zda je možné vytvořit alternativní jednoznačný identifikátor prohlížeče uživatele, jehož potřeba byla objasněna v kapitole Návrh architektury řešení. Architektura řešení stanovuje, že pro ukládání těchto identifikátorů bude použita právě technologie *localStorage*.

Pokud je podmínka splněna, dochází ke spuštění vnořené části kódu. V opačném případě, tedy v takovém, kdy *localStorage* v prohlížeči uživatele chybí, dochází k ukončení programu, a tedy k neúspěšnému pokusu o získání atributů z prostředí prohlížeče uživatele.

Vnořená část kódu nejprve prohledává uložiště *localStorage* a zjišťuje, zda v minulosti došlo k vytvoření identifikátoru. V případě, že dojde k nalezení identifikátoru v úložišti *localStorage*, je hodnota tohoto identifikátoru použita při vykonávání zbytku programu. V případě absence identifikátoru v úložišti *localStorage* je vytvořen nový identifikátor a následně uložen do úložiště *localStorage*.

Po získání identifikátoru následuje oblast kódu, ve které dochází ke sběru hodnot jednotlivých atributů prostředí webového prohlížeče uživatele. Je vytvořen objekt *result*, do kterého jsou ukládány hodnoty jednotlivých atributů získané po vykonání zdrojového kódu metod, který byl pro každou z nich vytvořen v kapitole Výběr zkoumaných atributů.

Program je zakončen skupinou příkazů, jejímž úkolem je odeslat obsah objektu *result* a identifikátor prohlížeče na backendovou část, která je reprezentována veřejně přístupnou URL adresou.

Výsledkem vývoje frontendové části je soubor s příponou *script.js*, který obsahuje zdrojový kód napsaný v jazyce JavaScript. Tento soubor lze spouštět v prohlížeči.

Zdrojový kód 7 – Hlavní funkce

```
(function () {
    if (typeof Storage !== "undefined") {
        // nastavení ID uživatele
        let userID = localStorage.getItem("userID");
        if (userID === null) {
            userID = uuidv4();
            localStorage.setItem("userID", userID);
        }

        let result = {};

        //
        // oblast s metodami pro sběr hodnot
        //

        // odeslání objektu result společně s identifikátorem uživatele
        let request = new XMLHttpRequest();
        request.open("POST", "/analytics");
        request.setRequestHeader("Content-Type", "application/json");

        request.send(JSON.stringify({
            userID: userID,
            result: result
        }));
    };
})();
```

4.3.2 Backendová část

Základními funkcemi, jejichž implementace byla zahrnuta v backendové části, jsou příjem HTTP požadavků, odesílání odpovědí na HTTP požadavky, zpracování přijatých dat a uložení dat do databáze.

Výsledkem backendové části je soubor *index.js*, který obsahuje zdrojový kód programu napsaný v programovacím jazyce JavaScript, a soubory *package.json* a *package-lock.json*, které nesou metadata o projektu Node.js a o použitých modulech. Při vývoji byly použity moduly *express*, *cors* a *mongodb*.

Po spuštění programu dochází k inicializaci použitých modulů Node.js. Dochází k vytvoření klienta MongoDB a k vytvoření připojení k databázi. Dále dochází ke vzniku instance aplikace frameworku Express, inicializaci jejích služeb a k provádění dalšího nastavení jako je vytvoření cest, které jsou pak na venek reprezentovány pomocí URL adres.

Byla definována cesta */analytics*, jejíž handler obsahuje veškerou aplikační logiku zajišťující, že při přijetí HTTP požadavku dojde k shromáždění těla a hlavičky požadavku a času jeho přijetí. Tato data vloží do objektu, který je následně uložen do databáze MongoDB pomocí vytvořeného připojení. Klientovi je v případě úspěšného provedení těchto kroků vrácena odpověď se stavovým kódem *200*. V případě chyby při ukládání dat dojde k nastavení stavového kódu na hodnotu *500*.

4.3.3 Databázová část

Pro zajištění databázové části byla využita služba MongoDB Atlas, která poskytuje plně spravované cloudové databázové služby. Uživatelé této služby se nestarají o správu databáze MongoDB a serverů, na kterých je databáze provozována.

Došlo k založení uživatelského účtu u služby MongoDB Altas. Prostřednictvím administrátorského rozhraní služby byla vytvořena databáze na vzdáleném serveru. Po vytvoření databáze byly obdrženy údaje, které umožňují přístup k databázi.

Od momentu přijetí přístupových údajů k databázi s ní bylo možné libovolně nakládat.

4.4 Nasazení řešení do praxe

Procesu získávání výsledků předcházelo testování vyvinutého řešení. Řešení bylo testováno v domácím prostředí na dostupných zařízeních. Samotné testy spočívaly v ověření správného fungování frontendové části. Backendová a databázová část byla otestována a laděna již při vývoji.

V případě frontendové části se ověřovalo, zda program v jakékoliv chvíli nekončil chybou a dále, zda byly výsledky skutečně odeslány na backendovou část. Testování proběhlo s těmito výsledky:

Tabulka 6 Testovaná zařízení a výsledky testu

Identifikace zařízení – operační systém – webový prohlížeč	Test proběhl úspěšně
Stolní počítač 1 – Windows 11 – Chrome	Ano
Stolní počítač 1 – Windows 11 – Firefox	Ano
Stolní počítač 2 – Windows 7 – Chrome	Ano
Stolní počítač 2 – Windows 7 – Firefox	Ano
Notebook 1 – Windows 11 – Chrome	Ano
Notebook 1 – Windows 11 – Firefox	Ano
iPad Pro – iOS 15 – Safari	Ano
iPad Pro – iOS 15 – Chrome	Ano
iPhone Xs Max – iOS 15 – Safari	Ano
iPhone Xs Max – iOS 15 – Chrome	Ano

Na základě těchto výsledků testování byla potvrzena schopnost navrhovaného řešení shromáždit a odeslat hodnoty jednotlivých atributů.

Následně došlo k nasazení řešení na webové stránky s reálným provozem. Jedná se o stránku, která je podle portálu S*imilarweb (https://www.similarweb.com/)* zařazena mezi 25 tisíc nejnavštěvovanějších stránek světa. Portál dále odhaduje, že stránku v lednu roku 2023 navštívilo 1.7 miliónu uživatelů. Nasazení proběhlo po dohodě s provozovatelem, který zajistil implementaci frontendové části do současné kódové základny webové stránky.

5 Výsledky a diskuse

Program pro sběr dat byl na webové stránce přítomný po dobu 48 hodin. Celkem bylo obdrženo 176 110 záznamů z prohlížečů uživatelů. Došlo k zpracování záznamů a odstranění záznamů na základě následujících kritérií:

- Poškozené záznamy: byl odstraněn jeden záznam, který neobsahoval žádná data.
- Záznamy, který pocházejí od crawlerů (automatizované programy, které procházejí internetové stránky a shromažďují o nich informace) a dalších automatizovaných prohlížečů. Většina automatizovaných nástrojů byla detekovatelná pomocí specifických hodnot atributu *userAgent*. Konkrétně se kontrolovala přítomnost textových řetězců *bot* (robot), *http* (většina crawlerů obsahuje HTTP odkaz na jejich vlastníka) a *lighthouse* (tester výkonnosti webových stránek) v hodnotě atributu *userAgent*. Hodnoty *userAgent*, které byly tímto způsobem odstraněny jsou např.:

  Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.268

  Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; Googlebot/2.1; +http://www.google.com/bot.html) Chrome/108.0.5359.130 Safari/537.36

  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.1.1 Safari/605.1.15 (Applebot/0.1; +http://www.apple.com/go/applebot)

  Dalším kritériem pro detekci automatizovaných nástrojů byly hodnoty atributu časové pásmo. Automatizované nástroje jsou nejčastěji provozovány na serverech, které používají časové pásmo UTC. Byla provedena kontrola, zda se ve textovém řetězci atributu *dateString* nachází textový řetězec *Coordinated Universal Time*. Mezi běžnými uživateli se vyskytuje jen zřídka, např. ve výsledcích od uživatelů z Velké Británie se vyskytovala hodnota *Greenwich Mean Time*, a to i přes fakt, že země leží ve stejné zeměpisné šířce jako písmo UTC.

Celkem bylo odstraněno 2 547 záznamů, což vyústilo v dataset s 173 563 záznamy obsahující informace o prohlížečích uživatelů. Odebrané záznamy tvořily 1,4 % výchozího datasetu a vzhledem k tomuto množství byl tento pokles označen za přijatelný.

Výsledky byly pro lepší prezentaci rozděleny do tří kategorií:

- Obecné informace o obdržených záznamech
- Informace o výkonnosti jednotlivých atributů
- Informace o výkonnosti otisku jako celku

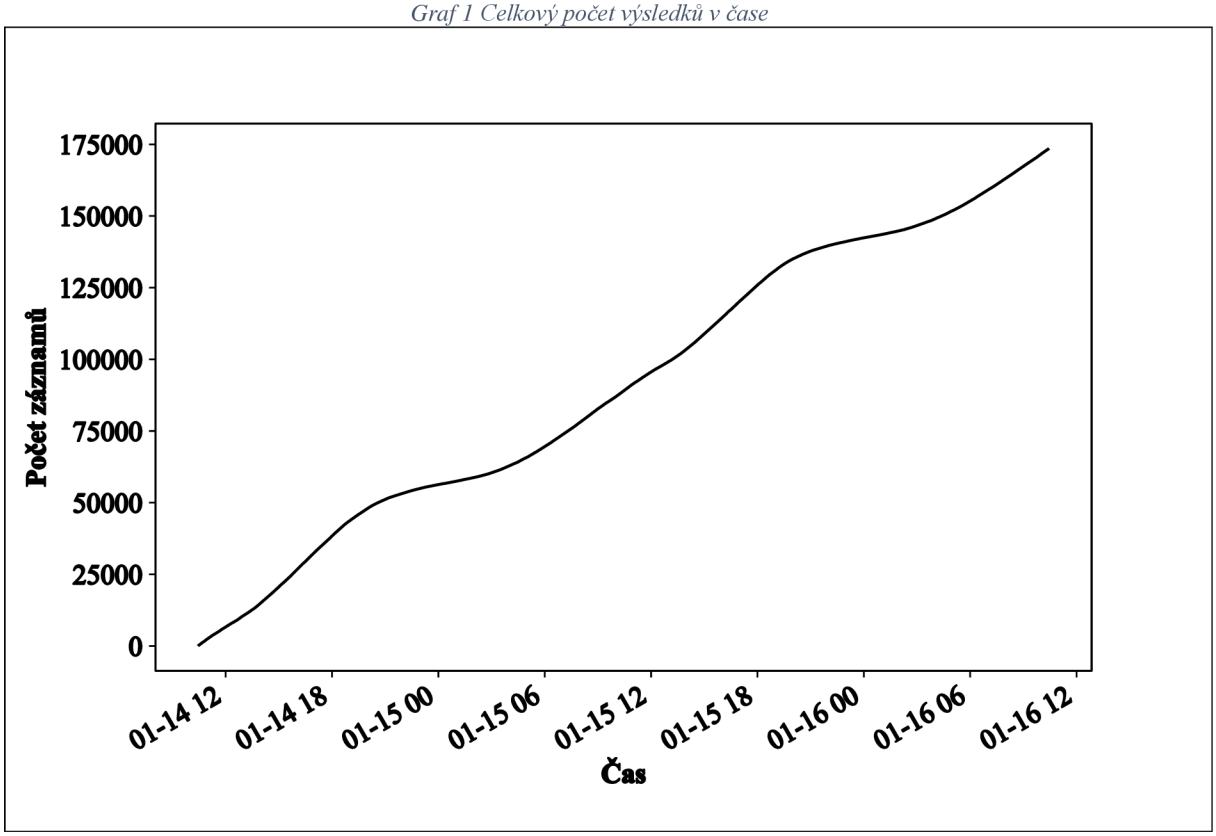
5.1 Obecné informace o obdržených záznamech

Výsledky v této skupině informují o obecných charakteristikách záznamů a průběhu jejich sběru. Následuje rozbor takových charakteristik, které podávají relevantní informace o uživatelích konkrétní stránky:

Průběh sběru záznamů

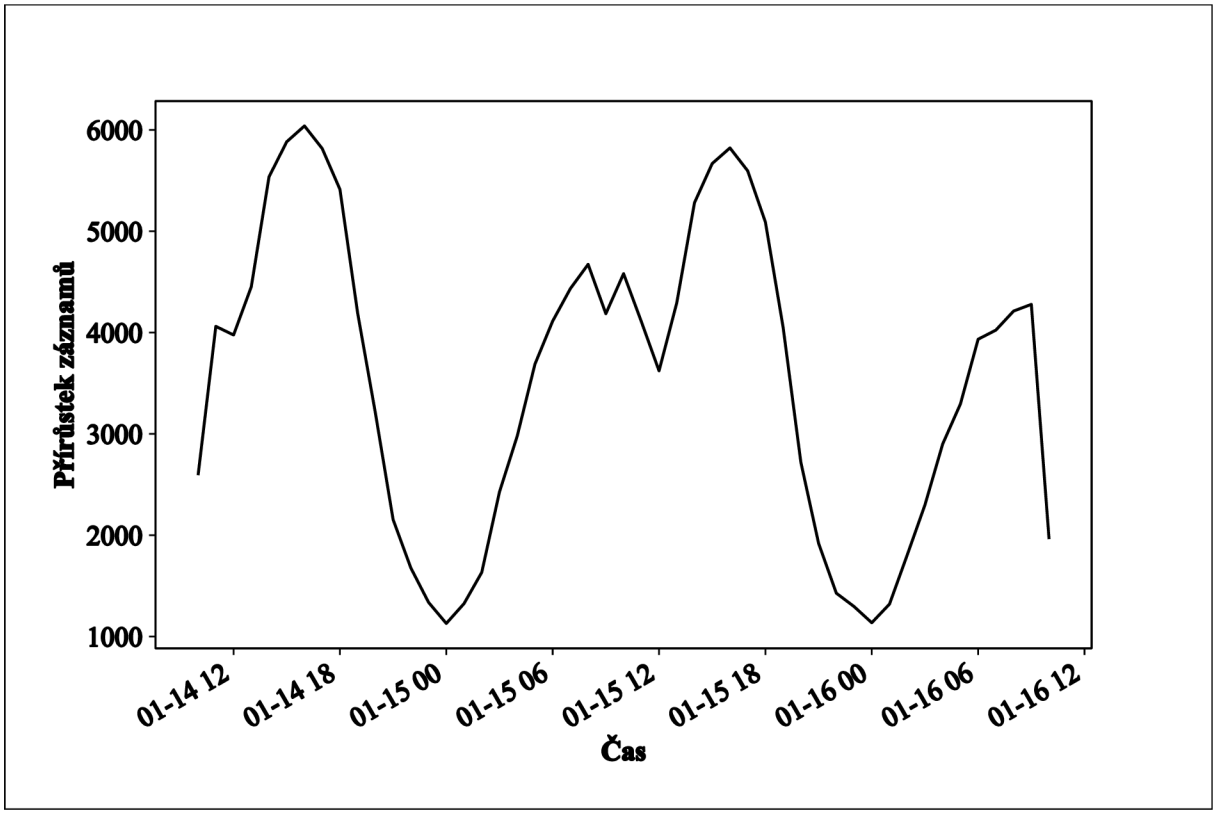
Následující metriky informují o rychlosti sběru dat. Tyto informace vypovídají především o uživateli navštívených webových stránkách a jejich návycích.

Graf 1 Celkový počet výsledků v čase



Průběh sběru dat lze vizualizovat tímto grafem, který ukazuje počet záznamů v čase. Lze pozorovat, že přírůstky záznamů jsou relativně konzistentní.

Graf 2 Přírůstek záznamů v intervalech



Zároveň dochází ke zjištění, že záznamy nejrychleji přibývaly mezi 15. a 17. hodinou. Nejvyšší evidovaný přírůstek byl 6 039 záznamů za hodinu a nejnižší 1 129 záznamů.

Charakteristiky prohlížečů a uživatelů

Metriky, které byly zahrnuty do této kategorie, informují o u identitě uživatelů, jejichž informace byly shromážděny, a o jimi používanými technologiemi.

Z dat bylo zjištěno, že 72 % uživatelů stránky mají v operačním systému nastavené časové pásmo UTC+5, což znamená, že oproti České republice jsou o čtyři hodiny napřed. Při bližším zkoumání bylo zjištěno, že se jedná o uživatele z Indie. Lokace uživatelů byla odhalena pomocí výskytu textového řetězce India Standard Time v hodnotě atributu *dateString* a dále podle nainstalovaných jazyků, které se skládaly z různých kombinací hodnot atributu *languages*. Příkladem nalezených jazyků jsou *hi* (hindština), *ta* (tamilština) nebo *ml* (malajálamština).

Dále proběhla analýza zařízení uživatelů. Více než 78.4 % záznamů bylo vytvořeno na mobilních zařízeních. Zbytek byl rozdělen mezi desktopové počítače, tablety, smart televize a herní konzole.

Tabulka 7 Přehled výskytu typů zařízení

Typ zařízení	Počet záznamů	Vyjádření v procentech
Mobilní zařízení	136 127	78,431 %
Desktopy	35 639	20,534 %
Tablety	1 648	0,950 %
Smart televize	148	0,085 %
Konzole	1	0,001 %

Bylo zjištěno, že nejpočetnějším operačním systémem je systém Android, který se vyskytuje v necelých 73,6 % záznamů. Druhou nejpočetnější skupinou jsou počítače se systémem Windows, které tvoří přibližně 14,8 % záznamů. Kromě zmíněných systémů byla pozorována přítomnost dalších jedenácti operačních systémů.

Tabulka 8 Přehled výskytu operačních systémů zařízení

Operační systém	Počet záznamů	Vyjádření v procentech
Android	127 711	73,582 %
Windows	25 729	14,824 %
iOS	10 153	5,850 %
Linux	6 906	3,979 %
Mac OS	2 720	1,567 %
Chromium OS	123	0,071 %
Neidentifikovaný	90	0,052 %
Ubuntu	66	0,038 %
Tizen	55	0,032 %
HarmonyOS	7	0,004 %
Xbox	1	0,001 %
Firefox OS	1	0,001 %
Chromecast	1	0,001 %

Záznamy poskytují detailní data o výrobci a modelu zařízení uživatele. Celkem bylo nalezeno 2675 unikátních zařízení. Jedná se především o mobilní zařízení, jelikož desktopové počítače a notebooky tyto informace neposkytují anebo je nemají.

Tabulka 9 Přehled výskytu modelů zařízení

Výrobce a model zařízení	Počet záznamů	Vyjádření v procentech
Apple iPhone	9 865	5,7 %
Apple Macintosh	2 720	1,6 %
Xiaomi Redmi Note 8 Pro	1 532	0,9 %
Xiaomi Redmi Note 7 Pro	1 236	0,7 %
Xiaomi Redmi Note 9 M2003J15SC	1 137	0,7 %
Xiaomi Redmi Note 8	1 037	0,6 %
Xiaomi Redmi Note 10S M2101K7BI	1 020	0,6 %
Xiaomi Redmi Note 10 Pro M2101K6P	944	0,5 %
Xiaomi Redmi Note 9 Pro Max	922	0,5 %
Xiaomi Redmi Note 9 Pro	920	0,5 %

Důležitou metrikou je statistika o využívaných webových prohlížečích. Bylo nalezeno 36 unikátních typů prohlížečů. Je zajímavé, že některé prohlížeče nejsou skutečnými prohlížeči, ale prohlížeči integrovanými do mobilních aplikací. Jako příklad lze uvést prohlížeč s názvem *Facebook*, na základě čehož lze odvodit, že uživatel k webové stránce přistupuje přes odkaz v aplikaci Facebook. Nejvíce je v záznamech zastoupen prohlížeč Google Chrome s 86 %.

Dále došlo k identifikaci 711 unikátních verzí prohlížečů. Zde je nejpočetnější hodnotou verze 108.0.0.0 prohlížeče Google Chrome s výskytem v 37 % záznamů, druhou nejpočetnější hodnotou je verze 109.0.0.0 prohlížeče Google Chrome.

Tabulka 10 Přehled výskytu typů prohlížečů

Název prohlížeče	Počet záznamů	Vyjádření v procentech
Chrome	149192	86,0 %
Mobile Safari	6836	3,9 %
Opera	3051	1,8 %
UCBrowser	2824	1,6 %
Firefox	2371	1,4 %
Edge	2170	1,3 %
Chrome WebView	1967	1,1 %
Samsung Browser	1925	1,1 %
Safari	1241	0,7 %
GSA	703	0,4 %

5.2 Informace o výkonnosti jednotlivých atributů

Každý obdržený záznam obsahuje 32 atributů z prostředí webového prohlížeče uživatele a hodnoty těchto atributů. Pro každý atribut byly zjištěny charakteristiky počet unikátních hodnot a hodnoty bez duplicitních výsledků. Tyto atributy lze vysvětlit na následujícím příkladu: mějme list hodnot [1, 1, 2, 2, 3], unikátní hodnoty jsou v takovém případě [1, 2, 3] a hodnoty bez duplicitních výsledků [3].

Tabulka 11 Přehled jednotlivých atributů a výskytu jejich hodnot

Název atributu	Počet unikátních hodnot	Počet hodnot bez duplicitních výskytů
userAgent	16 963	3 921
languages	5 262	1 885
speech	2 754	1 005
innerHeight	1 733	243
innerWidth	1 476	352
outerWidth	1 287	361
outerHeight	1 246	171
jsHeapSizeLimit	1 150	309
rendered	1 010	303
screenX	819	433
availHeight	770	113
availWidth	716	200
height	649	92
width	615	160
devicePixelRatio	453	82
dateString	436	81
screenY	404	186
language	344	99
extensionList	245	31
availTop	175	95
availLeft	156	75
hardwareConcurrency	87	25
timezone	79	35
vendor	77	28
platform	72	58
extensionLength	29	0
maxTouchPoints	16	1
deviceMemory	8	1
colorDepth	5	0
pixelDepth	5	0
doNotTrack	5	0

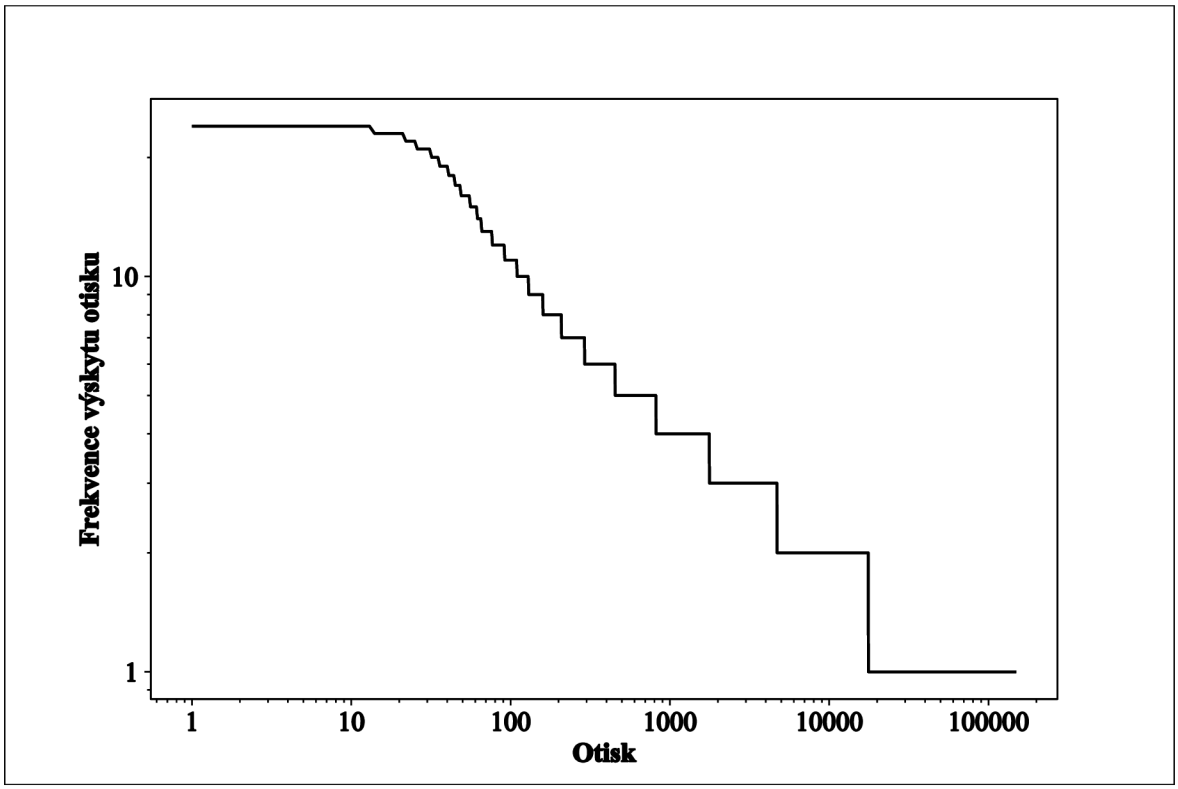
Tabulka s jednotlivými atributy a jejich charakteristikami je seřazena podle charakteristiky počet unikátních hodnot. Obě charakteristiky jsou naprosto rozhodující pro výkonnost řešení pro vytváření otisků. Nejvýkonnější atributy jsou takové, které mají vysoký počet unikátních hodnot a zároveň vysoký počet hodnot bez duplicitních výskytů. Při bližším zkoumání bylo zjištěno, že nejvýkonnějšími atributy jsou atributy *userAgent, languages* a *speech*. Další vhodným zdrojem atributů je objekt *Screen*. Naprostá většina jeho atributů se umisťuje na předních příčkách. Dalším vhodným atributem je atribut *gpu.renderer*, jehož hodnoty nesou informace o modelu grafické karty nebo čipu zařízení uživatele.

5.3 Informace o výkonnosti otisku jako celku

Metriky uvedené v této části se zaměřují na vyhodnocení výkonnosti programu pro sběr otisků webových prohlížečů uživatelů. Každý otisk webového prohlížeče byl vytvořen zkombinováním hodnot atributů, jejichž hodnoty jsou uloženy v jednotlivých záznamech.

Pro každý z 173 563 záznamů byl pomocí hashovací funkce md5 vytvořen otisk. Po dalším zpracování vytvořených otisků bylo zjištěno, že 148 735 (85,69 %) z nich je unikátních a 131 452 (75,73 %) dokonce distinktivních, tedy takové, které se vyskytly pouze jednou.

Graf 3 Frekvence výskytu jednotlivých otisků



Při analýze 50 otisků s nejvyšší frekvencí výskytu bylo zjištěno, že 28 z nich je tvořeno zařízeními Apple iPhone. Dalších 15 bylo vytvořeno prohlížečem Chrome, který běžel na populárních mobilních zařízeních jako je Xiaomi Redmi Note 7, Xiaomi Redmi 8 nebo Pro se systémem Android. Zbytek byl tvořen otisky prohlížeče Chrome na platformách Windows a Mac.

6 Závěr

Teoretická část na základě poznatků získaných z odborných materiálů a dalších zdrojů formuluje definici otisku webového prohlížeče. Stručně je zmíněna historie vzniku a použití souborů cookies a supercookies, které lze označit za předchůdce aktuálních metod jako je právě touto prací zkoumaný otisk webového prohlížeče.

Je popsán průběh získávání informací z prostředí prohlížečů a zařízení uživatelů, které následně slouží právě k vytváření otisku webového prohlížeče konkrétního uživatele. Jsou uvedeny způsoby využití takto vytvořených otisků zejména pro jednoznačnou identifikaci uživatele. Otisky jsou tvořeny kombinací shromážděných charakteristik, které jsou v této části práce popsány včetně způsobů jejich tvorby. Bylo zjištěno, že čím unikátnější je kombinace shromážděných charakteristik tvořící otisk, tím lepe lze identifikovat konkrétního uživatele.

Praktická část popisuje, jak jsou získané poznatky aplikovány do praxe. Nejprve je navržena architektura programového řešení skládajícího se ze tří částí: frontendová část, backendová část a databázová část, z nichž každá má nezastupitelnou úlohu. První frontendová zajišťuje sběr dat, druhá backendová, jejich zpracování a třetí databázová jejich záznam. Vše tři tyto části jsou důležité pro dosažení cíle této práce.

Následuje výběr zkoumaných atributů, u nichž bylo předpokládáno, že budou vhodné pro návrh programového řešení. Byly vybrány atributy zejména z objektů Navigator, Screen a Window. Po vybrání atributů bylo programové řešení podle návrhu architektury vytvořeno a nasazeno na webové stránky s reálnými uživateli za účelem otestování jeho efektivnosti.

Získaná data o otiscích webových prohlížečích a zařízeních uživatelů byla statisticky vyhodnocena, čímž se potvrdilo, že navrhované řešení je funkční a v praxi využitelné. Jednotliví uživatelé byli jednoznačně rozpoznáni v 75,73 % případů.

7 Seznam použitých zdrojů

[1] Kriminalistická daktyloskopie. In: *Policie České republiky* [online]. Policie České republiky [cit. 2023-01-05]. Dostupné z: https://www.policie.cz/clanek/kriminalisticka-daktyloskopie-266095.aspx

[2] What is Biometrics? How is it used in security?. In: *Kaspersky* [online]. Kaspersky [cit. 2023-01-05]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/biometrics

[3] Web Browser Market Share In 2022: 85+ Browser Usage Statistics. In: *Backlinko* [online]. Backlinko [cit. 2023-01-05]. Dostupné z: https://backlinko.com/browser-market-share

[4] BERNARDO, Vítor a Dulce DOMINGOS. Web-based Fingerprinting Techniques. In: *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications* [online]. SCITEPRESS - Science and Technology Publications, 2016, s. 271-282 [cit. 2023-01-05]. ISBN 978-989-758-196-0. ISSN 2184-2825. Dostupné z: doi:10.5220/0005965602710282

[5] ABOUT COVER YOUR TRACKS. In: *COVER YOUR TRACKS* [online]. COVER YOUR TRACKS [cit. 2023-01-05]. Dostupné z: https://coveryourtracks.eff.org/about

[6] Forensic Image Analysis. In: *Forensic's blog* [online]. Forensic's blog [cit. 2023-01-05]. Dostupné z: https://forensicfield.blog/forensic-image-analysis/

[7] ECKERSLEY, Peter. How Unique Is Your Web Browser?. In: ATALLAH, Mikhail J. a Nicholas J. HOPPER, ed., Mikhail ATALLAH, Nicholas HOPPER. *Privacy Enhancing Technologies* [online]. Vol 6205. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 1-18 [cit. 2023-01-05]. Lecture Notes in Computer Science. ISBN 978-3-642-14526-1. Dostupné z: doi:10.1007/978-3-642-14527-8_1

[8] UPATHILAKE, Randika, Yingkun LI a Ashraf MATRAWY. A classification of web browser fingerprinting techniques. In: *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)* [online]. Paris: IEEE, 2015, s. 1-5 [cit. 2023-03-15]. ISBN 978-1-4799-8784-9. ISSN 2157-4952. Dostupné z: doi:10.1109/NTMS.2015.7266460

[9] NIKIFORAKIS, N., A. KAPRAVELOS, W. JOOSEN, C. KRUEGEL, F. PIESSENS a G. VIGNA. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In: *2013 IEEE Symposium on Security and Privacy* [online]. Berkeley: IEEE, 2013, s. 541-555 [cit. 2023-01-05]. ISBN 978-0-7695-4977-4. ISSN 1081-6011. Dostupné z: doi:10.1109/SP.2013.43

[10] LAPERDRIX, Pierre, Walter RUDAMETKIN a Benoit BAUDRY. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In: *2016 IEEE Symposium on Security and Privacy (SP)* [online]. San Jose: IEEE, 2016, s. 878-894 [cit. 2023-01-05]. ISBN 978-1-5090-0824-7. ISSN 2375-1207. Dostupné z: doi:10.1109/SP.2016.57

[11] KAUR, Navpreet, Sami AZAM, Krishnan KANNOORPATTI, Kheng YEO a Bharanidharan SHANMUGAM. Browser Fingerprinting as user tracking technology. In: *2017 11th International Conference on Intelligent Systems and Control (ISCO)* [online]. IEEE, 2017, s. 103-111 [cit. 2023-03-15]. ISBN 978-1-5090-2717-0. Dostupné z: doi:10.1109/ISCO.2017.7855963

[12] HTTP. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP

[13] An overview of HTTP. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[14] GOTTFRIED, Jeremy. A complete guide to modern web applications. In: *Medium* [online]. Medium [cit. 2023-01-05]. Dostupné z: https://medium.com/jeremy-gottfrieds-tech-blog/a-complete-guide-to-modern-web-applications-793ae71b57ad

[15] HTTP State Management Mechanism. In: *HTTP State Management Mechanism* [online]. [cit. 2023-01-05]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc6265

[16] Using HTTP cookies. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

[17] BUJLOW, Tomasz, Valentin CARELA-ESPANOL, Beom-Ryeol LEE a Pere BARLET-ROS. A Survey on Web Tracking: Mechanisms, Implications, and Defenses. *Proceedings of the IEEE* [online]. 2017, **105**(8), 1476-1510 [cit. 2023-03-15]. ISSN 0018-9219. Dostupné z: doi:10.1109/JPROC.2016.2637878

[18] THE MAGIC COOKIE: HOW LOU MONTULLI CURED THE WEB'S AMNESIA. In: *Hidden Heroes* [online]. [cit. 2023-01-05]. Dostupné z: https://hiddenheroes.netguru.com/lou-montulli

[19] Distribution of Google segment revenues from 2017 to 2022. In: *Statista* [online]. Statista [cit. 2023-01-05]. Dostupné z: https://www.statista.com/statistics/1093781/distribution-of-googles-revenues-by-segment/

[20] How Does Facebook (Meta) Make Money?. In: *Investopedia* [online]. Investopedia [cit. 2023-01-05]. Dostupné z: https://www.investopedia.com/ask/answers/120114/how-does-facebook-fb-make-money.asp

[21] Digital Advertising - Worldwide. In: *Statista* [online]. Statista [cit. 2023-01-05]. Dostupné z: https://www.statista.com/outlook/dmo/digital-advertising/worldwide

[22] Jak dnes dělat úspěšnou reklamu. In: *Intuitivní marketing* [online]. [cit. 2023-01-05]. Dostupné z: https://intuitivnimarketing.cz/marketing/jak-v-dnesni-dobe-delat-uspesnou-reklamu/

[23] Most People Don't Finish Online Forms, Citing Security Concerns and Form Length. In: *Medium* [online]. The Manifest [cit. 2023-01-05]. Dostupné z: https://medium.com/@the_manifest/most-people-dont-finish-online-forms-citing-security-concerns-and-form-length-cf18ecf43644

[24] WILLS, Craig a Can TATAR. Understanding what they do with what they know. In: *Proceedings of the 2012 ACM workshop on Privacy in the electronic society* [online]. New York, NY, USA: ACM, 2012, s. 13-18 [cit. 2023-01-05]. ISBN 9781450316637. Dostupné z: doi:10.1145/2381966.2381969

[25] VASTEL, Antoine, Pierre LAPERDRIX, Walter RUDAMETKIN a Romain ROUVOY. FP-STALKER: Tracking Browser Fingerprint Evolutions. In: *2018 IEEE Symposium on Security and Privacy (SP)* [online]. IEEE, 2018, s. 728-741 [cit. 2023-03-15]. ISBN 978-1-5386-4353-2. Dostupné z: doi:10.1109/SP.2018.00008

[26] Evercookie. In: *Seon* [online]. [cit. 2023-01-05]. Dostupné z: https://seon.io/resources/dictionary/evercookie/#h-how-does-evercookie-work

[27] Device Fingerprinting: What Is It and How Exactly Does It Work?. In: *Seon* [online]. Seon [cit. 2023-01-05]. Dostupné z: https://seon.io/resources/device-fingerprinting/

[28] NAMPOINA ANDRIAMILANTO, Tompoariniaina. Leveraging browser fingerprinting for web authentication. In: *Leveraging browser fingerprinting for web authentication* [online]. Université Rennes 1, 2020 [cit. 2023-01-05]. Dostupné z: https://theses.hal.science/tel-03150590

[29] *Device Fingerprinting and Fraud Protection Whitepaper* [online]. Threat Metrix [cit. 2023-01-05]. Dostupné z: http://www.cse.iitd.ernet.in/~siy117527/sil765/readings/Device-Fingerprinting-and-Online-Fraud-Protection-Whitepaper.pdf

[30] Bot Detection: Identifying Bot Traffic with Open-source Browser Fingerprinting Techniques. In: *Fingerprint* [online]. [cit. 2023-01-05]. Dostupné z: https://fingerprint.com/blog/bot-detection/

[31] Browser Fingerprinting: Techniques, Use Cases & Best Practices. In: *AIMultiple* [online]. AIMultiple [cit. 2023-01-05]. Dostupné z: https://research.aimultiple.com/browser-fingerprinting

[32] JavaScript. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[33] Welcome to JavaScript and Web Browsers. In: *Learnhowtoprogram* [online]. [cit. 2023-01-05]. Dostupné z: https://www.learnhowtoprogram.com/introduction-to-programming/javascript-and-web-browsers/welcome-to-javascript-and-web-browsers

[34] Document Object Model (DOM). In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[35] CSS Object Model (CSSOM). In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/CSS_Object_Model

[36] Navigator. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator

[37] Screen. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Screen

[38] BODA, Károly, Ádám FÖLDES, Gábor GULYÁS a Sándor IMRE. User Tracking on the Web via Cross-Browser Fingerprinting. In: LAUD, Peeter, ed., Peeter LAUD. *Information Security Technology for Applications* [online]. Vol 7161. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 31-46 [cit. 2023-01-05]. Lecture Notes in Computer Science. ISBN 978-3-642-29614-7. Dostupné z: doi:10.1007/978-3-642-29615-4_4

[39] TAKEI, Naoki, Takamichi SAITO, Ko TAKASU a Tomotaka YAMADA. Web Browser Fingerprinting Using Only Cascading Style Sheets. In: *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)* [online]. IEEE, 2015, s. 57-63 [cit. 2023-03-15]. ISBN 978-1-4673-8315-8. Dostupné z: doi:10.1109/BWCCA.2015.105

[40] Browser as a rendering engine. In: *Collegenote* [online]. [cit. 2023-01-05]. Dostupné z: https://www.collegenote.net/curriculum/internet-technology-csit/44/234/

[41] Extensions 101. In: *Extensions 101* [online]. Google [cit. 2023-01-05]. Dostupné z: https://developer.chrome.com/docs/extensions/mv3/getstarted/extensions-101/

[42] Browser Extensions. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

[43] Extension Fingerprints. In: *Extension Fingerprints* [online]. Github [cit. 2023-01-05]. Dostupné z: https://github.com/z0ccc/extension-fingerprints

[44] <canvas>: The Graphics Canvas element. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas

[45] WebGL: 2D and 3D graphics for the web. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

[46] MOWERY, Keaton a Hovav SHACHAM. Pixel Perfect: Fingerprinting Canvas in HTML5. In: *Pixel Perfect: Fingerprinting Canvas in HTML5* [online]. University of California, San Diego [cit. 2023-01-05]. Dostupné z: https://hovav.net/ucsd/dist/canvas.pdf

[47] ACAR, Gunes, Christian EUBANK, Steven ENGLEHARDT, Marc JUAREZ, Arvind NARAYANAN a Claudia DIAZ. The Web Never Forgets. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* [online]. New York, NY, USA: ACM, 2014, s. 674-689 [cit. 2023-03-15]. ISBN 9781450329576. Dostupné z: doi:10.1145/2660267.2660347

[48] What is an IP address?. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-03-15]. Dostupné z: https://www.mozilla.org/en-US/products/vpn/more/what-is-an-ip-address/

[49] The difference between a VPN and a web proxy. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://www.mozilla.org/en-US/products/vpn/more/vpn-or-proxy/

[50] AL-FANNAH, Nasser Mohammed. One leak will sink a ship: WebRTC IP address leaks. In: *2017 International Carnahan Conference on Security Technology (ICCST)* [online]. IEEE, 2017, s. 1-5 [cit. 2023-03-15]. ISBN 978-1-5386-1585-0. Dostupné z: doi:10.1109/CCST.2017.8167801

[51] JAKOBSSON, Christer. *Peer-to-peer communication in web browsers using WebRTC* [online]. In: . [cit. 2023-01-05]. Dostupné z: http://www.diva-portal.org/smash/get/diva2:852726/FULLTEXT01.pdf

[52] 11 Most In-Demand Programming Languages. In: *Berkeley Extension* [online]. [cit. 2023-01-05]. Dostupné z: https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/

[53] Node.js. In: *Geeksforgeeks* [online]. [cit. 2023-01-05]. Dostupné z: https://www.geeksforgeeks.org/nodejs/

[54] Express/Node introduction. In: *MDN Web Docs* [online]. Mozilla Foundation [cit. 2023-01-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[55] AWS Elastic Beanstalk. In: *AWS* [online]. [cit. 2023-01-05]. Dostupné z: https://aws.amazon.com/elasticbeanstalk/

[56] *Why Use MongoDB and When to Use It?* [online]. In: . [cit. 2023-01-05]. Dostupné z: https://www.mongodb.com/why-use-mongodb

8 Seznam tabulek a grafů

8.1 Seznam tabulek

8.2 Seznam grafů

Přílohy

Příloha 1 - priloha.zip