elektrorevue
ISSN 1213 - 1539

Rok / Year:	Svazek / Volume:	Číslo / Number:	Jazyk / Language
2017	19	1	CZ

Identifikace webových prohlížečů s využitím metody Web Browser Fingerprinting

Identification of Web Browsers by using Web Browser Fingerprinting Method

Martin Štůsek, Pavel Mašek

martinstusek@phd.feec.vutbr.cz, masekpavel@feec.vutbr.cz

Fakulta elektrotechniky a komunikačních technologií VUT v Brně

DOI: -

Abstract: This article describes methods allowing identification of end stations based on information accessible through web browser. Information are analyzed without need to store any data on remote host computer. Identification method consists of creating of web browser fingerprint through data accessible via application programming interface (API) or headers of communications protocols. Accuracy of web browser identifications is more than 94 % regardless of the used operating system or web browser. In this article, most frequently used methods of web browser fingerprinting (JavaScript, Flash, WebGL) are described. Further, we detail protection mechanisms against web browser fingerprinting.

elektrorevue
ISSN 1213 - 1539

Identifikace webových prohlížečů s využitím metody Web Browser Fingerprinting

Martin Štůsek, Pavel Mašek

Fakulta elektrotechniky a komunikačních technologií VUT v Brně
Email: martinstusek@phd.feec.vutbr.cz, masekpavel@feec.vutbr.cz

Abstrakt *– Tento článek pojednává o možnostech identifikace koncových stanic na základě informací dostupných skrze webový prohlížeč, bez nutnosti ukládání specifických dat na hostitelském počítači. Metoda spočívá ve vytvoření otisku webového prohlížeče (Web Browser Fingerprinting) skrze data dostupná v rámci programátorského rozhraní či hlavičky komunikačních protokolů. Přesnost této metody se pohybuje nad hranicí 94 % a její úspěšnost není závislá na použitém operačním systému či webovém prohlížeči. V článku jsou popsány metody používané pro získání otisku prohlížeče, dále jsou uvedeny možnosti ochrany před tímto způsobem identifikace webových prohlížečů.*

1 Úvod

S rostoucím počtem uživatelů připojených do sítě Internet (3,5 miliardy v roce 2016) se prudce zvyšuje význam webových prohlížečů, skrze které je interpretována velká část internetového obsahu [1]. V posledních letech se díky technologiím jako HTML5 (HyperText Markup Language 5), JavaScript, Flash a CSS (Cascading Style Sheets) staly ze statických textových webových stránek dynamické komplexní aplikace. Tyto stránky pak mohou z důvodu přizpůsobení obsahu vyžadovat dodatečné informace ze strany klienta jako například rozlišení obrazovky, verzi webového prohlížeče či operačního systému. Webové prohlížeče proto implementují programátorská rozhraní poskytující tyto informace webovým serverům. Tato data pak nemusí sloužit pouze pro potřeby správné funkčnosti stránek, ale mohou být využity provozovatelem stránek k identifikaci uživatele bez nutnosti uložení unikátních dat pro identifikaci na straně klienta. Zmíněná technika se nazývá otisk webového prohlížeče (Web Browser Fingerprinting) a je využívána v hojné míře všemi velkými internetovými společnostmi jako jsou Microsoft, Facebook či Google [2]. Těmto společnostem pak získaná data slouží pro doručení cílené reklamy, na které stojí velká část zisků zmíněných společností. Díky technologii otisku webového prohlížeče mohou poskytovatelé webových služeb sledovat chování uživatelů i mimo vlastní webové stránky, například díky vysokému rozšíření modulů pro sociální sítě mohou tyto firmy získávat data ze všech webových stránek, kde jsou jejich moduly použity. Technika otisku webového prohlížeče je funkční i v anonymních režimech webového prohlížeče, ve kterém se do hostitelského počítače neukládají žádná data [3].

Otisk webového prohlížeče však může sloužit i pro zvýšení zabezpečení webových stránek, technologii lze využít zejména u internetového bankovnictví. I v případě, že útočník získá přihlašovací údaje uživatele, pokud nebude otisk webového prohlížeče souhlasit, může mu být odmítnut přístup, či bude vyžadováno dodatečné ověření [3].

V následujících sekcích budou popsány metody používané při vytváření otisku prohlížeče (sekce 2), dále bude popsáno několik studií zabývajících se otiskem webových prohlížečů (sekce 3). V poslední části pak budou popsány možnosti ochrany před technikou otisku webového prohlížeče (sekce 4).

2 Metody získávání parametrů pro vytvoření otisku webového prohlížeče

Pro vytváření otisku parametrů dostupných skrze programátorské rozhraní webového prohlížeče se využívá zejména technologie JavaScript, která je součástí všech moderních prohlížečů. V případě, že koncová stanice nepovoluje či neumožňuje spuštění interpreta JavaScript kódu, lze využít i technologii CSS, která slouží primárně pro definici vzhledu webové stránky [4], [5].

Dalším často využívaným prvkem pro generování otisku webového prohlížeče jsou parametry hlavičky aplikačního protokolu HTTP (Hypertext Transfer Protocol).

S příchodem technologie HTML5 se začíná postupně prosazovat generování otisku elementu `<canvas>` v kombinaci s technologií WebGL (Web Graphics Library), která skrze JavaScript pracuje přímo s grafickou kartou počítače. Tato metoda má proto přístup k podrobným informacím o grafické kartě počítače i použitých ovladačích, které tvoří unikátní otisk [6]. Poslední možností, která však v současné době ztrácí na významu, je získání otisku pomocí technologie Flash od společnosti Adobe [7].

V počátcích identifikace webových prohlížečů byly využívány také technologie Java či Silverlight, tyto techniky získávání otisku však již nejsou aktuální, jelikož většina webových prohlížečů z bezpečnostních důvodů neumožňuje použití modulů využívající rozhraní NPAPI (Netscape Plugin Application Programming Interface).

V současné době se proto nejčastěji používají techniky:

- Získání parametrů webového prohlížeče z JavaScript objektů.
- Získání seznamu dostupných písem s využitím:
  - JavaScript,
  - CSS.

elektrorevue
ISSN 1213 - 1539

- Získání parametrů HTTP hlaviček odesílaných při inicializaci spojení se serverem.
- Extrakce obrazových dat z elementu `<canvas>`.
- Získání otisku webového prohlížeče skrze technologii Flash.

2.1 JavaScript objekty

Jak již bylo zmíněno v úvodní sekci 1, nejpoužívanější technikou získání unikátních informací o webovém prohlížeči a koncové stanici je použití JavaScript objektů. Tyto informace může nést například parametr rozlišení obrazovky. Pro tento účel využívá JavaScript objekt `screen`, který obsahuje kromě rozlišení obrazovky i další parametry nesoucí informaci o oblasti pro vykreslení obsahu [8]. Seznam všech dostupných parametrů je uveden v tabulce 1.

Tabulka 1: Parametry objektu Screen.

Parametr	Popis
availHeight	Dostupná výška (bez panelu)
availWidth	Dostupná šířka (bez panelu)
colorDepth	Bitová hloubka
height	Celková výška obrazovky
pixelDepth	Počet bitů na pixel
width	Celková šířka obrazovky

Informace o běhovém prostředí koncové platformy nese objekt `navigator`, ten obsahuje informace například o použitém webovém prohlížeči či jeho verzi [8]. Seznam nejdůležitějších parametrů je dostupný v tabulce 2. Je nutné zmínit, že objekt `navigator` není definován v žádném standardu, a proto se může seznam položek v jednotlivých webových prohlížečích lišit, nebo jej nemusí obsahovat vůbec. Avšak všechny majoritní webové prohlížeče tento objekt obsahují [9].

Z pohledu unikátnosti dat použitelných pro vytvoření otisku webového prohlížeče jsou nejdůležitějšími parametry atributy `mimeType` a `plugins` objektu `navigator`.

Položka `mimeType` definuje seznam podporovaných internetových médií MIME (Multipurpose Internet Mail Extensions). Podporovaný datový typ `mimeType` je definován jako položka pole obsažená v objektu `navigator` a obsahuje informace o instalovaném modulu, popis datového typu a prefix používaný v HTTP hlavičce definující obsah přenášených dat v těle zprávy [8].

Atribut `plugins` obsahuje seznam nainstalovaných zásuvných modulů, kdy každý záznam obsahuje popis zásuvného modulu, jméno zásuvného modulu a cestu k souboru se zásuvným modulem.

2.2 Detekce písem

Jakýkoliv vytvořený HTML (HyperText Markup Language) dokument se skládá z HTML elementů. Každý

Tabulka 2: Parametry objektu Navigator.

Parametr	Popis
appCodeName	Vrací kódové označení prohlížeče
appName	Vrací název webového prohlížeče
appVersion	Vrací informace o verzi webového prohlížeče
cookieEnabled	Nese informaci zda jsou povoleny soubory cookies
geolocation	Vrací polohu uživatele
language	Vrací jazyk webového prohlížeče
onLine	Nese informaci zda je webový prohlížeč online
platform	Vrací název platformy pro kterou je prohlížeč zkompilován
product	Vrací název jádra prohlížeče
userAgent	Vrací hodnotu identifikující prohlížeč v hlavičce HTTP odesílanou v poli `user-agent`
plugins	Vrací seznam nainstalovaných zásuvných modulů
mimeTypes	Vrací seznam podporovaných datových typů

element dokumentu obsahuje atributy umožňující změnu vzhledu, jeden z těchto atributů nese název `font-family` a slouží pro definici použitého písma [4]. Do toho atributu lze uložit více souborů písem, největší prioritu má font, který je uveden na první pozici. Pokud však není písmo v systému nalezeno je zvoleno písmo z druhé pozice, v případě že není nalezeno ani druhé písmo v pořadí pokračuje až po poslední položku. Jestliže nebyl nalezen v systému žádný z uvedených fontů, je použito výchozí písmo webového prohlížeče [4].

Styl každého znaku je odlišný pro každé z použitých písem, to znamená, že šířka a výška textu v různých stylech písem budou také odlišné [10]. Z toho plyne i odlišná výška a šířka elementů obsahující text. Velikost elementu `<div>` při použití rozdílných stylů písem je uvedena v tabulce 3. Díky těmto odlišnostem ve velikosti elementů při použití různých stylů lze získat seznam písem nainstalovaných na klientské stanici. Tento seznam dostupných písem lze dále použít jako jednu z částí pro vytvoření otisku webového prohlížeče.

Pro vytvoření tohoto seznamu se využívá technologie JavaScript viz kapitola 2.2.1. Pokud je však tato technologie na straně koncové stanice nedostupná, lze v tomto případě využít technologie CSS viz kapitola 2.2.2 určené primárně pro definici vzhledu.

elektrorevue
ISSN 1213 - 1539

2.2.1 JavaScript detekce písem

Detekce písem s využitím technologie JavaScript je založena na vykreslení řetězce s použitím několika různých písem. Na element, který obsahuje řetězec se všemi znaky anglické abecedy (například „The quick brown fox jumps over the lazy dog“), jsou postupně aplikovány styly písem definované ve stylu elementu.

Tabulka 3: Velikost elementu div při použití rozdílných písem.

Název písma	Šířka	Výška
Sans	519 px	84 px
Arial	452 px	83 px
Calibri	416 px	83 px

Měřící skript elementu nejprve nastaví styl písma, který není v systému s vysokou pravděpodobností dostupný například font s názvem „nové-písmo“. V tomto případě použije webový prohlížeč výchozí styl písma. Dalším krokem je postupné procházení seznamu s definovanými styly písma, v každém kroku je spočítán rozdíl mezi šířkou a výškou výchozího písma a nyní používaného stylu. Pokud je rozdíl nulový, je jisté, že písmo není v systému nainstalováno [8].

2.2.2 CSS detekce písem

Pro získání seznamu nainstalovaných písem lze od verze 3 využít CSS souborů primárně definující vzhled [8]. Verze CSS3 přinesla podporu definic písem stažených z externích zdrojů, tzv. webové fonty. Pro definici těchto stylů slouží atribut `@font-face`.

Jak lze vidět, na ukázce Listing 1 je u každého stylu definován jeho název pomocí atributu `font-family`. Nejdůležitějším parametrem je však zdrojová adresa stylu písma, která je uvozena atributem `src`. V případě, že je písmo nainstalováno v systému, je použit lokální soubor písem uvozený slovem `local`. Pokud systém písmo neobsahuje, je stažen z adresy uvedené v atributu `url`. Tato adresa však může obsahovat adresu obsahující měřící skript, který vyhodnocuje data z přijatých dotazů. V příkladu je uvedena adresa `database.php`, uveden je však i atribut ve kterém je definován název požadovaného písma [8]. Skript na straně serveru si do databáze ukládá všechny přijaté dotazy a může si tak z uložených informací vytvořit seznam nainstalovaných písem na straně klienta.

Listing 1: Definice stylu písma v CSS

```
@font-face{
font-family: 'font1';
src: local('Arial'), url("database.php?fontname=
    Arial"); }
div#font1{ font-family: 'font1'; }
@font-face{
font-family: 'font2';
src: local('Century'), url("database.php?fontname
    =Century"); }
div#font2{ font-family: 'font2'; }
```

2.3 HTTP hlavičky

Komunikace mezi webovým serverem a koncovým zařízením probíhá téměř výhradně skrze protokol HTTP. Při sestavování spojení jsou v hlavičce protokolu zasílána na server data o klientské stanici. Tyto informace slouží serveru k přizpůsobení obsahu odesílaného na koncovou stanici. Lze je však využít i k vytvoření otisku webového prohlížeče.

Protokol HTTP definuje několik desítek možných parametrů přenášených v hlavičce [11], avšak pro potřeby generování otisku webového prohlížeče je vhodných pouze několik parametrů [12]. Seznam těchto parametrů je uveden v tabulce 4.

Tabulka 4: Parametry HTTP hlavičky.

Parametr	Popis
User-Agent	Nese informace o typu aplikace, operačním systému, výrobci software a jeho verzi
Accept	Informuje server o možnostech webového prohlížeče z hlediska typu internetového média (MIME)
Content-Encoding	Podporované metody komprese obsahu na straně koncové stanice
Content-Language	Preferovaný jazyk webového prohlížeče použitý pro odpověď

2.4 Element canvas

Element `<canvas>` je novým elementem definovaným standardem HTML5. Tento element poskytuje oblast, na kterou lze programově vykreslovat rasterizované grafické objekty. V současné době je podporován ve všech majoritních webových prohlížečích i operačních systémech [7].

Základní přístup vykreslování grafiky je založen na principu grafického kontextu (graphic context), který poskytuje programátorské rozhraní API (Application Programming Interface), skrze nějž lze volat metody obsluhující vykreslování obsahu. Ve specifikaci HTML5 je definován pouze 2D grafický kontext, umožňující vykreslování čar, polygonů, ale také Beziérových křivek či barevných gradientů [6].

2.4.1 Vykreslení textu

Pro účely vytvoření otisku webového prohlížeče se využívá zejména vlastnost elementu `canvas`, která umožňuje vykreslování textu přímo do elementu bez nutnosti využívat externí knihovny. Velikost a styl písma textu je možné jednoduše měnit pomocí atributů podobných CSS.

elektrorevue
ISSN 1213 - 1539

2.4.2 Extrakce obrazových dat

Pro účely pozdějšího zkoumání je nutné obrazová data z elementu `<canvas>` vyexportovat, k tomuto účelu slouží metoda `getImageData()`, která vrací objekt typu `ImageData`. Obsahem tohoto objektu je matice celých čísel ve formátu RGBA (Red Green Blue Alpha) interpretující jednotlivé pixely obrazu [6].

Element `<canvas>` dále poskytuje skrze programátorské rozhraní metodu `toDataURL(type)`, která obsah obrazových dat zakóduje jako URL (Uniform Resource Locator) adresu nesoucí všechna data původního obrazu kódovaného pomocí metody Base64 [6]. Takto zakódovaná data jsou pak převedena na heš, který je odeslán na server. Tyto heše poté slouží pro identifikaci webového prohlížeče.

Metoda je založena na poznatku, kdy jsou obrazová data vykreslena na různém hardware odlišně. Díky tomu je možné od sebe rozeznat jednotlivé stanice na základě jejich hardwarové konfigurace. Rozdíly mezi vykreslováním jsou patrné i mezi jednotlivými verzemi ovladačů [6]. Avšak při použití stejné hardwarové konfigurace se stejnou verzí ovladačů je nemožné od sebe stanice odlišit.

2.4.3 WebGL

S problematikou otisku elementu `<canvas>` úzce souvisí i technologie WebGL (Web Graphics Library). Tato technologie tvoří prostředníka mezi webovým prohlížečem a grafickým adaptérem koncové stanice. To umožňuje přistupovat webovému prohlížeči k prostředkům grafického adaptéru skrze programátorské rozhraní dostupné v jazyce JavaScript. Výsledná obrazová data jsou pak vykreslena do elementu `<canvas>`. WebGL podporuje všechny základní operace dostupné u elementu `<canvas>`, jako jsou vykreslení přímek, křivek ale také vyhlazování hran (Anti-aliasing). Vykreslení dat na grafické kartě obstarává část kódu nazvaná `Vertex shader`, syntaxe tohoto jazyka je podobná jazyku `C`. Tento kód je po přeložení zpracováván přímo grafickou kartou a spolupracuje s další částí kódu nazvanou `Fragment shader`, který zajišťuje rasterizaci scény (převod objektů do podoby matic pixelů) [7].

Výsledná data jsou vykreslena do elementu `<canvas>`, ze kterého lze podle postupu uvedeného v předchozí sekci 2.4.2 získat otisk webového prohlížeče. Stejně jako v případě vykreslení obrazových dat pomocí procesoru počítače skrze element `<canvas>` provádí každý grafický adaptér výpočty specifickým způsobem. Z tohoto důvodu se výstupní obrazová data na různých grafických adaptérech liší dle použitého hardware, což umožňuje vytvoření unikátního otisku webového prohlížeče.

Pro identifikaci lze použít i parametry nesoucí informace o použitém grafickém adaptéru, jedním z těchto parametrů je `WebGL Renderer`, který může obsahovat velmi podrobné informace o grafickém adaptéru a jeho verzi. Pro účely identifikace lze použít i jméno výrobce adaptéru `WebGL Vendor`. Avšak některé webové prohlížeče z bezpečnostních důvodů tyto informace neposkytují, nebo je nahrazují falešnými údaji [12].

2.5 Technologie Flash

I když je technologie Flash v moderních webových prohlížečích na ústupu a v mobilních systémech již není podporována na žádné platformě, má stále nezanedbatelný podíl na trhu. Vytvoření otisku webového prohlížeče na základě Flash pracuje na stejných principech jako předchozí metody.

Technologie Flash poskytuje bohaté programátorské rozhraní, které obsahuje množství informací o koncové stanici. Skrze programátorské rozhraní je možné získat seznam všech nainstalovaných písem, je implementována i vlastní metoda pro získání verze operačního systému, na kterém webový prohlížeč běží. Toto rozhraní navíc oproti výchozímu, dostupnému skrze JavaScript, poskytuje mnohem více informací. Verze sytému je zde popsána s mnohem podrobnějšími informacemi, například v systému Linux jsou zobrazeny přesné informace o verzi jádra systému [13]. Z těchto informací je pak snadné vytvořit otisk s vysokou entropií.

Technologie Flash [7] je však již na ústupu a v průběhu několika málo let bude odstraněna její podpora ze všech majoritních webových prohlížečů.

3 Studie zabývající se otiskem webových prohlížečů

Identifikací webových prohlížečů na základě jejich otisků se zabývá několik studií, v tomto článku budou porovnány tři z nich. Jedna z prvních studií zabývající se otisky webových prohlížečů byla provedena společností Electronic Frontier Foundation provozující službu Panopticlick [14], dalším zástupcem je služba AmIUnique [15] vyvýjená společnostmi Diversify European Project a INSA Rennes school. Posledním představitelem je služba Uniquemachine [16], která je provozována výzkumným týmem univerzity Lehigh.

Všechny zmíněné služby využívají kombinaci metod vytvoření otisku webového prohlížeče popsaných v sekci 2. Na obrázku 1 jsou zobrazeny výsledky studií provedených zmíněnými společnostmi. Jsou obsaženy pouze výsledky dat, které jsou získávány ve všech službách. Získaná data jsou interpretována ve formátu normalizované entropie, která byla získána dle vztahu:

$$NH = \frac{H(X)}{H_M} = \frac{-\sum_i P(x_i) log_2 P(x_i)}{log_2 N}. \quad (1)$$

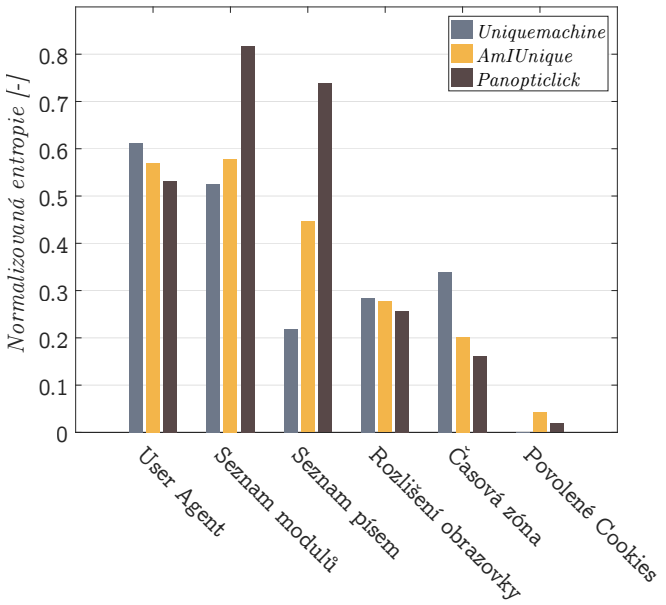
$H(X)$ značí entropii, kde X je proměná obsahující možné hodnoty $x_1, x_2, ..., x_i$ a $P(X)$ značí pravděpodobnost výskytu hodnoty. H_M značí maximální entropii, případ kdy jsou pravděpodobnosti výskytu všech prvků stejné. N značí počet získaných záznamů (otisků webového prohlížeče).

Jak lze vidět z nejnovější studie Uniquemachine [7], význam seznamu písem, který v nejstarší studii (Panopticlick) vykazoval hodnotu entropie 0,738 klesl na hodnotu 0,219. Tento prudký pokles je způsoben snižováním podpory technologie Flash v moderních webových prohlížečích [7]. Lze také vidět, že význam informace o stavu

elektrorevue
ISSN 1213 - 1539

povolení souborů Cookies ztrácí na významu a v poslední studii je hodnota entropie téměř nulová.

Na obrázku 1 lze také pozorovat že u nejstarší ze služeb, Panopticlick, je entropie seznamu zásuvných modulů vysoká, avšak u novějších studií její význam klesá. To je způsobeno odstraněním podpory zásuvných modulů skrze NPAPI, které má za následek snížení počtu nainstalovaných modulů ale i jejich rozmanitosti.



Obrázek 1: Porovnání studií otisků prohlížeče.

Úspěšnost identifikace webového prohlížeče pomocí jeho otisku má v případě služby AmIUnique hodnotu 90.84%, v případě nejnovější metody otisku webového prohlížeče poskytovaného službou Uniquemachine je však přesnost 99.24%, což je 8,4% nárůst oproti službě AmIUnique. Tento rozdíl je způsoben použitím nových přístupů pro generováni otisku webového prohlížeče. Studie UniqueMachine využívá pro zvýšení přesnosti výsledků zejména technologie WebGL, díky tomu umožňuje tato služba s úspěšností 83,24% identifikovat koncové stanice i při použití několika různých webových prohlížečů [7].

3.1 Metody služby Uniquemachine

Studie UniqueMachine obsahuje nejnovější poznatky z oblasti identifikace webových prohlížečů, využívá však také poznatků dřívějších studií a pro identifikaci webového prohlížeče používá metody zmíněné v sekci 2.

Pro zvýšení přesnosti identifikace však přidává několik nových technik, díky nimž je možná identifikace stanic i napříč webovými prohlížeči. Níže je uvedeno několik metod používaných službou Uniquemachine pro zpřesnění výsledků [7].

- *Rozlišení obrazovky a poměr stran:* V rámci identifikace jednoho webového prohlížeče je metoda rozlišení obrazovky přesná, avšak v případě více prohlížečů se stává nestabilní, jelikož uživatelé obsah přibližují. U některých webových prohlížečů se v tomto případě mění rozlišení obrazovky, proto je využito i poměru stran který se nemění.
- *Anti-aliasing:* Porovnává, jak je prováděno vyhlazování hran na jednotlivých zařízeních, protože použitý algoritmus vykreslování se liší v závislosti na konfiguraci software i hardware zařízení. Metoda není stabilní pro identifikaci napříč webovými prohlížeči, jelikož vyhlazování nemusí být podporováno ve všech webových prohlížečích.
- *Vykreslování přímek a křivek:* Je porovnáváno vykreslování jednoduchých 2D objektů na různých grafických adaptérech.
- *Kamera:* Porovnává se interpretace 3D objektů ve 2D rovině.
- *Světla a stíny:* Porovnává, jak grafická karta vykresluje zdroje světla, odrazy a stíny.
- *Ořez rovin:* Měří zpracovávání objektů, kterými prochází ořezová rovina pohledu.
- *AudioContext:* Vrací základní informace o zvukovém zařízení, tedy vzorkovací frekvenci počet kanálů a druh reproduktorů. Zdrojové kódy však obsahují doposud nepoužitou část, která umožňuje generování siglálu, ze kterého je poté vytvořen heš.
- *Počet jader procesoru:* Parametr prohlížeče přístupný skrze JavaScript objekt `hardwareConcurrency`, který obsahuje informaci o počtu logických procesorových jednotek.

4 Ochrana před identifikací webového prohlížeče

Bezstavová podstata otisku webového prohlížeče činí detekci či zabránění ve sledování parametrů velmi obtížnými. Existuje však několik metod, které mohou identifikaci na základě webového prohlížeče ztížit, či dokonce znemožnit.

- Jednou z možných ochran před identifikací otiskem webového prohlížeče je standardizace programátorských rozhraní napříč všemi webovými prohlížeči. V současné době všechny majoritní webové prohlížeče poskytují metody programátorského rozhraní dle standardu, navíc však implementují metody specifické pro daný webový prohlížeč. Díky těmto metodám lze rozpoznat jednotlivé prohlížeče, což zvyšuje pravděpodobnost úspěšné identifikace pomocí otisku webového prohlížeče. Při použití standardizovaného programátorského rozhraní není identifikace jednotlivých webových prohlížečů možná [13].
- Použití jednotného seznamu nainstalovaných písem, které webový prohlížeč poskytuje skrze programátorská rozhraní. Význam tohoto kroku je stejný jako v případě prvního bodu [13].

elektrorevue
ISSN 1213 - 1539

- Nejjednodušším způsobem znesnadnění identifikace webového prohlížeče je zakázání technologie JavaScript na koncovém zařízení. Díky tomuto kroku je znemožněna identifikace téměř pomocí všech dostupných metod včetně WebGL, které je úzce provázáno s jazykem JavaScript. Toto řešení však nese i svá úskalí. Moderní webové stránky obsahují velké množství JavaScript kódu, který zvyšuje uživatelský komfort při procházení webových stránek. V krajním případě může dojít i ke ztrátě funkčnosti stránek, například pokud je obsah stránky dynamicky generován pomocí JavaScriptu [13].

- Webový prohlížeč při inicializaci spojení se serverem odesílá v hlavičce HTTP velké množství parametrů popisující koncovou stanici. Snížením počtu těchto parametrů jen na nutné minimum (v případě HTTP je vyžadována pouze metoda a adresa serveru) by došlo k výraznému znesnadnění sledování webového prohlížeče [13].

- Seznam zásuvných modulů poskytovaných webovým prohlížečem skrze programátorské rozhraní obsahuje velké množství informací o nainstalovaných modulech. Jsou obsaženy podrobné informace o verzi zásuvného modulu, tato informace má poměrně vysokou entropii a dá se s výhodou využít pro identifikaci webového prohlížeče. Snížením množství informací poskytovaným skrze toto rozhraní by došlo k znatelnému ztížení identifikace [13].

- Otisk webového prohlížeče na základě elementu `<canvas>` je umožněn díky přístupnosti obrazových dat skrze kontext elementu. V případě zakázání této metody ztratí sledovací služba možnost vytvoření otisku a identifikace na základě elementu `<canvas>` pak není možná [13].

- Pro uživatele nejsnadnější metodou zabránění identifikace webového prohlížeče na základě otisku je využití rozšíření prohlížeče. Takovýchto zásuvných modulů existuje velké množství. Nejpokročilejším doplňkem je rozšíření Ghostery [17], které uživatele informuje o skriptech provádějících sledování prohlížeče. Dalším rozšířením je například AdBlock Plus [18], který slouží pro blokování reklam, ale umožňuje i blokování modulů sociálních sítí sledujících uživatele.

5 Závěr

Tento článek se zabýval možností identifikace webových prohlížečů na základě jejich otisků. V první části byly popsány nejčastěji používané metody získání otisků webového prohlížeče, na kterých jsou založeny téměř všechny služby zabývající se identifikací webových prohlížečů na základě jejich otisku. Dále byly popsány tři realizované studie zabývající se touto problematikou. Důraz byl kladen především na službu Uniquemachine, která využívá nejnovější poznatky z oblasti identifikace na základě otisku webového prohlížeče. Tato služba kromě metod používaných konkurenčními společnostmi využívá pro identifikaci zejména technologii WebGL. Díky těmto rozšířením byla zvýšena úspěšnost identifikace na více než 99% v rámci jednoho prohlížeče, respektive 83% v případě identifikace několika webových prohlížečů.

Problematika identifikace webových prohlížečů na základě jejich otisků je v současné době velmi aktuální, jelikož všechny nadnárodní internetové společnosti využívají svou implementaci identifikace. Účinná ochrana před takovýmto sledováním je velmi obtížná, avšak existuje několik možností ochrany. Některé z nich byly v tomto článku popsány.

V současné době služby nabízejí pouze identifikaci webového prohlížeče, avšak výzkumný tým zodpovědný za službu Uniquemachine pracuje na implementaci identifikace jednotlivých koncových stanic. Díky stále se zvyšující se komplexnosti dnešních prohlížečů, které skrze programátorská rozhraní zpřístupňují stále více informací o koncových stanicí, bude v budoucnu identifikace stále přesnější. Jednou z posledních funkcí, která může o uživateli prozradit více informací, je implementace přímé kontroly Bluetooth rozhraní skrze prohlížeč Google Chrome. Pokud sledovací server dokáže zjistit adresu adaptéru skrze rozhraní webového prohlížeče, bude mít tato informace velmi vysokou entropii, jelikož adresa zařízení je s velkou pravděpodobností unikátní.

Literatura

[1] *Internet Live Stats: Internet Users* [online]. 2017 [cit. 2017-02-01]. Dostupné z: http://www.internetlivestats.com/internet-users/

[2] FIORE, Ugo, Aniello CASTIGLIONE, Alfredo De SANTIS a Francesco PALMIERI. Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome. *2014 17th International Conference on Network-Based Information Systems*. IEEE, 2014, , 355-360. DOI: 10.1109/NBiS.2014.102. ISBN 978-1-4799-4224-4. Dostupné z: http://ieeexplore.ieee.org/document/7023976/

[3] NIKIFORAKIS, Nick, Alexandros KAPRAVELOS, Wouter JOOSEN, Christopher KRUEGEL, Frank PIESSENS a Giovanni VIGNA. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, 541-555. DOI: 10.1109/SP.2013.43. ISBN 978-0-7695-4977-4. Dostupné z: http://ieeexplore.ieee.org/document/6547132/

[4] TAKEI, Naoki, Takamichi SAITO, Ko TAKASU a Tomotaka YAMADA. Web Browser Fingerprinting Using Only Cascading Style Sheets. *2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*. IEEE,

elektrorevue
ISSN 1213 - 1539

2015, 57-63. DOI: 10.1109/BWCCA.2015.105. ISBN 978-1-4673-8315-8. Dostupné z: http://ieeexplore.ieee.org/document/7424801/

[5] SAITO, Takamichi, Kazushi TAKAHASHI, Koki YASUDA, Takayuki ISHIKAWA, Ko TAKASU, Tomotaka YAMADA, Naoki TAKEI a Rio HOSOI. OS and Application Identification by Installed Fonts. *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2016, 684-689. DOI: 10.1109/AINA.2016.55. ISBN 978-1-5090-1858-1. Dostupné z: http://ieeexplore.ieee.org/document/7474155/

[6] MOWERY, Keaton a Hovav SHACHAM. Pixel Perfect: Fingerprinting Canvas in HTML5. *Computer Science and Engineering*. 2012.

[7] CAO, Yinzhi, Song LI a Erik WIJMANS. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. *Internet Society*. 2017, **21**.

[8] KHADEMI, Amin Faiz, Mohammad ZULKERNINE a Komminist WELDEMARIAM. *An Empirical Evaluation of Web-Based Fingerprinting*. DOI: 10.1109/MS.2015.77. ISBN 10.1109/MS.2015.77. Dostupné také z: http://ieeexplore.ieee.org/document/7106402/

[9] *W3schools.com: The Navigator Object* [online]. [cit. 2017-02-01]. Dostupné z: http://www.w3schools.com/jsref/obj_navigator.asp

[10] NIKIFORAKIS, Nick, Alexandros KAPRAVELOS, Wouter JOOSEN, Christopher KRUEGEL, Frank PIESSENS a Giovanni VIGNA. On the Workings and Current Practices of Web-Based Device Fingerprinting. *IEEE Security*. 2014, **12**(3), 28-36. DOI: 10.1109/MSP.2013.160. ISSN 1540-7993. Dostupné z: http://ieeexplore.ieee.org/document/6679044/

[11] FIELDING, Roy a Jim GETTYS. *Hypertext Transfer Protocol – HTTP/1.1* [online]. United States, 1999 [cit. 2017-02-01]. Dostupné z: https://tools.ietf.org/html/rfc2616

[12] LAPERDRIX, Pierre, Walter RUDAMETKIN a Benoit BAUDRY. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, 878-894. DOI: 10.1109/SP.2016.57. ISBN 978-1-5090-0824-7. Dostupné z: http://ieeexplore.ieee.org/document/7546540/

[13] UPATHILAKE, Randika, Yingkun LI a Ashraf MATRAWY. A classification of web browser fingerprinting techniques. *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2015, 1-5. DOI: 10.1109/NTMS.2015.7266460. ISBN 978-1-4799-8784-9. Dostupné z: http://ieeexplore.ieee.org/document/7266460/

[14] *Panopticlick: Is your browser safe against tracking?* [online]. [cit. 2017-02-01]. Dostupné z: https://panopticlick.eff.org/

[15] *Am I Unique?* [online]. [cit. 2017-02-01]. Dostupné z: https://amiunique.org/

[16] *Uniquemachine* [online]. [cit. 2017-02-01]. Dostupné z: http://www.uniquemachine.org/

[17] *Ghostery* [online]. [cit. 2017-02-01]. Dostupné z: https://www.ghostery.com/

[18] *Adblock Plus: Surf the web without annoying ads!* [online]. [cit. 2017-02-01]. Dostupné z: https://adblockplus.org/