WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI

PRACA DYPLOMOWA

STACJONARNE STUDIA I°

Temat pracy: **METODY CYFROWEGO ODCISKU PALCA PRZEGLĄDAREK INTERNETOWYCH I URZĄDZEŃ PODŁĄCZONYCH DO INTERNETU—WYZWANIA I ROZWIĄZANIA**

INFORMATYKA W MEDYCYNIE

(kierunek studiów)

INFORMATYCZNE SYSTEMY ZARZĄDZANIA W MEDYCYNIE

(specjalność)

Dyplomant: **Artur WOLFF**

Promotor pracy: **dr inż. Rafał KASPRZYK**

Warszawa 2021

Oświadczenie

Wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum WAT.

Dnia .....

Pracę przyjąłem

promotor pracy
dr inż. Rafał Kasprzyk

Spis treści

1. Wprowadzenie do fingerprintingu **7**

1.1. Podstawowe pojęcia 7

1.1.1. Nomenklatura używana w tej pracy 7

1.1.2. Definicje 8

1.1.3. Właściwości fingerprintu 9

1.2. Fingerprinting a Internet 10

1.2.1. Początki Internetu 10

1.2.2. Założenia funkcjonowania Internetu i ich realizacja 11

1.2.3. Początki śledzenia użytkowników Internetu 12

1.3. Fingerprinting w branży komputerowej 15

1.3.1. Fingerprinting audio, wideo i technologia ACR 15

1.3.2. Fingerprinting klucza publicznego 16

2. Problematyka prywatności i anonimowości w Internecie **17**

2.1. Prywatność i anonimowość 17

2.1.1. Łączenie danych 18

2.1.2. Czy prywatność jest nam potrzebna? 19

2.2. Metody identyfikacji użytkowników 19

2.3. Zagrożenia związane z fingerprintingiem 21

2.4. Możliwości ochrony przed fingerprintingiem 21

2.4.1. Perspektywa użytkownika 22

2.4.2. Perspektywa twórców oprogramowania 22

2.4.3. Czy da się skutecznie zapobiec fingerprintingowi? 23

2.5. Inne zastosowania fingerprintingu 24
2.5.1. Potencjał w operacjach wojskowych w cyberprzestrzeni 24

3. Metody fingerprintingu urządzeń i przeglądarek 25
3.1. Pojęcie pasywnego i aktywnego fingerprintingu 25
3.2. Źródła danych identyfikacyjnych 26
3.3. Fingerprinting implementacji stosu TCP/IP 26
3.3.1. Opis algorytmu 27
3.4. Fingerprinting przeglądarek 28
3.4.1. Canvas fingerprinting 30
3.4.2. Web Audio fingerprinting 32
3.4.3. Pozostałe techniki 34

4. Eksperymentalny algorytm klasyfikatora fingerprintów 37
4.1. Motywacja 37
4.2. Opis algorytmu 37
4.2.1. Odległość Levenshteina 39
4.3. Pseudokod 40
4.4. Ocena złożoności czasowej i pamięciowej 44
4.5. Ocena efektywności 46
4.5.1. Opis utworzonego rozwiązania 46
4.5.2. Testy efektywności 46

5. Podsumowanie 50

Wstęp

Wstęp

Rozdział 1.

Wprowadzenie do fingerprintingu

1.1. Podstawowe pojęcia

1.1.1. Nomenklatura używana w tej pracy

Pisząc o odcisku palca, użyto (także w tytule pracy) ogólnie przyjętego skrótu myślowego oznaczającego odbitkę linii papilarnych, czyli formę językową uznawaną za poprawną przez specjalistów od daktyloskopii.

Użycie formy językowej „odcisk palca” w terminie „cyfrowy odcisk palca” ma wiele sensu. Jeszcze bez zdefiniowania tego specjalistycznego terminu możemy domyślić się, co oznacza. Oczywiście wynika to z faktu, że cyfrowy odcisk palca i analogowy odcisk palca są ze sobą w pewien sposób powiązane (koncepcja cyfrowego odcisku palca czerpie z wartości wynikających ze stosowania odbitek ludzkich linii papilarnych w dziedzinie kryminalistyki).

Angielskie słowo „fingerprint” tłumaczy się jako odcisk palca, jednakże w zagranicznych publikacjach dotyczących cyfrowego odcisku palca rzadko występuje termin „digital fingerprint”. Jak piszą Flood i Karlsson [10, s. 4], kontekst użycia jest na tyle wyraźny, że użycie samego „fingerprint” jest wyczerpujące.

Zachodnie nazewnictwo ma tę przewagę, że jest zdecydowanie bardziej kompaktowe. Także w przypadku słowotwórczego zabiegu *fingerprinting* oznaczającego czynność; szukając polskiego odpowiednika, musielibyśmy sięgnąć po „cyfrowe znakowanie”. Z uwagi na tę kompaktowość i łatwość użycia w pracy preferowane

będzie użycie oryginalnej nomenklatury.

1.1.2. Definicje

W kolejnych punktach zawarto najważniejsze definicje i powiązane pojęcia analogiczne do tych obecnych w literaturze [7, 10], które będą używane w przeciągu całej pracy.

Fingerprint

Wektor cech pozwalający zidentyfikować dowolny zbiór danych.

Aby *fingerprint* pełnił praktyczną funkcję identyfikacyjną, tak jak odbitka ludzkich linii papilarnych pełni praktyczną funkcję identyfikacyjną, często stosuje się algorytm, który kojarzy wektor cech z określonej długości (zwykle krótkim) ciągiem bajtów (identyfikatorem; można go także rozumieć jako etykieta). Takim algorytmem może być na przykład wysokiej wydajności funkcja skrótu (niekoniecznie zdatna do zastosowań kryptograficznych—na przykład MurmurHash). W niektórych źródłach można także spotkać się z taką definicją, że *fingerprint* to już sam wynik wyżej wspomnianego algorytmu [26, s. 123–132]. Taka definicja nie zmienia istoty *fingerprintu*, ale jest zdecydowanie mniej przydatna w kontekście *fingerprintingu* urządzeń podłączonych do Internetu i przeglądarek internetowych.

Fingerprint urządzenia podłączonego do Internetu

Wektor cech pozwalający zidentyfikować urządzenie podłączone do Internetu.

Instalacja przeglądarki internetowej

Instalacja na konkretnym urządzeniu. W przypadku zmiany ustawień, konfiguracji i liczby *pluginów*/rozszerzeń oraz aktualizacji przeglądarki instalacja przeglądarki pozostaje ciągle tą samą instalacją.

Fingerprint przeglądarki internetowej

Wektor cech pozwalający zidentyfikować instalację przeglądarki internetowej.

1.1.3. Właściwości fingerprintu

Ludzkie linie papilarne są na ogół niepowtarzalne, niezmienne i nieusuwalne. Z wartości wynikających ze stosowania ich odbitek w swojej dziedzinie badawczej czerpie (także etymologicznie) koncepcja *fingerprintu* i dlatego też *fingerprint* z dobrze dobranymi cechami będzie odzwierciedlać podobne właściwości.

W przypadku *fingerprintu* urządzeń podłączonych do Internetu i przeglądarek internetowych najważniejszymi jego właściwościami są unikalność / różnorodność (niepowtarzalność) oraz stabilność (niezmienność), przy czym zwiększenie unikalności lub stabilności ma najczęściej negatywny wpływ na drugi parametr [7, s. 11].

Jedną ze stosowanych metod pomiaru unikalności *fingerprintu* urządzeń i przeglądarek jest entropia Shannona [7, s. 6].

Entropia Shannona

Wartość entropii można rozumieć jako liczbę pytań binarnych potrzebnych do sklasyfikowania losowo wybranego elementu z danego zbioru. Zatem entropia Shannona zbioru D z etykietami $\{l_0, l_1, \ldots, l_{n-1}\}$ wyraża się wzorem

$$H(D) = -\sum_{i=0}^{n-1} p(l_i) \log_2 p(l_i)$$

gdzie $p(l_i)$ to wyrażona ułamkiem częstość $x \in D$ mającego etykietę l_i. W przypadku, w którym każda etykieta występuje tak samo często, entropia ma wartość maksymalną równą $\log_2 n$.

Przykład Jeśli zbiór *fingerprintów* przeglądarek internetowych posiada 32 bity entropii, to w przypadku losowego wyboru jednego z nich oczekujemy, że w najlepszym przypadku tylko 1 na 4294967295 przeglądarek będzie miała taki sam *fingerprint*.

Stabilność

W przypadku dodania kolejnej cechy do wektora cech identyfikującego urządzenie lub przeglądarkę zwykle zwiększy to entropię, ale także zmniejszy stabilność *fingerprintu*. Dzieje się tak, ponieważ jest to kolejna rzecz, która może zmienić się w czasie. Jeśli jedną z cech wejściowych jest wersja oprogramowania urządzenia lub wersja przeglądarki (która zwykle zmienia się parę razy w ciągu roku) to kolejne *fingerprinty* mogą odbiegać od siebie i naiwny klasyfikator korzystający z algorytmu reagującego na najmniejsze zmiany (na przykład funkcja skrótu) mógłby nadać takiemu urządzeniu/przeglądarce kolejną etykietę zamiast potraktowania jej jako poprzednio widzianą instalację.

1.2. Fingerprinting a Internet

Aby lepiej zrozumieć istotę *fingerprintu* i motywację stojącą za stosowaniem *fingerprintingu* w kontekście urządzeń podłączonych do Internetu oraz przeglądarek internetowych wspominając o różnych innych obszarach przetwarzania komputerowego, w których wykorzystywany jest *fingerprinting* w stosownych mu celach, kolejne punkty posłużą jako referencja (także historyczna).

1.2.1. Początki Internetu

Początek Internetu, jaki znamy obecnie to początek stworzonej w 1969 roku na potrzeby amerykańskiego wojska sieci ARPAnet. ARPAnet była implementacją niezależnych prac Paula Barana, Donalda Daviesa i Leonarda Kleinrocka z lat 60. XX wieku. Na samym początku swojego istnienia Internet wykorzystywany był do tego, aby rozpraszać obliczenia pomiędzy wiele komputerów—w tym wypadku chodziło o superkomputery znajdujące się w innych ośrodkach badawczych (ARPAnet powstało na Uniwersytecie Kalifornijskim w Los Angeles) [21]. W tym samym okresie powstawały inne globalne sieci komputerowe zapoczątkowane zwykle w innym celu (na przykład komunikacyjnym, rozrywkowym), które później połączono z ARPAnet.

Badacze historii Internetu wskazują na fakt, iż gwałtowny rozwój Internetu zawdzięcza się właśnie komunikacyjnemu i rozrywkowemu aspektowi konkurencyjnych sieci [15].

1.2.2. Założenia funkcjonowania Internetu i ich realizacja

Po tym, jak w 1989 Tim Berners-Lee oraz Robert Cailliau utworzyli projekt sieci dokumentów hipertekstowych, czyli tego, co obecnie znamy jako World Wide Web i strony internetowe, osoby prywatne oraz instytucje komercyjne zaczęły dostrzegać korzyści z użytkowania Internetu, a szczególnie z wykorzystania go jako medium reklamy i sprzedaży [21]. Zniesienie zakazu wykorzystywania Internetu do celów zarobkowych w 1991 roku zakończyło chwilę, w której Internet był medium naukowego dyskursu i zapoczątkowało okres istnienia Internetu dla mas, który trwa do dziś.

Perspektywa techniczna

Podstawą struktury obecnego Internetu jest model TCP/IP i koncepcyjnie składa się ze współpracujących ze sobą 4 warstw [13]:

1. dostępu do sieci
2. kontroli transportu
3. Internetu
4. aplikacji

W najwyższej z warstw, czyli warstwie aplikacji działają takie usługi jak przeglądarka czy serwer WWW. To najbardziej interesująca warstwa z perspektywy niniejszej pracy, ale *fingerprinting* urządzeń podłączonych do Internetu może odbywać się także w niższych warstwach.

1.2.3. Początki śledzenia użytkowników Internetu

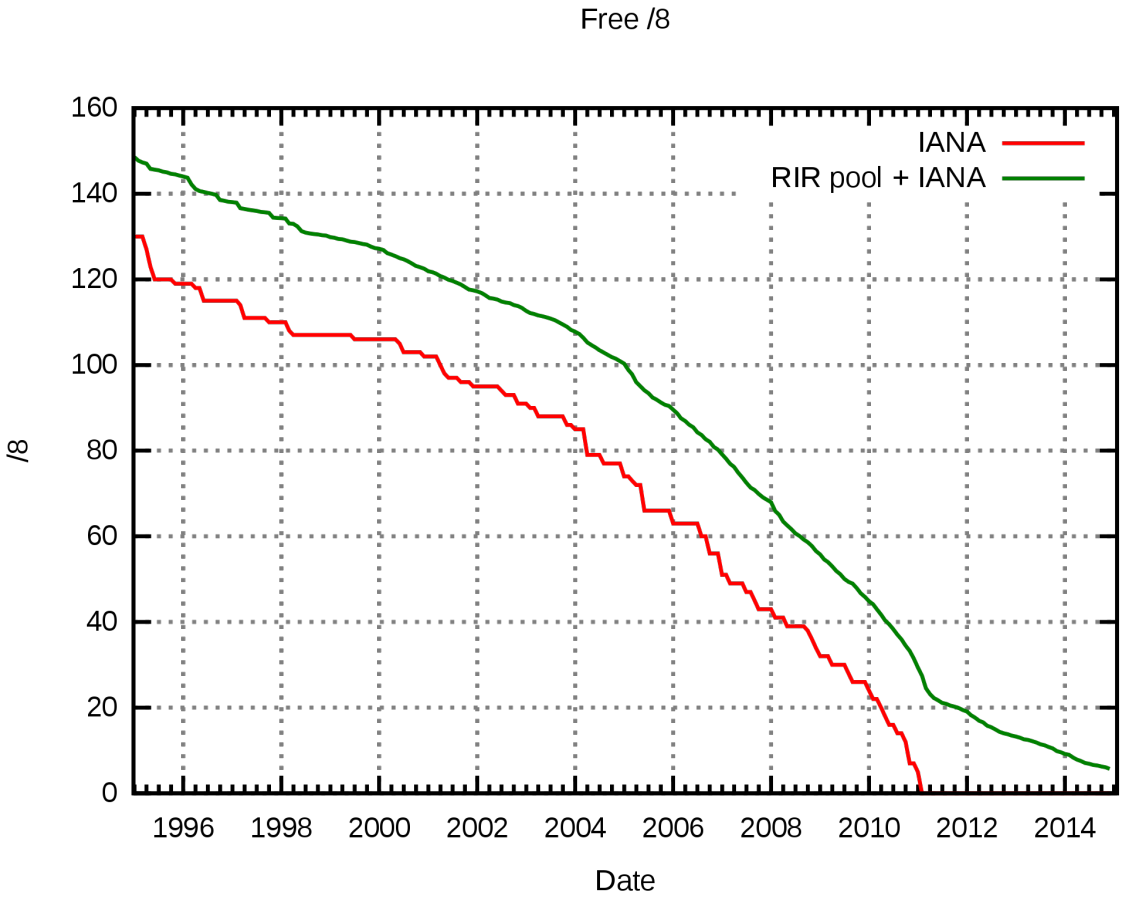
Gwałtowny rozwój komercyjnego Internetu sprawił, że firmy zajmujące się reklamą i sprzedażą w Internecie zaczęły także dostrzegać korzyści płynące z identyfikacji i śledzenia użytkowników Internetu. W szczególności zaczęto analizować aktywność i zachowanie użytkowników. Oprócz instytucji komercyjnych identyfikacją i śledzeniem użytkowników zainteresowane są instytucje rządowe, co dobitnie pokazał wyciek poufnych, tajnych i ściśle tajnych dokumentów NSA w 2013 roku. Metody identyfikacji, a zarazem śledzenia użytkowników zmieniały się w czasie wraz z rozwojem Internetu.

Adres IPv4

Adres IPv4 na początku istnienia Internetu był swego rodzaju globalnym identyfikatorem, dzięki któremu można było unikatowo identyfikować użytkowników Internetu. Adres IPv4 to 32-bitowy identyfikator. Prosta estymacja pozwala nam zauważyć, że adresów IP w wersji czwartej jest około $4,3$ miliarda[1]. Internet dzisiaj to wielomiliardowa społeczność, a liczba urządzeń podłączonych do Internetu zdecydowanie przewyższa wyżej wymienioną estymację. Już w 1992 roku zauważono, że w najbliższym czasie pula adresów IPv4 zostanie wyczerpana [11]. W następnych latach proponowano kolejne rozwiązania (takie jak na przykład NAT [8]), które implementowali dostawcy usług internetowych, pozwalając na łączenie się wielu urządzeń za pośrednictwem jednego, publicznego adresu IPv4. Wyczerpywanie się kolejnych pul adresów pokazuje Rys. 1.

Biorąc pod uwagę powyższe, na mocy Zasady Szufladkowej Dirichleta możemy stwierdzić, że adres IPv4 nie jest już identyfikatorem, który mógłby unikatowo identyfikować każde urządzenie podłączone do Internetu. W tym momencie warto także zaznaczyć, że o ile nowy standard IPv6 pozwalałby na taką identyfikację, to został on zaprojektowany z myślą o prywatności i posiada szereg rozszerzeń, które w przyszłości (kiedy Internet w pełni przejdzie na adresację w wersji szóstej) mają zapobiegać precyzyjnej identyfikacji [18].

[1] W rzeczywistości liczba dostępnych adresów jest niższa: https://stackoverflow.com/a/2437185



Źródło: https://upload.wikimedia.org/wikipedia/commons/c/cf/Ipv4-exhaust.svg

Rys. 1. Wolne pule adresów IPv4 w czasie

Cookies

Małe porcje informacji zapisywane na urządzeniu użytkownika w obszarze pamięci trwałej przeglądarki po interakcji ze stroną internetową, która je zapisuje. Powstały głównie ze względu na potrzebę poprawienia doświadczeń użytkowników ze stronami internetowymi tak, aby zapamiętywać pewien stan o znaczeniu dla danej sesji dla danego użytkownika (na przykład stan koszyka w sklepie internetowym).

Cookies dzielą się na tak zwane *first-party cookies* i *third-party cookies*. O ile pierwsze z wymienionego podziału faktycznie desygnowane są do tego, aby spełniać wymienioną funkcję, to *third-party cookies* mogą być nadawane przez (na przykład) skrypty reklamowe umieszczone na serwującej je stronie, dzięki czemu użytkownik może być śledzony w kontekście całej sieci reklamowej. Ubogie mechanizmy kontroli *cookies* w przeglądarkach internetowych i obawy związane z naruszaniem prywatności użytkowników przez śledzenie wykorzystujące *cookies* doprowadziły do powstania dyrektywy Unii Europejskiej dotyczącej obowiązku informacyjnego, która zawiera m.in. obowiązek informowania o polityce stosowania *cookies*. Doprowadziło to wcześniej także do powstania rozszerzeń w przeglądarkach, takich jak nagłówek Do Not Track i Tryb Prywatny, które w domyśle miały pomóc częściowo rozwiązać wspomniany problem.

Niektóre przeglądarki internetowe, takie jak Apple Safari (Intelligent Tracking Prevention silnika WebKit) lub Mozilla Firefox, wykorzystują obecnie natywne mechanizmy inteligentnego blokowania *third-party cookies*. Powstało także wiele rozszerzeń do przeglądarek, które pozwalają blokować niechciane *cookies*.

Inne metody identyfikacji użytkowników

Fingerprinting urządzeń podłączonych do Internetu i przeglądarek internetowych to jedna ze zbioru wymyślnych technik, które zaczęto stosować ze względu na ułomność lub postępujące ograniczenia metod wykorzystujących na przykład adresy IP urządzeń, lub *cookies*. *Fingerprinting* przeglądarek po raz pierwszy opisał Eckersley jako technikę, która pozwala serwerom WWW jednoznacznie zidentyfikować urządzenie użytkownika za pomocą informacji wysyłanych przez przeglądarkę in-

ternetową, wtedy kiedy te informacje są unikalne dla większości z nich, tworząc ich *fingerprint* [7]. Relację urządzeń i przeglądarek pomiędzy ich użytkownikami opisuje 2.2.

1.3. Fingerprinting w branży komputerowej

Fingerprinting to technika wykorzystywana w wielu obszarach w dyscyplinie informatyki. *Fingerprinting* urządzeń podłączonych do Internetu i przeglądarek internetowych to tylko pewien wycinek zastosowań tej koncepcji. Definicje związane z zastosowaniami *fingerprintu* innych bytów mogą być bardziej specyficzne lub mogą eksponować inne, charakterystyczne właściwości. Kolejne punkty posłużą jako referencja, do ukazania jak szeroko wykorzystywana jest omawiana koncepcja.

1.3.1. Fingerprinting audio, wideo i technologia ACR

Metody *fingerprintingu* akustycznego i cyfrowych materiałów wideo znane także jako technologia Automatic Content Recognition (ACR) zostały zaprezentowane w 2012 podczas Consumer Electronics Show, pokazując, że urządzenia dostępne dla zwykłego konsumenta mogą być na tyle sprytne, by wyszukiwać informacje na podstawie kontekstu [19]. Technologia ACR sparametryzowana jest w podobny sposób co algorytmy *fingerprintingu* urządzeń i przeglądarek, czyli istotny jest balans pomiędzy niepowtarzalnością a stabilnością *fingerprintu*. Klasyfikacja *fingerprintu* audio lub wideo musi działać w podobny sposób do zachowania ludzkiego moderatora, czyli w przypadku kiedy materiał jest nieodróżnialny dla ludzkiego ucha, lub oka jako ten, wobec którego przeprowadzany jest proces rozpoznawania, to powinien on zostać oflagowany. Proces wykorzystywany przez technologię Automatic Content Recognition określany jest mianem *perceptual hashing*.

1.3.2. Fingerprinting klucza publicznego

W kryptografii klucza publicznego w celach autoryzacji klucza publicznego pozyskanego w niezaufany sposób (na przykład ściągając go ze strony internetowej) poprzez zaufany kanał wymiany informacji (zwykle rozmowa telefoniczna), który nie pozwala na autoryzację całego klucza w efektywny sposób, stosuje się jego skrót nazywany „fingerprintem klucza publicznego". Wynik zastosowania odpowiedniej funkcji skrótu na kluczu publicznym jest na tyle kompaktowy, że pozwala na efektywną manualną autoryzację, czyli ręcznie przez człowieka.

W celu ułatwienia wymiany *fingerprintów* kluczy publicznych poprzez kanały głosowe powstała lista słów PGP, która analogicznie do alfabetu fonetycznego NATO asocjuje każdą kolejną porcję bitów *fingerprintu* klucza z odpowiednim słowem w języku angielskim.

Rozdział 2.

Problematyka prywatności i anonimowości w Internecie

Technika, o której traktuje ta praca, jest ważnym tematem głównie dlatego, że wykorzystanie jej do identyfikacji użytkowników ma istotne implikacje w obszarze prywatności i anonimowości.

2.1. Prywatność i anonimowość

Prywatność internetowa to pewien podzbiór relacji pomiędzy gromadzeniem i rozpowszechnianiem danych, technologią, społecznym oczekiwaniom wobec prywatności i prawnymi oraz politycznym problemami orbitującymi wokół tych zagadnień [16]. W celach orientacyjnych stosowanym uproszczeniem jest pogląd, że prywatność internetowa to możliwość zatrzymania związanych z własną aktywnością danych dla siebie. Anonimowością jest zatem sytuacja, w której przekazywanie takich danych jest zablokowane.

Prywatność i anonimowość użytkowników Internetu stała się zagadnieniem jeszcze przed nadejściem ery Internetu [5], pozwalając na koniec lat dziewięćdziesiątych na zaognienie dyskusji, która trwa do dzisiaj. Użytkownicy Internetu mają różne oczekiwania wobec poziomu ich prywatności w sieci. Mniej wyczuleni na punkcie prywatności użytkownicy są w stanie pójść na pewny kompromis pomiędzy

wykorzystaniem ich danych a oferowanym na tej podstawie potencjalnym usprawnieniem ich doświadczeń w Internecie (na przykład na konkretnych stronach internetowych, w kontekście sieci reklamowych lub w innych szerszych, kontekstach). Akceptują oni ryzyko zbyt szczegółowego profilowania, potencjalnych naruszeń prywatności i inwigilacji. Inni użytkownicy dążą (mniej lub bardziej) do utrzymania anonimowości takiej, jaka panowała w Internecie na początku jego istnienia [23, s. 54–69].

2.1.1. Łączenie danych

Istnieją firmy specjalizujące się w pozyskiwaniu, kupowaniu i przetwarzaniu danych użytkowników w różnych celach (zwykle reklamowych). Nie wszystkie firmy rynku danych przetwarzają dane w sposób naruszający prywatność użytkowników, ale techniki takie jak łączenie danych z różnych źródeł (bez uprzedniej anonimizacji) mogą lub będą prowadzić do nadużyć.

Bardzo sławny przypadek firmy Cambridge Analytica, która tworzyła profile psychologiczne użytkowników i używała ich do manipulacji opinią publiczną za pomocą mediów społecznościowych to jeden z przykładów firmy, która łącząc dane, poważnie naruszyła prywatność użytkowników (m.in. Facebooka).

Istnieje wiele różnych dróg, dzięki którym w Internecie i w świecie rzeczywistym użytkownicy będą nieświadomie profilowani lub inwigilowani, a łączenie danych z różnych źródeł istotnie poprawi dokładność obrazu użytkownika. Jedną z takich dróg jest używanie stron internetowych będących częścią większej sieci reklamowej, która łączy dane na przykład z portali *social media*, wyników wyszukiwań, wysyłanych formularzy, a nawet z takich źródeł jak systemów automatycznego wykrywania twarzy w sklepach stacjonarnych. Niektóre informacje, które można wnioskować po automatycznym przetworzeniu to orientacja seksualna, poglądy polityczne i religijne, rasa, historia użycia substancji psychoaktywnych, estymowany iloraz inteligencji czy osobowość [14].

2.1.2. Czy prywatność jest nam potrzebna?

Ochrona prywatności ma swoich przeciwników i zwolenników. „nie obchodzi mnie prywatność, bo nie mam nic do ukrycia” to argument przytaczany przez przeciwników ochrony prywatności. Jedną z ważnych osób, które opowiedziały się niegdyś za taką argumentacją, jest Eric Schmidt, były CEO firmy Google. Istnieje silna polaryzacja pomiędzy przeciwnikami i zwolennikami ochrony prywatności. W swojej książce autobiograficznej Edward Snowden (słynny amerykański *whistleblower*) stwierdził, że „oświadczyć, że nie obchodzi cię prywatność, bo nie masz nic do ukrycia, to mniej więcej to samo, co oświadczyć, że nie obchodzi cię wolność słowa, ponieważ nie masz nic do powiedzenia” [23]. Bruce Schneier, amerykański kryptograf i ekspert w dziedzinie bezpieczeństwa komputerowego podsumowuje, że zbyt wiele osób myśli o tym argumencie jak o wyborze pomiędzy bezpieczeństwem a prywatnością. „Prawdziwym wyborem jest wybór pomiędzy wolnością a kontrolą” [22]. Prawo do prywatności zawiera się w Powszechnej Deklaracji Praw Człowieka [2].

2.2. Metody identyfikacji użytkowników

Tak ja zauważono wcześniej w niniejszej pracy—wraz z komercjalizacją Internetu powstał i ewoluował szereg różnych metod identyfikacji urządzeń, przeglądarek i tym samym użytkowników. Istnieją różne zastosowania identyfikacji, ale jednym z najbardziej powszechnie omawianych (i kontrowersyjnych) jest śledzenie użytkowników (formalnie: łączenie ze sobą wielu wizyt jednego użytkownika na tej samej platformie) [9, s. 3].

Warto także zauważyć, że *fingerprinting* przeglądarek internetowych to jedna z dróg identyfikacji urządzeń podłączonych do Internetu. Jest ona jednak na tyle reprezentatywna, że często mówi się o *fingerprintingu* urządzeń jako *fingerprintingu* przeglądarek. Dzieje się tak, ponieważ dzisiejsze przeglądarki internetowe podczas interakcji z serwerami WWW mogą aktywnie lub pasywnie przekazywać zestaw danych na tyle szeroki [7], że zawiera on w sobie *fingerprint* urządzenia, na którym

działa przeglądarka. Przeważający ogrom aktywności użytkowników wokół „przeglądania" Internetu i wiele danych „oferowanych" przez przeglądarki internetowe naturalnie sprawia, że wysiłki identyfikujące użytkowników rozważane są głównie w kontekście tychże. Niniejsza praca stara się zapewnić pewien (choć często niewidoczny) podział. W kolejnym rozdziale niniejszej pracy omówiony jest także *fingerprinting* urządzeń podłączonych do Internetu, kiedy niemożliwe jest wykorzystanie do celów identyfikacyjnych przeglądarki internetowej użytkownika.

Identyfikacja użytkowników nie zawsze jest także synonimem z identyfikacją urządzeń czy przeglądarek, ale odsetek przypadków, kiedy w dzisiejszym świecie więcej niż jedna osoba korzysta z np. jednej przeglądarki internetowej, wydaje się stosunkowo niski. Powodem dla takiej estymacji jest fakt, iż wspomniane przeglądarki, które posiadają istotny ułamek udziałów w rynku przeglądarek internetowych, posiadają mechanizmy pozwalające na ich jednoznaczną personalizację (parowanie z personalnym kontem Google w przeglądarce Google Chrome, integracja przeglądarki Safari z ekosystemem firmy Apple, parowanie z personalnym kontem Firefox w przeglądarce Firefox itd.). Co więcej, jeden użytkownik może korzystać z wielu przeglądarek, co utrudnia jednoznaczną identyfikację, ale nie czyni jej niemożliwą. Istnieją bowiem metody *fingerprintingu* urządzeń i przeglądarek pozwalające na identyfikację użytkowników pomiędzy przeglądarkami internetowymi zainstalowanymi na tym samym urządzeniu.

Gdyby podsumować metody identyfikacji użytkowników w dzisiejszym Internecie to ich niedługa lista prezentowałaby się następująco:

- Użycie *cookies* (w szczególności *third-party cookies*);
- Użycie *supercookies*;
- Użycie *fingerprintingu*.

Oczywiście nic nie stoi na przeszkodzie, aby każda z tych metod była używana w połączeniu z innymi.

2.3. Zagrożenia związane z fingerprintingiem

Al-Fannah i Mitchell [9, s. 1] konstatują, że identyfikacja za pomocą *fingerprintingu* jest znacznie trwalsza niż ta bazująca na *cookies*, praktycznie bez możliwości kontroli przez użytkowników i nietrywialna do wykrycia. Istnieją zatem przynajmniej trzy powody, dla których *fingerprinting* stanowi istotnie większe zagrożenie dla prywatności użytkowników (większe niż wszystkie dotychczasowe):

- Na tę chwilę nie są znane proste sposoby, aby z całkowitą pewnością wykryć, że dana platforma lub strona internetowa używa *fingerprintingu*;
- Użytkownicy mogą kontrolować moc śledzenia za pomocą *cookies*, regularnie usuwając je lub całkowicie blokując. Jak zostało nadmienione wcześniej, istnieją także regulacje prawne dotyczące użycia *cookies*. Nie ma jednak żadnych porównywalnych, równie prostych do użycia technik kontroli *fingerprintingu*;
- *Fingerprinting* (w przeciwieństwie do *cookies*) nie polega na jednej, konkretnej właściwości HTTP. *Fingerprinting* bazuje na wielu technologiach, aby zbierać różne informacje o właściwościach i konfiguracji przeglądarki lub systemu operacyjnego. Każda z tych informacji to szansa, aby zastosować *fingerprinting*.

Co więcej, *fingerprinting* może być użyty do tworzenia *supercookies*, czyli specjalnych *cookies*, które po usunięciu mogą zostać ponownie utworzone, jeśli ta sama przeglądarka zostanie wykryta przez użycie *fingerprintingu* [9, s. 2].

2.4. Możliwości ochrony przed fingerprintingiem

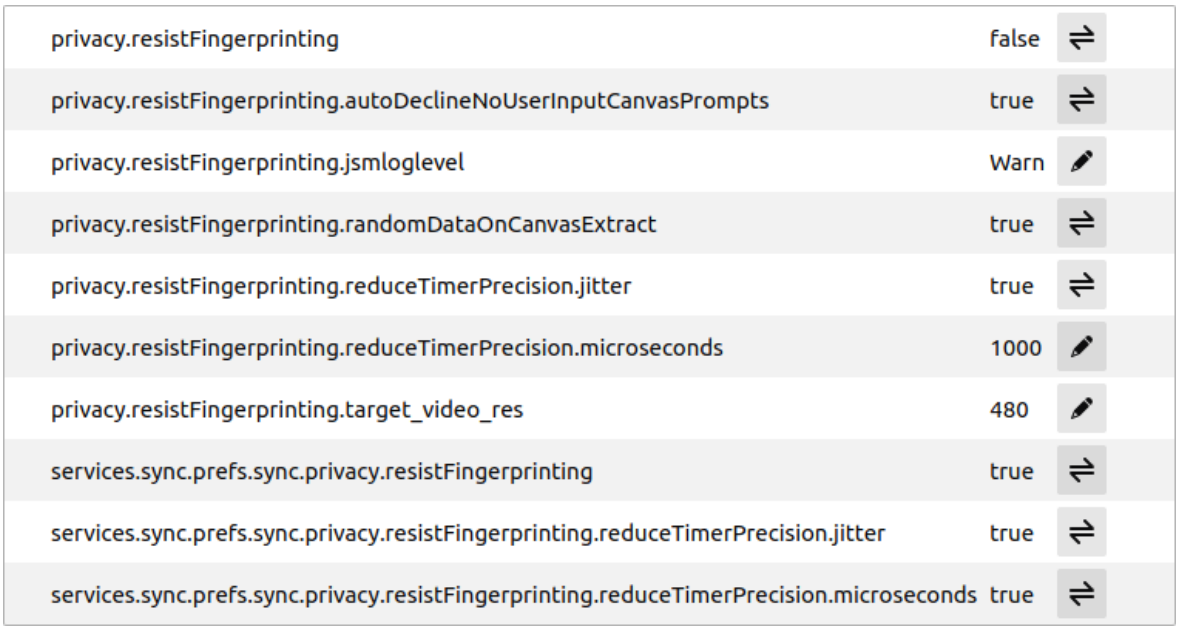
Autorzy monografii dotyczącej metod kontroli i ochrony przed *fingerprintingiem* wyróżniają dwa typy rozwiązań ochrony przed *fingerprintingiem*: możliwe do zastosowania przez użytkowników i takie, które powinny zostać zaimplementowane przez autorów oprogramowania [9].

2.4.1. Perspektywa użytkownika

Kontrola i ochrona, którą mogą zastosować użytkownicy to wybór przeglądarki i jej konfiguracja (np. Firefox posiada pewne ustawienia dotyczące ograniczania *fingerprintingu*—Rys. 2) lub instalacja dodatkowych rozszerzeń, które blokują lub ograniczają *fingerprinting*, blokując JavaScript, fałszując atrybuty lub blokując przesyłanie ich zawartości. Niestety każdy z wymienionych sposobów wiąże się obecnie z istotną degradacją jakości przeglądania Internetu. Przy wyborze aktualnie prawdopodobnie najbardziej skupionej na prywatności przeglądarki Tor możemy spotkać się z wieloma ostrzeżeniami. Jest to na przykład prośba o niemaksymalizowanie okna przeglądania, aby zapobiec efektywnemu używaniu atrybutu rozdzielczości ekranu podczas ewentualnego *fingerprintingu*. Używając wyżej wymienionych sposobów, musimy także liczyć się z tym, że wiele stron internetowych może po prostu przestać działać. Niektóre z rozszerzeń, chociaż często ograniczonych przez to na jak wiele pozwalają autorzy oprogramowania, mogą paradoksalnie przyczynić się do dokładniejszego *fingerprintingu*. Omawiany *fingerprintability paradox* nazywa sytuację, w której próby ograniczenia *fingerprintingu* nieintencjonalnie tworzą nowe źródło danych, których można użyć w trakcie *fingerprintingu*. Problem widoczny jest szczególnie, wtedy kiedy danego rozszerzenia używa niewielu użytkowników. Niektóre z rozszerzeń mogą także fałszować atrybuty w taki sposób, że są one bardzo rzadkie lub wręcz nierealistyczne. Taka sytuacja sprzyja unikalnej identyfikacji. Paradoks był szeroko omawiany w środowisku naukowym [7, 24]. Autorzy monografii podkreślają też, że nie ma obecnie żadnych poważnych badań dotyczących efektywności istniejących rozszerzeń mających zapobiegać przed *fingerprintingiem*.

2.4.2. Perspektywa twórców oprogramowania

Zakres działań, które mogą podjąć twórcy przeglądarek internetowych, jest znacznie szerszy niż to, czego mogą dokonać sami użytkownicy. Aktualnie wszystkie główne przeglądarki internetowe różnią się w stopniu dokładności *fingerprintu*, jaki może zostać utworzony na podstawie danych przez nie dostępnych. Redukcja informacji lub całkowite wyeliminowanie niektórych nagłówków zapytań HTTP, jednorodność



Źródło: about:config

Rys. 2. Opcje ograniczające *fingerprinting* w Firefox 79.0

w implementacji, kontekstowy dostęp i wycofywanie starych oraz niepotrzebnych funkcji z API przeglądarek to tylko niektóre z kroków, jakie mogłoby podjąć konsorcjum przeglądarek, aby drastycznie zwiększyć ich prywatność. Istnieje szereg rekomendacji dotyczący tej kwestii [4, 6, 7, 20]. Warto także zauważyć, że praktycznie żaden z kroków możliwych do podjęcia od tej strony nie wiąże się z utratą jakości przeglądania/funkcjonowania stron internetowych. Niestety niektórzy producenci przeglądarek mogą być niechętni do podejmowania kroków mających na celu ograniczyć *fingerprinting*. Powodem tej sytuacji może być fakt, że czerpią oni korzyści finansowe z serwisów internetowych korzystających z tej metody identyfikacji [9, s. 13].

2.4.3. Czy da się skutecznie zapobiec fingerprintingowi?

Al-Fannah i Mitchell podsumowują [9, s. 18], że wydaje się nieprawdopodobne, abyśmy w najbliższym czasie przestali słyszeć o *fingerprintingu*. Badania pokazują, że jego użycie ciągle rośnie. Identyfikacja za pomocą *fingerprintingu* ma także inne

zastosowania niż śledzenie użytkowników (patrz 2.5), więc całkowite pozbycie się go bez zaproponowania innej alternatywy wydaje się niemożliwe do osiągnięcia w praktyce. Jako alternatywę autorzy proponują *Unique Browser Identifier* (UBI). Sami jednak podkreślają paradoksalną naturę proponowanego rozwiązania i obawy przed jego potencjalnie niepoprawnymi implementacjami [9, s. 16]. Biorąc pod uwagę brak stanowczych kroków ze strony producentów przeglądarek i w przypadku braku innych alternatyw przewiduje się, że efektywne zapobieżenie *fingerprintingowi* będzie w najbliższej przyszłości niemożliwe. Mając na uwadze to, jak dużym jest zagrożeniem dla prywatności użytkowników, bezsprzecznie jest to ważny przedmiot przyszłych badań.

2.5. Inne zastosowania fingerprintingu

Valentin Vasilyev, który zapoczątkował cieszący się dużą popularnością otwartoźródłowy projekt **fingerprintjs**[2] i jest obecnie współzałożycielem firmy FingerprintJS[3], na jej stronie internetowej[4] pokazuje, że *fingerprinting* posiada wiele innych, różnych zastosowań w ramach identyfikacji użytkowników. Są to między innymi zastosowania dotyczące wykrywania masowego i zautomatyzowanego tworzenia fałszywych kont oraz fałszerstw (w kontekście transakcji elektronicznych) w serwisach internetowych. Także firma Cloudflare zapobiega atakom na serwisy swoich klientów, korzystając z techniki Google Picasso[5] (*fingerprinting* urządzeń i przeglądarek opracowany przez Google [3]). *Fingerprinting* może być zatem także czymś w rodzaju drugiej warstwy uwierzytelniającej w serwisach internetowych.

2.5.1. Potencjał w operacjach wojskowych w cyberprzestrzeni

Perspektywa CyberOps i InfoOps

[2] https://github.com/fingerprintjs/fingerprintjs2
[3] https://www.linkedin.com/in/valentin-vasilyev
[4] https://fingerprintjs.com
[5] https://support.cloudflare.com/hc/en-us/articles/360045224651

Rozdział 3.

Metody fingerprintingu urządzeń i przeglądarek

Od kiedy Eckersley [7] po raz pierwszy opisał *fingerprinting* przeglądarek, metody *fingerprintingu* zmieniały się i ewoluowały razem z nimi. Sytuacja *fingerprintingu* poszczególnych protokołów czy implementacji stosu TCP/IP jest raczej odmienna. Choć te metody znane są już od lat dziewięćdziesiątych (a ich implementacje to takie narzędzia jak nmap), są relatywnie mniej „przebojowym" tematem badań i eksperymentów. Niniejszy rozdział enumeruje przez szeroką gamę metod, prezentując implementacje najważniejszych i rozważając ich skuteczność.

3.1. Pojęcie pasywnego i aktywnego fingerprintingu

Fingerprinting może być przeprowadzany w sposób pasywny lub aktywny. Pasywny polega całkowicie na informacjach, do których zdobycia nie jest wymagana interakcja z drugą stroną połączenia (np. nagłówki HTTP (takie jak *User-Agent*))—te informacje i tak zostają wysłane. Aktywny *fingerprinting* opiera się z kolei na przykład na użyciu skryptów do wydobycia większej ilości informacji (takich jak konfiguracja przeglądarki) [9, s. 3].

3.2. Źródła danych identyfikacyjnych

Mówiąc o warstwach modelu TCP/IP możemy wyróżnić następujące źródła danych (w kolejności warstw modelu TCP/IP zaczynając od warstwy dostępu do sieci):

1. Różnice w implementacjach protokołów (takich jak CDP);
2. Różnice w implementacjach protokołów (takich jak ICMP);
3. Różnice w implementacjach stosu TCP/IP;
4. Różnice w implementacjach protokołów i aplikacji korzystających z nich.

3.3. Fingerprinting implementacji stosu TCP/IP

Jedną z ciekawych właściwości różnic w implementacjach stosu TCP/IP jest fakt, iż ruch sieciowy z komputera ofiary może być analizowany w celu wykrycia, jakiego systemu operacyjnego używa (na przykład). Powodem tej sytuacji jest przeniesienie odpowiedzialności o ustaleniu wartości pewnych parametrów protokołów TCP i IP na stronę zajmującą się implementacją. Różne wersje tego samego systemu operacyjnego i różne systemy operacyjne używają różnych wartości tych parametrów. Może pozwalać to na unikalną identyfikację lub chociaż oszacowanie jakiego typu technologią jest druga strona połączenia.

Parametry, które mogą różnić się pomiędzy implementacjami stosu TCP/IP to między innymi:

- początkowy rozmiar pakietu;
- początkowy TTL;
- szerokość okna;
- maksymalna długość segmentu;
- wartość progowa, po której okno zaczyna być skalowane;

Operating System (OS)	IP Initial TTL	TCP window size
Linux (kernel 2.4 and 2.6)	64	5840
Google's customized Linux	64	5720
FreeBSD	64	65535
Windows XP	128	65535
Windows 7, Vista and Server 2008	128	8192
Cisco Router (IOS 12.4)	255	4128

Źródło: [12]

Rys. 3. Różnice w początkowym TTL i szerokości okna pomiędzy systemami

- flagi *don't fragment*, *sackOK* i *nop*.

Sumaryczna ilość informacji, które dają wymienione parametry to 67 bitów. Natomiast Hjelmvik [12] pokazuje, że jedynie nawet początkowy TTL i szerokość okna wystarczają, aby rozpoznać system operacyjny, z którego pochodzi analizowany ruch sieciowy (Rys. 3).

3.3.1. Opis algorytmu

O ile analiza ruchu sieciowego i wydobywanie odpowiednich parametrów jest relatywnie bezpośrednie w przypadku większości z nich, to pewnym problemem jest początkowa wartość TTL. W przypadku tego parametru host wysyłający pakiet ustawi wartość równą domyślnej dla danego systemu operacyjnego, ale będzie ona pomniejszana o jeden dla każdego routera na drodze do hostu docelowego. Zatem początkową wartością TTL będzie zaobserwowany TTL plus liczba skoków.

3.4. Fingerprinting przeglądarek

Znane metody *fingerprintingu* przeglądarek, które można uznać za praktycznie stosowane (na przykład będące treścią implementacji popularnych, otwartoźródłowych bibliotek czy tematem badań nad prywatnością i bezpieczeństwem komputerowym) to techniki, które polegają na użyciu takich danych jak:

- nagłówek User-Agent;
- informacja o tym, czy przeglądarka jest tzw. marionetką[6];
- język przeglądarki;
- głębia koloru ekranu urządzenia;
- wielkość RAM urządzenia;
- proporcja pomiędzy fizycznymi a logicznymi pikselami ekranu urządzenia;
- liczba procesorów logicznych urządzenia;
- rozdzielczość ekranu urządzenia;
- dostępna rozdzielczość ekranu urządzenia;
- przesunięcie strefy czasowej zegara systemowego;
- strefa czasowa zegara systemowego;
- możliwość dostępu do magazynu `sessionStorage`;
- możliwość dostępu do magazynu `localStorage`;
- możliwość dostępu do IndexedDB;
- możliwość prototypowania `HTMLElement`;
- wsparcie dla WebSQL;

[6]https://developer.mozilla.org/en-US/docs/Web/API/Navigator/webdriver

- klasa procesora urządzenia;
- platforma, na której działa przeglądarka;
- nagłówek Do Not Track;
- zainstalowane *pluginy*;
- *fingerprint* elementu `canvas`;
- *fingerprint* WebGL;
- rozszerzenie `WEBGL_debug_renderer_info`;
- użycie rozszerzeń blokujących reklamy;
- informacja o tym, czy przeglądarka fałszuje język;
- informacja o tym, czy przeglądarka fałszuje rozdzielczość ekranu urządzenia;
- informacja o tym, czy przeglądarka fałszuje system operacyjny;
- informacja o tym, czy przeglądarka fałszuje informacje o sobie;
- wsparcie dla interakcji dotykowych przez urządzenie;
- lista obsługiwanych czcionek;
- lista obsługiwanych czcionek Flash;
- *fingerprint* `AudioContext`[7];
- lista podłączonych urządzeń peryferyjnych do urządzenia.

Zbieranie powyższych danych zostanie szczegółowo omówione w niniejszym rozdziale.

[7] https://audiofingerprint.openwpm.com/

3.4.1. Canvas fingerprinting

Przeglądarki internetowe są coraz bardziej skomplikowanymi programami komputerowymi i coraz częściej udostępniają stronom internetowym funkcjonalności do tej pory oferowane jedynie przez system operacyjny. Pewnym naturalnym schematem, aby zaimplementować niektóre zaawansowane funkcjonalności przeglądarek, jest korzystanie z zasobów systemowych i sprzętowych hosta. Na przykład przetwarzanie grafiki używając karty graficznej, daje ogromny skok wydajnościowy i bardziej oszczędza baterię w przypadku urządzeń mobilnych. Nie inaczej jest w przypadku elementu `canvas`. HTML5 zawiera w sobie element, na którym można programowo rysować, a użycie natywnych mechanizmów systemu operacyjnego *renderingu* czcionek do tekstu (na przykład[8]) gwarantuje, że rezultat będzie zoptymalizowany pod kątem wyświetlania na konkretnym ekranie i zgodny z oczekiwaniami użytkownika. Ściślejsze powiązanie przeglądarki internetowej z funkcjonalnością systemu operacyjnego i samym sprzętem oznacza, że strony internetowe mają większy dostęp do tych zasobów, a zachowanie przeglądarki różni się w zależności od zachowania tych zasobów [17, s. 1]. Trudno wyobrazić sobie liczbę warstw abstrakcji od samego sprzętu, przez sterowniki urządzeń i system operacyjny, aż po wyświetlany efekt na ekranie użytkownika. Im dalej wgłąb tego stosu, tym więcej nieścisłości uwypukli się, rozważając różne konfiguracje sprzętowe, systemy operacyjne i przeglądarki internetowe. Z tej prawidłowości został wcześniej w niniejszej pracy wyciągnięty wniosek, że *fingerprint* urządzenia może zawierać się w *fingerprincie* przeglądarki.

Analogiczną techniką jest *fingerprinting* WebGL, która także polega na rysowaniu w elemencie `canvas` (scen trójwymiarowych).

Historia

W 2012 roku Keaton Rowery i Hovav Schacham z Uniwersytetu Kalifornijskiego w San Diego opublikowali artykuł „Pixel Perfect: Fingerprinting Canvas in HTML5” [17],

[8] Oczywiście element `canvas` pozwala na rysowanie praktycznie dowolnych rzeczy (obiektów): https://developer.mozilla.org/pl/docs/Web/API/Canvas_API/Tutorial

który jako pierwszy poruszył temat Canvas fingerprintingu. Autorzy określili wówczas tę technikę jako deterministyczną, o wysokiej entropii, niezależną od innych technik, transparentną (niezauważalną) dla użytkownika i prostą do zastosowania [17, s. 1]. Podsumowują, że wszystkie nowe możliwości przeglądarek internetowych mają swój koszt [17, s. 3]. Ich eksperymenty polegały głównie na rysowaniu tekstu w elemencie `canvas` i pokazały, że co najmniej używany system operacyjny, wersja przeglądarki, karta graficzna, zainstalowane czcionki, hinting czcionek na poziomie subpiksela, antyaliasing, a nawet konkretne ułożenie pikseli na fizycznym ekranie (coś, co mogą brać pod uwagę takie technologie jak ClearType) grają role w generowaniu bitmapy z elementu `canvas`. Nawet czcionka Arial, która ma ponad 30 lat *renderowała* się „na nowe i interesujące sposoby" w eksperymentach [17, s. 6]. Canvas fingerprinting rozgłos zdobył jednak dopiero po publikacji „The Web Never Forgets: Persistent Tracking Mechanisms in the Wild" z 2014 roku [1]. Wtedy o Canvas fingerprintingu pisały wszystkie największe media na świecie.

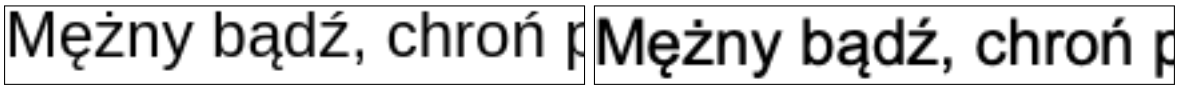
Pseudokod

Algorytm 1. Canvas fingerprinting

Dane : Element canvas

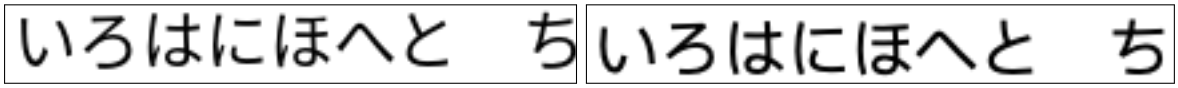
Wynik: Reprezentacja graficzna elementu canvas

if *przeglądarka obsługuje wybranie kontekstu graficznego* **then**
    ctx ← jeden z kontekstów graficznych, np. 2d;
    `// użycie` ctx, `aby wpłynąć na element` canvas
    if *przeglądarka obsługuje pakowanie elementu* canvas **then**
        return *reprezentacja elementu* canvas;
    return *informacja o braku możliwości wykonania algorytmu*;
return *informacja o braku możliwości wykonania algorytmu*;



Źródło: Wykonanie Implementacji 1 w Firefox 81.0.2/Ubuntu 20.04.1 i Safari 13.0.5/macOS 10.15.3

Rys. 4. Rezultat wykonania Implementacji 1 w Firefox/Ubuntu i Safari/macOS



Źródło: Wykonanie Implementacji 1 w Firefox 81.0.2/Ubuntu 20.04.1 i Safari 13.0.5/macOS 10.15.3

Rys. 5. Rezultat wykonania Implementacji 1 w Firefox/Ubuntu i Safari/macOS

Implementacja w JavaScript

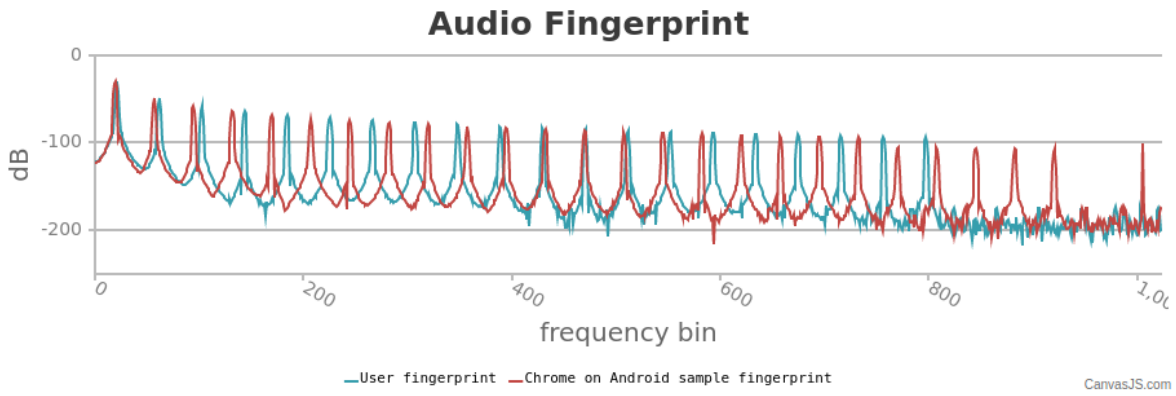
Implementacja 1 przedstawia implementację Canvas fingerprintingu w JavaScript. Różnice pomiędzy wykonaniem algorytmu w przeglądarce Firefox działającej na systemie Ubuntu i przeglądarce Safari działającej na systemie macOS przedstawiają Rys. 4 i Rys. 5. Niestety różnice widoczne są gołym okiem bez konieczności *hashowania* obrazów. Najprawdopodobniej świadczy to o olbrzymim potencjale tej techniki w odniesieniu do *fingerprintingu*. Badania przeprowadzone przez Mowery'ego i Shachama potwierdzają, że użycie tej techniki pozwala uzyskać relatywnie wysoką entropię [17].

3.4.2. Web Audio fingerprinting

Techniki *fingerprintingu* używające Web Audio API (i AudioContext API) działają na podobnej zasadzie co techniki używające Canvas API. W przypadku tych rozwiązań schemat działania jest mniej więcej taki sam: przekaż przeglądarce zadanie do wykonania i zaobserwuj, jak je wykonuje (wykonała), a następnie zbuduj na podstawie tych (i innych) danych jej *fingerprint*. W przypadku Web Audio fingerprintingu używany jest stos dźwiękowy (w sensie technologii), z którego korzysta przeglądarka. Otrzymuje ona do wykonania zadanie wygenerowania sygnału dźwiękowego na podstawie preferencji dźwiękowych urządzenia i zainstalowanego sprzętu audio.

```
1  'use␣strict';
2
3  const font = '18pt␣Arial';
4  const textBaseline = 'top';
5  const text = 'Mężny␣bądź,␣chroń␣pułk␣twój␣i␣sześć␣flag.';
6
7  const canvas = document.createElement('canvas');
8
9  const ctx = canvas.getContext('2d');
10 if (ctx === null) {
11     throw new Error('Canvas␣fingerprinting␣failed!');
12 }
13
14 ctx.font = font;
15 ctx.textBaseline = textBaseline;
16
17 canvas.width = Math.ceil(ctx.measureText(text).width);
18
19 ctx.font = font;
20 ctx.textBaseline = textBaseline;
21 ctx.fillText(text, 0, 0);
22
23 document.body.appendChild(canvas);
24
25 const dataURL = canvas.toDataURL();
26
27 const w = window.open(dataURL);
28 if (w === null) {
29     console.debug(dataURL);
30 }
```

Implementacja 1. Podstawowy Canvas fingerprinting w JavaScript



Źródło: https://audiofingerprint.openwpm.com/

Rys. 6. Rezultat wykonania Implementacji 2 w Firefox/Ubuntu i Chrome/Android

Historia

Łatwo domyślić się, że Web Audio fingerprinting narodził się tuż po „skandalach" związanych z Canvas fingerprintingiem. Zgodnie z najlepszą wiedzą autora niniejszej pracy nie istnieje żaden kamień milowy (oprócz wspomnianego), który można by uznać za początek lub eskalację użycia tej techniki. Jej możliwości badał m.in. projekt Princeton Web Transparency & Accountability Project[9][10].

Implementacja w JavaScript

Pseudokod został pominięty ze względu na zbyt dużą szczegółowość i konieczność przemycenia szczegółów implementacyjnych. Przedstawiona Implementacja 2 jest najprostszym wariantem Web Audio fingerprintingu. Różnice pomiędzy wykonaniem algorytmu w przeglądarce Firefox działającej na systemie Ubuntu i przeglądarce Chrome działającej na systemie Android przedstawia Rys. 6.

3.4.3. Pozostałe techniki

Pozyskanie innych, wymienionych na powyższej liście danych jest niestety dużo prostsze niż opisany wcześniej Canvas fingerprinting i Web Audio fingerprinting.

[9] https://webtap.princeton.edu/
[10] https://audiofingerprint.openwpm.com/

```
1  const ctx = new AudioContext();
2
3  const oscillator = ctx.createOscillator();
4  const analyser = ctx.createAnalyser();
5  const gain = ctx.createGain();
6  const scriptProcessor = ctx.createScriptProcessor(4096, 1, 1);
7
8  gain.gain.value = 0; // Wyciszenie.
9
10 // Zmiana wyjścia oscylatora na falę trójkątną.
11 oscillator.type = 'triangle';
12
13 oscillator.connect(analyser);
14 analyser.connect(scriptProcessor);
15 scriptProcessor.connect(gain);
16 gain.connect(ctx.destination);
17
18 scriptProcessor.onaudioprocess = function (bins) {
19     bins = new Float32Array(analyser.frequencyBinCount);
20
21     analyser.getFloatFrequencyData(bins);
22     analyser.disconnect();
23
24     const output = Array.from(bins);
25
26     scriptProcessor.disconnect();
27     gain.disconnect();
28
29     console.info(output); // Prezentacja wyniku.
30 };
31
32 oscillator.start(0); // Uruchomienie przetwarzania potoku.
```

Implementacja 2. Podstawowy Web Audio fingerprinting w JavaScript

```
1 console.info(navigator.userAgent)
```

Implementacja 3. Wypisywanie nagłówka User-Agent do konsoli JavaScript

Zwykle polega to na skorzystaniu z API przeglądarek, aby odczytać konkretny atrybut. W ogólności jest to trywialne zadanie. Oprócz tego potrzebna jest wiedza, jakich atrybutów pozyskiwanie ma sens i jak je interpretować oraz przetwarzać. Dla przykładu Implementacja 3 przedstawia pozyskiwanie nagłówka User-Agent.

Rozdział 4.

Eksperymentalny algorytm klasyfikatora fingerprintów

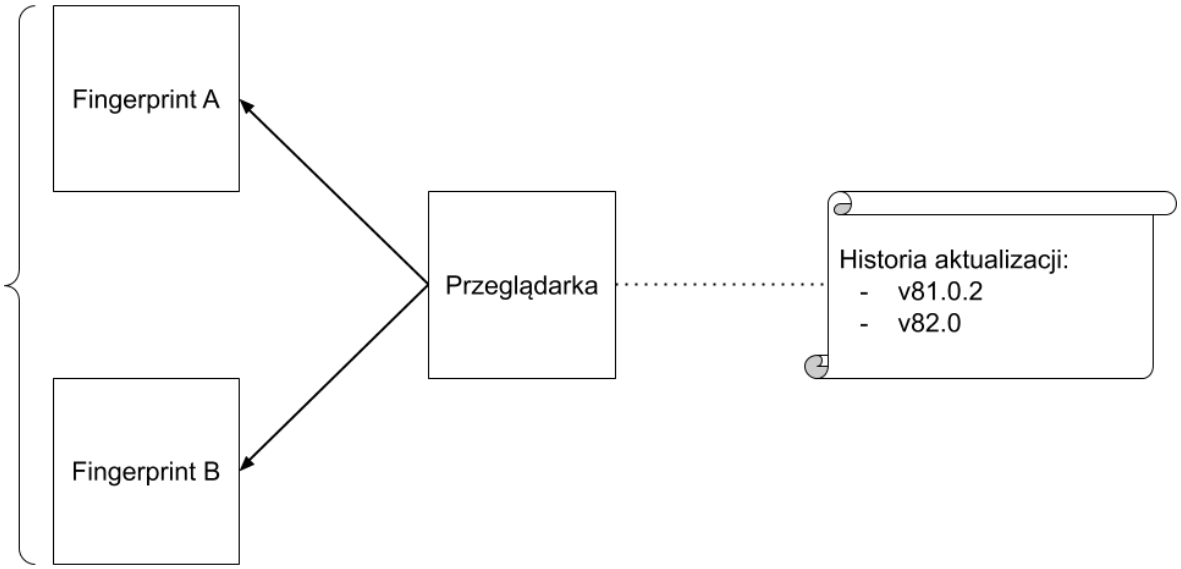
4.1. Motywacja

Wcześniej w niniejszej pracy zostało zaznaczone, że *fingerprint* może być bardziej unikalny kosztem jego stabilności i odwrotnie. Aby zgrupować kolejne *fingerprinty* tej samej przeglądarki internetowej, których różnice wynikają z naturalnych przemian tej przeglądarki (np. aktualizacja), możemy użyć klasyfikatora opartego na odpowiednim algorytmie. Tę relację przedstawia Rys. 7. Dzięki takiemu rozwiązaniu jesteśmy w stanie zachować wiarygodny wskaźnik entropii, używając średnio bardziej unikalnych *fingerprintów* o mniejszej stabilności.

Podobne (choć bardziej ograniczone) rozwiązanie zastosował Eckersley w swojej pracy, w której po raz pierwszy opisał *fingerprinting* przeglądarek internetowych [7, s. 13].

4.2. Opis algorytmu

Aby zgrupować kolejne, podobne do siebie *fingerprinty* należy wyznaczyć podobieństwo każdego nowego *fingerprintu* do wszystkich przechowywanych i utożsamić go z tym, do którego jest najbardziej podobny (powyżej pewnej wartości podobień-



Źródło: Diagram przygotowany przez autora niniejszej pracy

Rys. 7. Relacja *fingerprint*–przeglądarka

stwa). Można to zrobić na parę sposobów. Jednym z najsensowniejszych rozwiązań wydaje się podejście, w którym określa się podobieństwo cech (komponentów) *fingerprintu*, pomiędzy którymi istnieje relacja większości (częściowy porządek). Tak można porównywać na przykład nagłówki User-Agent (zawiera w sobie wersję oprogramowania) jako komponent dwóch porównywanych *fingerprintów*. Z tych porównań można wnioskować o ogólnym podobieństwie dwóch *fingerprintów*.

Porównywanie podobieństwa komponentów, które bezpośrednio lub pośrednio wynikają z np. parametrów sprzętowych urządzenia (na przykład *fingerprint* elementu `canvas`), wydaje się bezcelowe. Ich wartość jest zwykle stała i należałoby zastanowić się nad istnieniem częściowego porządku w takim zbiorze. Jednakże ta niezmienność i dalej porównywanie tych komponentów na zasadzie równoważności może być przydatne w szacowaniu, w jakim stopniu ogólne podobieństwo jest wiarygodne.

Metryką określającą podobieństwo do siebie dwóch komponentów będzie odległość Levenshteina, która jest powszechnie stosowaną miarą odmienności skończonych ciągów znaków.

4.2.1. Odległość Levenshteina

Algorytmy wyznaczające odległość Levenshteina nie są na tyle popularne, aby znalazły one swoją implementację w bibliotekach standardowych większości popularnych języków oprogramowania. Z tego względu autor niniejszej pracy zdecydował się na własnoręczną implementację.

Definicja

Odległość Levenshteina pomiędzy dwoma łańcuchami znaków a i b o długościach odpowiednio i i j, zdefiniowana jest jako

$$\mathrm{ol}(i,j) = \begin{cases} \max(i,j) & \text{jeśli } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{ol}(i-1,j)+1 \\ \mathrm{ol}(i,j-1)+1 \\ \mathrm{ol}(i-1,j-1) & \text{jeśli } a_i = b_j, \\ \mathrm{ol}(i-1,j-1)+1 & \text{jeśli } a_i \neq b_j. \end{cases} & \text{w innym przypadku.} \end{cases}$$

Algorytm Wagnera–Fischera

Bezpośrednia implementacja (z definicji) algorytmu wyznaczającego odległość Levenshteina ma zasadniczy problem: tak zaimplementowany algorytm będzie działać w czasie ponadwielomianowym. Aby się o tym przekonać, wystarczy narysować drzewo wywołań rekurencyjnych i zauważyć, że kolejne wywołania nie są rozłączne (nie jest to oczywiście formalny dowód). Wyznaczanie odległości Levenshteina cechuje własność optymalnej podstruktury i bazując na programowaniu dynamicznym, można znaleźć algorytm działający w czasie wielomianowym.

Jednym z takich algorytmów jest algorytm Wagnera–Fischera[11] (Algorytm 2) [25]. Implementacja 4 przedstawia implementację algorytmu Wagnera–Fischera. Implementacja 5 jest optymalnym wariantem tego algorytmu (biorąc pod uwagę zużycie pamięci; Implementacja 4 zużywa kwadratową ilość pamięci a Implementacja 5 liniową). Taki wariant jest stosowany później w algorytmie klasyfikatora.

4.3. Pseudokod

Algorytm 3 obrazuje eksperymentalny algorytm klasyfikatora *fingerprintów*. Funkcja `hardwareRelatedCompatibility` odpowiada za binarne porównanie binarnych

[11] Algorytm miał wielu wynalazców. Nazwa „Wagnera–Fischera” powinna być traktowana jedynie jako zwyczajowa: https://en.wikipedia.org/wiki/Wagner-Fischer_algorithm#History

Algorytm 2. Algorytm Wagnera–Fischera

Dane : ciągi znaków a i b o długościach odpowiednio m i n

Wynik: odległość Levenshteina

```
d ← [0 ... m, 0 ... n];
for i ← 1 to m do
    d[i, 0] ← i;
for j ← 1 to n do
    d[0, j] ← j;
for j ← 1 to n do
    for i ← 1 to m do
        if a[i − 1] = b[j − 1] then
            c ← 0;
        else
            c ← 1;
        d[i, j] ← minimum(d[i − 1, j] + 1, d[i, j − 1] + 1, d[i − 1, j − 1] + c);
return d[m, n];
```

```
1  func WagnerFischer(a, b string) int {
2      m, n := utf8.RuneCountInString(a), utf8.RuneCountInString(b)
3      d := make([][]int, m+1)
4      for i := range d {
5          d[i] = make([]int, n+1)
6      }
7      for i := 1; i <= m; i++ {
8          d[i][0] = i
9      }
10     for j := 1; j <= n; j++ {
11         d[0][j] = j
12     }
13     j := 1
14     for _, r2 := range b {
15         i := 1
16         for _, r1 := range a {
17             var cost int
18             if r1 != r2 {
19                 cost = 1
20             }
21             d[i][j] = min(d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1]+cost)
22             i++
23         }
24         j++
25     }
26     return d[m][n]
27 }
```

Implementacja 4. Algorytm Wagnera–Fischera w Go

```go
func LinearSpace(a, b string) int {
    m, n := utf8.RuneCountInString(a), utf8.RuneCountInString(b)
    if m > n {
        a, b = b, a
        m, n = n, m
    }
    d := make([]int, m+1)
    for i := 1; i <= m; i++ {
        d[i] = i
    }
    j := 1
    for _, r2 := range b {
        prev := d[0]
        d[0] = j
        i := 1
        for _, r1 := range a {
            var cost int
            if r1 != r2 {
                cost = 1
            }
            curr := d[i]
            d[i] = min(curr+1, d[i-1]+1, prev+cost)
            prev = curr
            i++
        }
        j++
    }
    return d[m]
}
```

Implementacja 5. Algorytm Wagnera–Fischera w Go (liniowa pamięć)

komponentów (w tym wcześniej omawianych, teoretycznie niezmiennych komponentów). Funkcja `similarity` bazując na odległości Levenshteina, przeprowadza porównanie dla reszty komponentów. Implementacja 6 przedstawia implementację tego algorytmu w Go.

Algorytm 3. Eksperymentalny klasyfikator

Dane : Zbiór *fingerprintów* G i *fingerprint* f

Wynik: *fingerprint*, do którego f jest najbardziej podobny, jeśli istnieje

```
max ← 0;
foreach g ∈ G do
    if hardwareRelatedCompatibility(f, g) < α then
        continue;
    ratio ← similarity(f, g);
    if ratio > max then
        max ← ratio;
        candidate ← g;
if max ≥ β then
    return candidate;
return NULL;
```

4.4. Ocena złożoności czasowej i pamięciowej

Załóżmy, że `hardwareRelatedCompatibility` działa w czasie stałym. Zgodnie z wcześniejszym ustaleniami `similarity` powinna działać w czasie co najwyżej kwadratowym, proporcjonalnym do długości *fingerprintu* (algorytm Wagnera–Fischera działa w czasie $\Theta(mn)$, gdzie m i n to długości porównywanych ciągów). Warto jednak zauważyć, że długość *fingerprintu* jest stała. Proponowany algorytm przegląda wszystkie zapisane wcześniej *fingerprinty* i dla każdego obrotu wykonuje stałą

```go
func (e ExperimentalInMemory) Do(f Fingerprint) (Fingerprint, bool) {
    var (
        max       float64
        candidate Fingerprint
    )

    e.mtx.RLock()
    for _, fingerprints := range e.fingerprints {
        for _, g := range fingerprints {
            if hardwareRelatedCompatibility(f, g) < 0.85 {
                continue
            }

            if ratio := similarity(f, g); ratio > max {
                max, candidate = ratio, g
            }
        }
    }
    e.mtx.RUnlock()

    if max > 0.95 {
        return candidate, true
    }

    return Fingerprint{}, false
}
```

Implementacja 6. Eksperymentalny klasyfikator w Go

pracę. Możemy zatem stwierdzić, że proponowany algorytm działa w czasie liniowym.

Mając na uwadze rozważania z poprzedniego akapitu, możemy także stwierdzić, że pamięć używana przez proponowany algorytm jest stała.

4.5. Ocena efektywności

4.5.1. Opis utworzonego rozwiązania

W celu zbadania efektywności proponowanego algorytmu utworzono system złożony z dokumentowej bazy danych przechowującej zbierane przez aplikację serwera (implementującego Algorytm 3; Implementacja 6) *fingerprinty*, który serwował także uruchamiany po stronie przeglądarki klienta (ofiary) skrypt je generujący. Komponenty używane do generowania *fingerprintu* przedstawiają Tab. 1 i Tab. 2. Listing kodu serwera i kodu uruchamianego po stronie klienta dostępny jest w Dodatku A.

4.5.2. Testy efektywności

Po wejściu na stronę interfejs widoczny na Rys. 8 informował użytkownika o tym, czy *fingerprint* jego przeglądarki/urządzenia znajduje się już w bazie danych, skrócie *fingerprintu*, czasie wykonania skryptu i zebranych komponentach.

Krótkie (dobrowolne) testy przeprowadzone na grupie znajomych autora pokazały, że prosta heurystyka bez problemu poradziła sobie z unikalną identyfikacją użytkowników, a także identyfikacją pomiędzy normalnym trybem przeglądarki a tzw. trybem prywatnym (najnowsze wersje niektórych przeglądarek internetowych fałszują wartości komponentów pomiędzy trybami—w domyśle w celach ochrony prywatności). Algorytm radził sobie często także w unikalnej identyfikacji użytkowników pomiędzy przeglądarkami zainstalowanymi na tym samym urządzeniu.

Dalsza praca

Tab. 1. Komponenty używane w implementacji klasyfikatora

Komponent	Związany z konfiguracją sprzętową?
fingerprint elementu `canvas`	Tak
`Error.prototype.toSource()`	Nie
`eval.toString().length`	Nie
`navigator.browserLanguage`	Nie
`navigator.cpuClass`	Tak
`navigator.deviceMemory`	Tak
`navigator.hardwareConcurrency`	Tak
`navigator.language`	Nie
`navigator.languages`	Nie
`navigator.maxTouchPoints`	Tak
`navigator.oscpu`	Nie
`navigator.platform`	Nie
`navigator.plugins`	Nie
`navigator.productSub`	Nie
`navigator.systemLanguage`	Nie
`navigator.userLanguage`	Nie
`navigator.vendor`	Nie
`TouchEvent`	Tak
`window.chrome`	Nie
`window.indexedDB`	Nie
`window.Intl.DateTimeFormat()`	Nie
`window.localStorage`	Nie
`window.OfflineAudioContext`	Tak
`window.ontouchstart`	Tak
`window.openDatabase`	Nie
`window.screen.availHeight`	Tak
`window.screen.availWidth`	Tak
`window.screen.colorDepth`	Tak

Źródło: Zrzut ekranu wykonany na urządzeniu Apple iPad Air

Rys. 8. Interfejs aplikacji klienckiej eksperymentalnego klasyfikatora

Tab. 2. Komponenty używane w implementacji klasyfikatora cd.

Komponent	Związany z konfiguracją sprzętową?
`window.screen.height`	Tak
`window.screen.width`	Tak
`window.sessionStorage`	Nie
`window.webkitOfflineAudioContext`	Tak
Lista czcionek	Nie
Obsługa *cookies*	Nie
Przesunięcie strefy czasowej	Nie

Rozdział 5.

Podsumowanie

Spis rysunków

1. Wolne pule adresów IPv4 w czasie 13

2. Opcje ograniczające *fingerprinting* w Firefox 79.0 23

3. Różnice w początkowym TTL i szerokości okna pomiędzy systemami 27
4. Rezultat wykonania Implementacji 1 w Firefox/Ubuntu i Safari/macOS 32
5. Rezultat wykonania Implementacji 1 w Firefox/Ubuntu i Safari/macOS 32
6. Rezultat wykonania Implementacji 2 w Firefox/Ubuntu i Chrome/Android 34

7. Relacja *fingerprint*–przeglądarka 38
8. Interfejs aplikacji klienckiej eksperymentalnego klasyfikatora 48

Spis tablic

1. Komponenty używane w implementacji klasyfikatora 47
2. Komponenty używane w implementacji klasyfikatora cd. 49

Spis algorytmów

1 Canvas fingerprinting 31

2 Algorytm Wagnera–Fischera 41

3 Eksperymentalny klasyfikator 44

Spis implementacji

1. Podstawowy Canvas fingerprinting w JavaScript 33
2. Podstawowy Web Audio fingerprinting w JavaScript 35
3. Wypisywanie nagłówka User-Agent do konsoli JavaScript 36
4. Algorytm Wagnera–Fischera w Go 42
5. Algorytm Wagnera–Fischera w Go (liniowa pamięć) 43
6. Eksperymentalny klasyfikator w Go 45

Bibliografia

[1] Gunes Acar i in. „The web never forgets: Persistent tracking mechanisms in the wild”. W: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, s. 674–689.

[2] United Nations. General Assembly. *Universal declaration of human rights*. T. 3381. Department of State, United States of America, 1949.

[3] Elie Bursztein i in. „Picasso: Lightweight Device Class Fingerprinting for Web Clients”. W: *Workshop on Security and Privacy in Smartphones and Mobile Devices*. 2016.

[4] Alissa Cooper i in. „Privacy considerations for internet protocols”. W: *Internet Architecture Board* (2013).

[5] Edward E David Jr i Robert M Fano. „Some thoughts about the social implications of accessible computing”. W: *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*. 1965, s. 243–247.

[6] N Doty. „Fingerprinting guidance for Web specification authors”. W: *W3C, Unofficial Draft, Oct* (2014).

[7] Peter Eckersley. „How unique is your web browser?” W: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2010, s. 1–18.

[8] Kjeld Egevang, Paul Francis i in. *The IP network address translator (NAT)*. Spraw. tech. RFC 1631, may, 1994.

[9] Nasser Mohammed Al-Fannah i Chris Mitchell. „Too little too late: can we control browser fingerprinting?” W: *Journal of Intellectual Capital* (2020).

[10] Erik Flood i Joel Karlsson. „Browser fingerprinting". W: (2012).

[11] Vince Fuller i in. *Supernetting: An address assignment and aggregation strategy*. Spraw. tech. RFC-1338, June, 1992.

[12] Erik Hjelmvik. „Passive os fingerprinting". W: *Netresec blog* (2011).

[13] Robert Kahn i Vint Cerf. „A protocol for packet network intercommunication". W: *IEEE Transactions on Communications* 22.5 (1974), s. 637–648.

[14] Michal Kosinski, David Stillwell i Thore Graepel. „Private traits and attributes are predictable from digital records of human behavior". W: *Proceedings of the national academy of sciences* 110.15 (2013), s. 5802–5805.

[15] Eric Maigret. *Socjologia komunikacji i mediów*. Warszawa: Oficyna Naukowa, 2012.

[16] MG Michael. *Uberveillance and the Social Implications of Microchip Implants: Emerging Technologies: Emerging Technologies*. IGI Global, 2013.

[17] Keaton Mowery i Hovav Shacham. „Pixel perfect: Fingerprinting canvas in HTML5". W: *Proceedings of W2SP* (2012), s. 1–12.

[18] Thomas Narten, Richard Draves i Suresh Krishnan. *Privacy extensions for stateless address autoconfiguration in IPv6*. Spraw. tech. RFC 3041, January, 2001.

[19] Sheau Ng. „A brief history of entertainment technologies". W: *Proceedings of the IEEE* 100.Special Centennial Issue (2012), s. 1386–1390.

[20] Nick Nikiforakis i in. „On the workings and current practices of web-based device fingerprinting". W: *IEEE security & privacy* 12.3 (2014), s. 28–36.

[21] Gil Press. „A very short history of the Internet and the Web". W: *Forbes. Luettavissa: https://www. forbes. com/sites/gilpress/2015/01/02/a-very-short-history-of-the-internet-andthe-web-2* (2015).

[22] Bruce Schneier. „The eternal value of privacy". W: *Comment on Wired. com, May* (2006).

[23] Edward Snowden. *Pamięć nieulotna*. Kraków: Insignis, 2019. ISBN: 978-83-66360-15-0.

[24] Christof Ferreira Torres, Hugo Jonker i Sjouke Mauw. „FP-Block: usable web privacy by controlling browser fingerprinting". W: *European Symposium on Research in Computer Security*. Springer. 2015, s. 3–19.

[25] Robert A Wagner i Michael J Fischer. „The string-to-string correction problem". W: *Journal of the ACM (JACM)* 21.1 (1974), s. 168–173.

[26] Jun Wu. *The beauty of mathematics in computer science*. CRC Press, 2018.