Browser Fingerprinting: Tracking ohne Spuren zu hinterlassen

Henning Tillmann

Diplomarbeit
zur Erlangung des akademischen Grades
Diplominformatiker

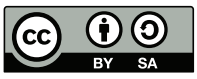
Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Aktualisierte Fassung vom 20. Oktober 2013 mit redaktionellen Änderungen.
Ursprüngliche Version eingereicht am 24. Mai 2013.
Gutachter: Prof. Dr. Coy und Prof. Dr. Malek.

Vielen Dank an alle, die an der Feldstudie teilgenommen und auf sie hingewiesen haben. Ohne die Artikel auf zeit.de, netzpolitik.org, heise.de, golem.de, u. v. m. hätte ich 23.709 Fingerprints niemals sammeln können.

Sascha, Katja, Lena, Jennifer, Florian, Marlis, Heiner, Judith, Andi, Andreas – Danke!
Die Arbeit ist Heinrich Bröker gewidmet.

Verwendete musikalische Hilfsmittel zwischen August 2012 und Mai 2013: Metallica, Queen, Kyuss, Deftones, Johnny Cash, Death (Chuck Schuldiner), Volbeat, Gorilla Biscuits, Maserati, Cynic, Refused, Hundred Seventy Split.



http://www.henning-tillmann.de

Abstract

Bei dem Aufruf einer Website werden Informationen, u. a. Angaben über das verwendete Betriebssystem und den verwendeten Browser, an den Server gesendet. Über lokal ausgeführte Skripte oder Plugins lassen sich weitere Konfigurationsmerkmale abfragen. So können die verwendete Bildschirmauflösung, die Systemfarben, die Liste der installierten Schriftarten und weitere Eigenschaften des Geräts ermittelt und dann zum Server übertragen werden. Aus diesen Merkmalen kann ein virtueller Fingerabdruck (Browser Fingerprint) erstellt werden, der den Nutzer möglicherweise eindeutig identifiziert.

In dieser Diplomarbeit werden die einzelnen Merkmale des Fingrprintings vorgestellt und in einer begleitenden Feldstudie eingesetzt. Ihr geht die Hypothese voraus, dass viele, wenn auch nicht alle, Browser Fingerprints einzigartig sind. Ebenso wird in dem Dokument erläutert, wie diese zum Tracking von Nutzern verwendet werden können, welche weiteren Möglichkeiten abseits der klassischen Verwendung von Cookies existieren und ob sich der Nutzer davor schützen kann.

Abstract

By requesting a website, certain types of information, like the name of the used operating system and browser, are transmitted to the server. Client-side scripting and plugins can also determine other configuration characteristics, i. e. screen resolution, system colors, list of installed fonts, among others. All fetched data can be combined to create a virtual fingerprint of the specific client computer. Conceivably, a browser fingerprint could be unique and clearly identify an associated device.

This diploma thesis describes possible fingerprinting characteristics and is accompanied by a field research, in which the hypothesis that many—although not all—browser fingerprints are unique will be tested. Moreover, this document illustrates how those can be used for tracking purposes, what other possible tracking mechanism besides the classic use of cookies are available and finally, whether users can protect themselves against unwanted tracking.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ASCII	American Standard Code for Information Interchange
AJAX	Asynchronous JavaScript and XML
BDSG	Bundesdatenschutzgesetz
CSS	Cascading Style Sheets
DIPS	Device Independent Pixels
DOM	Document Object Model
EFF	Electronic Frontier Foundation
EuGH	Europäischer Gerichtshof
GIF	Graphics Interchange Format
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
KiB	Kibibyte
LAN	Local Area Network
LSO	Local Shared Object
MD5	Message-Digest Algorithm 5
MIME	Multipurpose Internet Mail Extensions
NAT	Network Address Translation

OSI	Open Systems Interconnection Reference Model
PDF	Portable Document Format
PNG	Portable Network Graphics
RGB	Farbanteil Rot, Grün und Blau
SQL	Structured Query Language
SWF	Shockwave Flash
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
UTF-8	Universal Character Set Transformation Format
WWW	World Wide Web
W3C	World Wide Web Consortium
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

1 Einführung

Die Grundidee des World Wide Web (WWW) und somit von Websites liegt in der Verknüpfung und Verlinkung verschiedener Ressourcen. Hyperlinks sind damit für Internetseiten elementar, der Übergang von einem Webangebot zum nächsten äußerst einfach. Schon die erste Implementierung des für das World Wide Web benötigten Standards Hypertext Transfer Protocol (HTTP) beinhaltete den Eintrag „Referer“ [sic!].[1] Dadurch kann der aufrufenden Seite mitgeteilt werden, von welchem anderen Dokument der Besucher hergeleitet wird. Dies ist der Anfang des Trackings im World Wide Web: Anbieter von Inhalten können herausfinden, woher die Besucher hauptsächlich stammen, also welche referenzierende Seite ihnen am meisten Besucher verschafft.

Mit der Einführung von HTTP-Cookies (meist nur kurz: Cookies) Mitte der 1990er Jahre wurde es möglich, dass Textinformationen auf den Computern der Webseitenbesucher abgelegt werden konnten (siehe Kapitel 2.3.3, S. 21). Cookies werden seitdem häufig dazu verwendet, eine einzigartige Zeichen- bzw. Zahlenkombination auf dem Gerät des Nutzers zu hinterlegen, wodurch dieses eindeutig identifiziert werden kann. Cookies werden, sofern sie nicht abgelaufen sind, vom Nutzer unterbunden oder gelöscht werden, bei einem erneuten Aufruf der Webseite mitgeschickt. Somit ist es für Anbieter von Webinhalten nicht nur möglich zu ermitteln, über welchen Hyperlink er auf seine Seite gestoßen ist („Referer“), sondern auch, ob und wann er das Webangebot erneut aufruft.

Um auch Seitenaufrufe über mehrere Webangebote verfolgen zu können, kann ein Drittanbieter-Cookie verwendet werden. Dabei wird eine Resource von einem Anbieter auf unterschiedliche Webpräsenzen eingebunden, wodurch die Aufrufe der Seiten einem Benutzer zugeordnet werden können. Insbesondere die Werbebranche nutzt dieses Verfahren, um personalisierte Werbung anzeigen

[1] Vgl. Berners-Lee: *Basic HTTP as defined in 1992*, Abschnitt „Request“.

zu können.

Dieses Vorgehen stößt aber insbesondere bei Datenschützern auf Kritik. Wenn ein Anbieter eines Werbenetzwerkes so groß ist, dass viele Webseiten seine Werbebanner schalten, kann dieser Anbieter ein umfassendes Profil und eine Chronik der besuchten Websites einer Person (genauer: der Personen, die sich einen Computer mit einem Browser teilen) erstellen. Werden diese Daten z. B. mit persönlichen Informationen aus sozialen Netzwerken oder auch dem Klarnamen des Nutzers verknüpft, so hat der Werbenetzbetreiber einen umfassenden Überblick über die Vorlieben und Interessen einer daher auch namentlich bekannten Person.

Auch in der Politik werden genau diese Datenschutzprobleme diskutiert. Als Beispiel sei hier die sog. „Cookie-Richtlinie“ der Europäischen Union genannt.[2] Diese Richtlinie, die von den Mitgliedsstaaten in nationales Recht überführt werden muss, sieht u. a. vor, dass Cookies zum Tracking durch Drittanbietern nicht mehr ohne Einverständnis möglich sein werden. So heißt es in der Richtlinie:

> Die Mitgliedstaaten stellen sicher, dass die Speicherung von Informationen oder der Zugriff auf Informationen, die bereits im Endgerät eines Teilnehmers oder Nutzers gespeichert sind, nur gestattet wird, wenn der betreffende Nutzer auf der Grundlage von klaren und umfassenden Informationen, die er gemäß der Richtlinie 95/46/EG über die Zwecke der Verarbeitung erhält, seine Einwilligung gegeben hat. Dies steht einer technischen Speicherung oder dem Zugang nicht entgegen, wenn der alleinige Zweck die Durchführung der Übertragung einer Nachricht über ein elektronisches Kommunikationsnetz ist oder wenn dies unbedingt erforderlich ist, damit der Anbieter eines Dienstes der Informationsgesellschaft, der vom Teilnehmer oder Nutzer ausdrücklich gewünscht wurde, diesen Dienst

[2] Vgl. EU-Richtlinie 2002/58/EG, zuletzt geändert durch 2009/136/EC.

zur Verfügung stellen kann. [3]

Da auch häufig verwendete Browser, wie Firefox, künftig Drittanbieter Cookies in den Werkseinstellungen ablehnen werden,[4] suchen Anbieter von Werbenetzwerken nach alternativen Möglichkeiten, um ein Tracking von Nutzern durchführen zu können. Eine Option wäre das Erstellen von „Browser Fingerprints“, also Fingerabdrücken der System- und Browserkonfiguration des Besuchers. Anders als bei Cookies müssten hierbei keine Informationen auf dem Client-Computer abgelegt werden. Einige Merkmale eines Fingerabdrucks werden bereits bei dem Aufruf einer Webressource vom Client mitgeschickt, andere können durch Scriptsprachen, die vom verwendeten Browser lokal ausgeführt werden, ermittelt und dann zum Server gesendet werden.

Nach der grundlegenden Einführung in das Tracking im World Wide Web (Kapitel 2), wird in dieser Diplomarbeit das Konzept des Browser Fingerprintings vorgestellt und die Ermittlung von Merkmalen ausführlich beschrieben (Kapitel 3). Neben HTTP-Cookies und Fingerprints können auch weitere Verfahren zum Tracking von Nutzern eingesetzt werden, die in Kapitel 4 kurz angerissen werden. Im Rahmen der Diplomarbeit wurde eine ausführliche Feldstudie durchgeführt, bei der insgesamt 23.709 Fingerprints von unterschiedlichen Computern gesammelt wurden. Dabei wurden viele der vorgestellten Merkmale eingesetzt. Ebenso wurde geprüft, ob eine alternative Trackingmethode über Cache-Grafiken verlässlich funktioniert. Die ermittelten Daten der Studie werden in Kapitel 5 ausgewertet.

Generell versucht diese Diplomarbeit die Frage zu beantworten, ob unterschiedliche Teilnehmer auch unterschiedliche Browser Fingerprints liefern und sie darüber zu identifizieren sind. Es wird vorab die Vermutung aufgestellt, dass viele, wenn auch nicht alle, Browser Fingerprints eindeutig und in der

[3] EU-Richtlinie 2002/58/EG, Artikel 5, Absatz 3.
[4] Vgl. Zlotos: *Firefox 22 soll Tracking-Cookies unterdrücken*.

Stichprobenmenge einzigartig sind.

Hinweise zum Text

Der Text ist, sehr zum Bedauern des Autors, im generischen Maskulinum geschrieben. In Vorabversionen wurde von jedem Nomen und Pronomen die sowohl männliche als auch weibliche Version ausgeschrieben. Es entstanden insbesondere bei eingedeutschten Begriffen wie „Provider“, „Client“ aber auch bei sehr häufig auftretenden Begriffen wie „Nutzer“ oder „Drittanbieter“ äußerst unleserliche Konstrukte. Möglicherweise kann hier bei einer Aktualisierung des Dokuments eine bessere Lösung gefunden werden.

In dem Dokument werden sowohl Fuß- als auch Endnoten verwendet. Fußnoten sind durch eine hochgestellte Zahl zu erkennen (Beispiel^{0}), werden am Seitenende erläutert und enthalten entweder weitergehende Informationen oder Quellennachweise. Endnoten werden durch ein hochgestelltes Sternchen (Asterisk) und einer Zahl markiert (Beispiel^{*0}). Diese verweisen auf MySQL-Abfragen, die am Ende des Dokuments aufgelistet sind. Der interessierte Leser kann die Ergebnisse der Feldstudie damit selbst nachvollziehen.

Angaben in Prozent sind generell gerundet.

2 Tracking im World Wide Web

In diesem Abschnitt wird der Begriff des Trackings eingeführt, die handelnden Akteure beschrieben und das hauptsächlich benutzte Tracking per Cookies erläutert.

2.1 Relevante Akteure

Es existieren je nach Art des Trackings verschiedene Akteure. Die eingriffsintensivste Form des Trackings ist die komplette Überwachung des Internetverkehrs einer Person. Hier können aufgerufene Internetseiten protokolliert und der ausgetauschte Inhalt mitgeschnitten werden. Diese Form des Trackings steht in der Regel nur Strafverfolgungsbehörden in Form der Telekommunikationsüberwachung zur Verfügung.[5] Technisch hätten jedoch auch Internetzugangsanbieter (engl. *Access-Provider*)[6] die Möglichkeit, detaillierte Protokolle über die Internetaktivitäten aufzuzeichnen. Ebenso könnte dies auch mit kriminellen Hintergrund in Form von Schadsoftware auf Client-Computern oder aber durch das Mitschneiden des Datenverkehrs (sog. „Man-In-The-Middle"-Angriff) geschehen.

Diese Aspekte werden in der Arbeit jedoch nicht beleuchtet. Hier soll vor allem der Bezug zum Tracking aufgrund der Darstellung von (personalisierter) Werbung beleuchtet und entsprechende Akteure beschrieben werden.

2.1.1 Der Inhalteanbieter

Der Inhalteanbieter, engl. *Content-Provider*, ist „Anbieter eigener, selbst erstellter Inhalte".[7] Es handelt sich damit um die Person bzw. die Organisation,

[5] In Deutschland hauptsächlich über § 100a der Strafprozessordnung geregelt.
[6] Vgl. Uecker: „Host-Provider, Content-Provider, Access-Provider oder was?", S. 6.
[7] Vgl. ebd., S. 6.

welche die Inhalte in Form von Texten, Fotos, Videos oder anderweitiger Formate erstellt und öffentlich zugänglich macht.[8] Bei einer Nachrichtenseite ist dies beispielsweise eine Redaktion, bei einer privaten Homepage eine Einzelperson. Der Inhalteanbieter kann selbst vor allem lokales Tracking verwenden, um das Nutzungsverhalten der Benutzer auszuwerten. Eine häufig eingesetzte Software ist *Piwik*, welche die Aufrufe, Verweildauer, etc. eines Nutzers protokolliert.[9]

Ebenso kann der Inhalteanbieter zwecks Monetarisierung seiner Inhalte Werbeanzeigen schalten. Dazu ist er häufig auf Betreiber von Werbenetzwerken angewiesen, der dem Inhalteanbieter Werbebanner zur Verfügung stellt. Diese können dann auf der Website eingebunden werden.

2.1.2 Der Betreiber von Werbenetzwerken

Der Betreiber von Werbenetzwerken, vom Europäischer Gerichtshof (EuGH) als Internetreferenzierungsdienste bezeichnet,[10] stellt die technische Infrastruktur zur Verfügung, um Werbung auf Websites anbieten zu können. Dabei agiert er als Bindeglied zwischen dem Werbungtreibenden und dem Inhalteanbieter, auf dessen Website die Werbung erscheint. Der Betreiber von Werbenetzwerken erhält vom Werbungtreibenden die darzustellenden Inhalte und sorgt möglichst dafür, dass die Werbung bei Personen angezeigt wird, die einem bestimmten Profil entsprechen. Der Internetreferenzierungsdienst versucht daher u. a. über eine Chronik der besuchten Websites ein Interessenprofil des Nutzers zu erstellen.[11] Ebenso ist er dafür zuständig, entsprechende Klicks auf diese Werbebanner zu messen; u. U. auch Klicks, die zu einem direkten Onlinekauf führen. Dies

[8] Die technische Bereitstellung der Inhalte übernimmt hingegen der Host-Provider.

[9] Online verfügbar auf `http://piwik.org` (abgerufen am 10. Mai 2013).

[10] Vgl. LEHOFER: *Internetreferenzierungsdienst: EuGH beschreibt Google im Google-Urteil mit einem Wort, das Google (noch) nicht kennt* bzw. EuGH, Az. C-236/08.

[11] Möglicherweise werden, sofern verfügbar, auch weitere Informationen für das Profil hinzugezogen. So ist es technisch möglich, Suchanfragen oder Daten aus sozialen Netzwerken zu verknüpfen.

ist insbesondere relevant für sog. Affiliate-Netzwerke.[12]

2.1.3 Der Werbungtreibende

Der Werbungtreibende möchte sein Produkt möglichst effektiv auf Internetseiten platzieren. Neben Art, Größe und Aufruf- und Klickzahlen ist für ihn wichtig, dass die Werbung bei den Personen angezeigt wird, die tatsächlich Interesse an dem Produkt haben könnten. Die Rolle des Werbetreibenden ist für das Browser Fingerprinting nicht relevant.

2.1.4 Der Nutzer

Der Nutzer ruft Webinhalte vom Server des Inhalteanbieters ab. In dieser Diplomarbeit wird synonym der Begriff Client benutzt, der generell insbesondere die Kombination des verwendeten Computer mit dem benutzten Browser bezeichnet. Es ist tatsächlich durchaus möglich und üblich, dass sich mehrere Personen einen Computer und einen Nutzeraccount teilen. Durch die identische Konfiguration sind tatsächliche Personen daher nicht immer unterscheidbar. In einer in den USA durchgeführten Umfrage im Jahre 2012 haben 68 % der Teilnehmer angegeben, dass sie mit personalisiertem Tracking nicht einverstanden sind.[13] Dem entgegen steht eine von der BITKOM in Deutschland durchgeführten Umfrage, dass 64 % der deutschen Internetnutzer Werbeeinblendungen bevorzugen, wenn sie dadurch Geld sparen können.[14] Ob dies auch personalisierte Werbung einbezieht, die durch Tracking entsteht, ist jedoch nicht bekannt. Generell zeigen auch andere Studien, dass Nutzer (generelle) Werbung akzeptieren, ein Tracking selbst jedoch nicht wünschen, dies aber auch nicht

[12] Siehe Lammenett: *Praxiswissen Online-Marketing*, S. 39 ff.
[13] Vgl. AFP: *Americans not fans of online targeted ads: survey.*
[14] Vgl. Bundesverband Informationswirtschaft: *Internet: Mehrheit will lieber Werbung als Gebühren (Presseinformation).*

aktiv verhindern.[15]

2.2 Schutz von personenbeziehbaren Daten

Das Bundesdatenschutzgesetz (BDSG) soll „den Einzelnen davor [..] schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsrecht beeinträchtigt wird“.[16] Personenbezogene Daten dürfen in der Regel nur dann gespeichert werden, wenn der Nutzer explizit eingewilligt hat (Verbot mit Erlaubnisvorbehalt).[17] Damit Daten also unter das BDSG fallen, müssen diese personenbezogen sein, also einer bestimmten (natürlichen) Person zuordenbar sein. Wenn Daten mittelbar, somit durch Verknüpfung weiterer Informationen, einen Personenbezug ermöglichen, werden diese als personenbeziehbar bezeichnet.[18] Aber auch personenbeziehbare Daten unterliegen dem Schutz des BDSG.[19]

Obwohl es das Prinzip des Verbots mit Erlaubnisvorbehalt bei personenbezogenen Daten gibt, dürfen jedoch Nutzungsprofile zum „Zwecke der Werbung, der Marktforschung oder zur bedarfsgerechten Gestaltung“ pseudonymisiert angelegt werden. Der Nutzer hat jedoch die Möglichkeit, dem zu widersprechen (ein sog. „Opt-Out“). Ebenso dürfen diese Nutzungsprofile nicht mit Informationen über die tatsächliche Person zusammengeführt werden.[20]

---

[15] Siehe Beuth: *Der Privatsphäre-Revolution fehlen die Revolutionäre*.

[16] Vgl. § 1 Absatz 1 Bundesdatenschutzgesetz (BDSG).

[17] Dies ergibt sich aus § 14 Absatz 2 BDSG.

[18] Siehe Schwenke: „Datenschutzrechtliche Gestaltungsanforderungen an die Individualisierung“, S. 97.

[19] Siehe § 3 Absatz 1 BDSG.

[20] Vgl. § 15 Absatz 3 Telemediengesetz (TMG).

2.3 Klassische Trackingverfahren

Bei dem Tracking der Nutzer auf Webseiten wird hier vereinfacht zwischen zwei Kategorien unterschieden:

- **Lokales Tracking:** Bei dem lokalen Tracking wird das Verhalten eines Nutzers auf dem Webangebot eines Anbieters aufgezeichnet.
- **Globales Tracking:** Bei dem globalen Tracking wird das Verhalten eines Nutzers über mehrere Webangebote verschiedener Anbieter aufgezeichnet. Insbesondere soll ein Nutzer auf Webangebot Y wiedererkannt werden, der mehrere Tage zuvor auf den Seiten des Angebots X war. Hier wird in der Regel ein Drittanbieter (engl. *Third Party*) hinzugezogen, der sowohl auf X als auch Y eingebunden wird, für die Wiedererkennung der Nutzer zuständig ist und ggf. passende Werbung schaltet.

Die Diplomarbeit bezieht sich hauptsächlich auf das globale Tracking.

Üblicherweise erhält jeder Nutzer, sowohl beim lokalen als auch globalen Tracking, eine eindeutige Identifikation, die serverseitig gespeichert wird. Es kann sich dabei um eine Zahl, eine beliebige Zeichenfolge oder um einen eindeutigen Nutzernamen handeln. Im Laufe dieses Diplomarbeit wird diese Identifikation als UID oder *uid* (*User* oder *Unique Identification*) bezeichnet. Neben diesem Wert werden typischerweise weitere Informationen gespeichert, z. B. das Nutzungsverhalten oder bei Werbenetzwerken die Interessen und Vorlieben des Nutzers.

Damit bei einem erneuten Aufruf einer Website durch einen Nutzer wieder eine Zuordnung zur auf dem Server gespeicherten UID erfolgen kann, muss dieser Wert entweder übergeben oder ermittelt werden können. In den folgenden Abschnitten werden verschiedene Verfahren vorgestellt.

2.3.1 URL-Rewriting

Damit die UID über mehrere Seiten, sowohl lokal und in manchen Fällen auch global, weitergegeben werden kann, können Hyperlinks auf Webseiten um den UID-Wert als Parameter ergänzt werden. Aus

`http://server.tld/file`

wird dann beispielsweise

`http://server.tld/file?uid=12345`.

Die Übertragung der UID erfolgt hier relativ trivial. Es muss jedoch sichergestellt werden, dass tatsächlich jede Uniform Resource Locator (URL) die Identifizierungsinformation enthält und diese auch korrekt ist. Hier entsteht das größte Problem, da der Nutzer den Wert in der Adressleiste des Browsers ändern kann und somit keine erfolgreiche Zuordnung mehr stattfinden kann. Ebenso muss der Nutzer ausschließlich Hyperlinks bei der Navigation zwischen den Seiten verwenden.

Wird ein globales Tracking verwendet, müssen sich die unterschiedlichen Content-Provider auf einheitliche UIDs einigen. Wird hingegen ein Drittanbieter eingesetzt, muss die per Parameter übergebene UID-Information an den eingebetteten Inhalt des Drittanbieters übergeben werden.

2.3.2 Navigation mit HTML-Formularen

Ähnlich wie beim URL-Rewriting kann die Nutzerkennung auch durch in Formularen von Hypertext Markup Language (HTML) übertragen werden. Dabei muss die Navigation auf den Seiten ausschließlich über das Absenden von Formularen erfolgen. Der Vorteil ist dabei, dass die Daten, wenn gewünscht, nicht

als Parameter in der URL, sondern im Hauptteil der HTTP-Anfrage mitgeschickt werden können (sog. HTTP-POST statt HTTP-GET).[21] Dadurch kann der (durchschnittliche) Nutzer den zu übermittelnden Wert nicht ändern. Allerdings müssen die Formulardaten bei einem Neuladen einer Seite stets erneut mitgeschickt werden. Ebenso wie bei URL-Rewriting kann ein Tracking nicht stattfinden, wenn ein Nutzer eine Ressource direkt über die Adressleiste aufruft. Somit eignen sich beide Verfahren in der Praxis nur selten.

2.3.3 HTTP-Cookies

HTTP-Cookies, wie definiert in [RFC2109] und [RFC2965], sind Informationen in Textform, die ein Content-Provider bei einem Nutzer platzieren kann. Das Anlegen eines Cookies kann direkt über den HTTP-Header veranlasst werden, welches der Client empfängt (über den Steuerdateneintrag `Set-Cookie`).[22] Cookies können aber ebenso auch von clientseitigen Skriptsprachen wie JavaScript gesetzt und geändert werden (JavaScript: `document.cookie`). Es wird generell zwischen Sessioncookies, also Informationen, die nur für die aktuelle Browsersitzung verfügbar sind, und persistenten Cookies unterschieden.[23] Letztere können Tage, Wochen oder gar Jahre auf dem Rechner des Nutzers erhalten bleiben. Cookie-Informationen werden jeweils exklusiv für eine Domain abgelegt, können sogar nur auf bestimmte Pfade beschränkt werden. Websites von unterschiedlichen Domänen können nicht auf die jeweils anderen Cookies zugreifen. Ist auf dem Client ein HTTP-Cookie für die Domain `server.tld` hinterlegt, so werden bei jedem Aufruf von Ressourcen der Domain die Informationen im Header des HTTP-Requests mitgeschickt (im Header-Eintrag `Cookie`). Diese Daten werden jedoch nicht beim Aufruf von `server2.tld` angehängt und

[21] Siehe auch Berners-Lee: *Basic HTTP as defined in 1992*.
[22] Weitere Informationen zu Steuerdaten bzw. Header im folgenden Kapitel.
[23] Vgl. Rouse: *Transient cookie (session cookie)*.

können auch nicht durch clientseitige Skripte dieser Domain ausgelesen oder geändert werden.

Cookies können dafür verwendet werden, um Login-Daten eines Nutzers zu speichern. Da diese Informationen bei HTTP unverschlüsselt übertragen werden, können sie durch „Man-In-The-Middle-Angriffe“ abgefangen und die übertragenen Informationen missbräuchlich verwendet werden.[24] Eine Abhilfe kann hier Hypertext Transfer Protocol Secure (HTTPS) sein, welches Daten verschlüsselt überträgt.

Cookies können ebenso innerhalb einer Domain auch zum (lokalen) Tracking verwendet werden. Speichert eine Website eine UID auf dem Client ab, wird diese bei jedem erneuten Aufruf mitgesendet. So kann das Nutzerverhalten auf einem Webangebot nachvollzogen werden.

Browser können im Allgemeinen so konfiguriert werden, dass sie keine Cookies, alle Cookies oder alle außer Drittanbieter-Cookies akzeptieren. Bei Drittanbieter-Cookies – die offiziell erst in der neueren Spezifikation [RFC6265] eingeführt worden sind, aber bereits seit längerem von allen populären Browsern unterstützt werden – handelt es sich um Cookies, die von Websites von fremden Domains angelegt werden. Ruft der Nutzer Inhalte von `server.tld` auf und enthält die angeforderte Seite Inhalte (beispielsweise Grafiken) von `dritt.tld`, so können sowohl für `server.tld` als auch für `dritt.tld` unabhängige Cookies angelegt werden. Da in der angeforderten Ressource von `dritt.tld` in dem HTTP-Request-Headerfeld `Referer` die URL von der übergeordneten Seite (also von `server.tld`) erscheint, kann diese Information zum (globalen) Tracking verwendet werden. Hierbei legt `dritt.tld` ein Cookie mit einer UID auf dem Client ab. Wenn dieser dann `server2.tld` aufruft, welche ebenfalls eine Ressource von `dritt.tld` eingebunden hat, so wird die gespeicherte UID erneut übertragen. Es kann somit das Verhalten eines Nutzers nachvollzogen werden

[24] Siehe Böhm: *Fotofilter-App: Sicherheitslücke bei Instagram*.

– über alle Seiten, die Ressourcen von `dritt.tld` eingebunden haben.

3 Browser Fingerprinting

In diesem Kapitel wird das Verfahren des Browser Fingerprintings eingeführt und viele Merkmale, aus denen sich ein Fingerprint zusammensetzt, beschrieben. Dabei wird zwischen passivem und aktivem Fingerprinting unterschieden. Ebenso wird am Ende des Kapitels ein Überblick über bestehende Forschungsergebnisse gegeben.

3.1 Einführung

Dass bestimmte Eigenschaften eines verwendeten Geräts, auch unabhängig von der Nutzung im Internet, abgerufen werden, ist nicht neu. Software auf Desktop-Computern benutzen dieses Verfahren seit Jahren, um die rechtmäßige Verwendung zu überprüfen. So wurde mit Windows XP die Produktaktivierung eingeführt, bei der laut Microsoft u. a. der Prozessor-Typ und Informationen über die Grafikkarte und die Festplatte gesammelt werden, um daraus einen Fingerabdruck des Geräts zu erstellen.[25] Wird ein bereits verwendeter Produktschlüssel mit einer anderen Konfiguration verwendet, so ist möglicherweise eine erneute Aktivierung notwendig.

Fingerprints sind aber nicht nur zur Vermeidung von unrechtmäßiger Verbreitung und Nutzung von Software einsetzbar. Insbesondere die Werbebranche versucht in den letzten Jahren das Verfahren einzuführen und bezeichnet es teilweise als „die nächste Generation der Onlinewerbung“.[26] Obwohl der Datenaustausch im Netz durch bestimmte Protokolle standardisiert ist und die Annahme bestehen könnte, dass sich die Nutzer nur durch das Betriebssystem und den verwendeten Browser unterscheiden, gibt es viele verschiedene

[25] Siehe Microsoft Corporation: *Technical Details on Microsoft Product Activation for Windows XP*.

[26] Vgl. Angwin und Valentino-Devries: *Race Is On to 'Fingerprint' Phones, PCs*.

Merkmale, die von Client zu Client unterschiedlich sein können. Einige dieser Merkmale sind seit der Geburtsstunde des WWW abrufbar, andere sind in den letzten Jahren hinzugekommen. Außerdem lassen sich weitere Informationen über sog. „Hacks“ ermitteln, die vermutlich Sicherheitslücken in den Browsern geschuldet sind. Hier sei der „CSS-History-Hack“ erwähnt, bei dem ein Inhalteanbieter herausfinden konnte, welche (fremden) Websites der Nutzer zuvor aufgerufen hatte. Dabei wurden mehrere Hyperlinks auf einer Website zu verschiedenen Seiten dargestellt und durch ein Skript überprüft, ob diese z. B. blau (noch nicht besucht) oder rot (Nutzer hat die Seite bereits aufgerufen) dargestellt werden.[27] Dies ist in den meisten Browsern nicht mehr möglich.[28] Diese Informationen konnten aber als ein Merkmal für ein Browser Fingerprint verwendet werden.

Die folgenden Abschnitte stellen einige, zurzeit abrufbare, Merkmale dar. Passive Merkmale werden automatisch durch den Abruf einer Webressource mitgeschickt und liegen dem Server direkt vor. Aktive Merkmale müssen durch clientseitige Skriptsprachen (hier: JavaScript) oder Plugins (hier: Flash) ausgelesen und nachträglich zum Server übermittelt werden.

3.2 Passives Fingerprinting

Um Webinhalte auf einem Browser darstellen zu können, übernehmen sowohl Browser selbst, als auch das darunter liegende Betriebssystem, die Kommunikation mit dem Webserver, der die gewünschten Daten und Dokumente bereitstellt. Inhalte im WWW, die typischerweise in HTML beschrieben sind, werden über das Hypertext Transfer Protocol (HTTP) übertragen. HTTP ist dabei jedoch nur eins der möglichen Protokolle in der Anwendungsschicht des vierschichtigen Transmission Control Protocol / Internet Protocol (TCP/IP)

[27] Vgl. Grossman: *I know where you've been.*
[28] Siehe u. a. Kirsch: *Firefox-Entwickler stopfen altes CSS-Leck.*

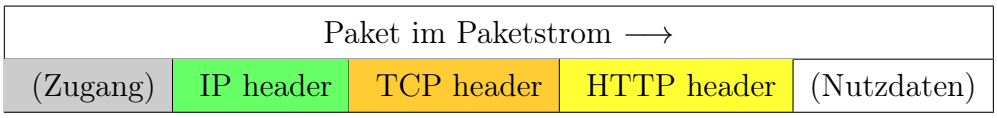
Referenzmodells.[29] Folgende Tabelle gibt einen Überblick über die vier Schichten und die für Webinhalte hauptsächlich benutzten Protokolle.[30]

TCP/IP-Schicht	Relevantes Protokoll für Browser-Fingerprints
Anwendung	HTTP oder auch HTTPS
Transport	Transmission Control Protocol (TCP)
Internet	Internet Protocol (IP)
Netzzugang	(Nicht relevant, z. B. Ethernet)

Im Internet ist der Datenaustausch paketbasiert, d. h. die Kommunikation von einem Gerät zu einem anderen erfolgt nicht über eine feste Leistung und einem gleichbleibendem Weg, sondern durch Aufteilung der Daten in mehrere kleine Pakete, die unabhängig voneinander u. U. über verschiedene Routen versendet werden. Jedes der zu übermittelnden HTTP-Pakete ist in einem Paket der nächsthöheren Schicht gekapselt. Dabei besteht jedes einzelne Paket in jeder Schicht aus Steuer- bzw. Kopfdaten (engl. *header*) und aus Nutzdaten (engl. *payload* oder schlicht *data*), in denen dann das Paket der nächsthöheren Schicht bzw. bei HTTP der tatsächliche Inhalt integriert ist. Betrachtet man das Schichtenmodell im Paketstrom ergibt sich folgende Darstellung:

[29] Das TCP/IP-Schichtenmodell basiert auf dem siebenschichtigem Open Systems Interconnection Reference Model (OSI) Referenzmodell, das zwar ein standardisiertes Designmodell für Kommunikationsprotokolle ist, jedoch praktisch nicht verwendet wird. Mehr dazu in Tanenbaum: *Computernetzwerke*, S. 30 ff.

[30] Es existieren weitere Protokolle für die einzelnen Schichten. Bei dem Einsatz von Webdokumenten kommen typischerweise aber die gelisteten Protokolle zum Einsatz, vgl. dazu [RFC1945], S. 7.



Für ein Browser-Fingerprint können Informationen, die in den einzelnen Steuerdaten vom Clientcomputer zum Server übermittelt werden, interessant sein. Da diese Kopfdaten immer übermittelt werden, kann von einem passiven Fingerprinting gesprochen werden, da auf der Seite des Clients kein spezielles Programm oder Skript gestartet werden muss, um aktiv Daten abzufragen. Wenn Einträge der Steuerdaten für ein Fingerprinting o. ä. archiviert werden, so geschieht dies, ohne dass der Client dies bemerkt.[31] Folgende Tabelle stellt eine Auswahl an potentiell aussagekräftigen Kopfdaten dar:

Bezeichnung	Zugehörige Schicht
Quell-IP-Adresse	IP
Quellport	TCP
Aufrufende Seite („Referer“ [sic!])	HTTP
Bezeichnung des Browsers („User-Agent“)	HTTP
Akzeptierende Dateitypen	HTTP
Akzeptierende Zeichensätze	HTTP
Akzeptierende Kompressionsformate	HTTP
Akzeptierende Sprachen	HTTP

[31] Zwar wird der aktive Fingerprint vermutlich von den meisten Anwendern ebenso nicht bemerkt, dennoch könnte dieser im HTML- bzw. JavaScript-Quellcode auf dem Clientcomputer oder durch Analyse der ausgehenden Datenpakete nachgewiesen werden.

3.2.1 IP-Adresse

IP-Adressen werden in Netzwerken, die auf dem IP-Protokoll basieren (somit also auch im Internet), verwendet, um den Datenaustausch zwischen zwei oder mehreren Rechnern zu ermöglichen. Durch die IP-Adresse können Geräte einzeln oder – seltener und in dem Fall des Browser Fingerprintings nicht von Bedeutung: – in Gruppen (Multicast) angesprochen werden. Der Client-Computer, der eine Website besucht, übersendet in seiner Datenanfrage (engl. *request*) – bedingt durch den TCP/IP-Protokollstack – seine IP-Adresse. Die IP-Adresse des Clients wird ebenso wie die IP-Adresse des Servers im Kopfbereich des Datenpakets in der IP-Schicht vermerkt. Bei den Paketen, die der Server als Antwort (engl. *response*) an den Client zurücksendet, werden Quell- und Ziel-IP-Adresse lediglich getauscht. Unter der theoretischen Annahme, dass jedes Gerät eine feste IP-Adresse besitzt und jede IP-Adresse genau einem Gerät zugeordnet werden kann, wäre diese ein sehr aussagekräftiges Merkmal für einen Browser Fingerprint. In der Praxis existieren aber zwei elementare Einschränkungen:

(a) **Dynamische IP-Adressen:** Im Endanwenderbereich erhalten Kunden keine feste und dauerhafte IP-Adresse. Stattdessen besitzt der Access-Provider [32] einen IP-Adresspool, aus dem heraus die Kunden momentan verfügbare Adressen erhalten. Der Anbieter benötigt daher weniger Adressen als Kunden, da die Wahrscheinlichkeit, dass alle Kunden gleichzeitig online sind, gering ist. Für den Kunden ergeben sich dadurch Vorteile im Datenschutz: Durch die stetige Neuvergabe der IP-Adressen kann nicht von einer Adresse auf ein Gerät, geschweige denn einen Nutzer, ge-

[32] Der Access-Provider oder Zugangsanbieter ermöglicht den Zugang zum Internet und kann dabei entweder über eigene Netzinfrastruktur verfügen oder Netze eines anderen Anbieters nutzen. Typische Beispiele in Deutschland sind die Telekom, 1&1 und Unitymedia, vgl. Uecker: „Host-Provider, Content-Provider, Access-Provider oder was?“

schlossen werden. Dies ist nur in der Zeit möglich, in der die Adresse noch nicht erneut vergeben wurde (typischerweise mehrere Stunden). Ob und wann dem Client eine neue IP-Adresse zugeordnet wurde, ist nur dem Access-Provider und seinem jeweiligen Kunden bekannt.[33]

(b) **Network Address Translation (NAT):** Bei NAT im Bezug zum Internet wird eine (öffentliche) IP-Adresse von mehreren Geräten geteilt, d. h. mehrere Client-Computer sind unter einer Adresse ansprechbar.[34] Dies tritt z. B. auf, wenn in einem Haushalt ein privates Local Area Network (LAN) betrieben wird, das gebündelt über einen Router mit dem Internet verbunden ist. Alle Geräte dieses Netzwerks verwenden bei dem Aufruf von Internetinhalten die gemeinsame IP-Adresse. Bei jeder TCP/IP-Verbindung setzt der heimische Router die Port-Nummer fest (siehe Abschnitt 3.2.2, Seite 31) und kann damit die zu empfangenden Antwortpakete im Heimnetzwerk korrekt zuordnen. Auch bei Access Providern findet teilweise, insbesondere im Bereich des Mobilfunks, ein NAT statt. Dadurch können sich u. a. Smartphones von sich völlig unbekannten Personen dieselbe IP-Adresse teilen.[35]

Diese zwei Eigenschaften der Adressvergabe beruhen mitunter auf der Knappheit der verfügbaren IPv4-Adressen, der (noch) aktuellen und hauptsächlichen Adressierungsart im Internet.[36] Durch die zukünftige Etablierung des Nachfolgestandards IPv6 sind theoretisch statt lediglich $2^{32} \approx 4,29 \cdot 10^6$ (IPv4) sogar

[33] Vgl. Mahlmann und Schindelhauer: „Das Internet — unter dem Overlay“, S. 49.

[34] Serverseitig kann auch eine Internetpräsenz intern von mehreren unterschiedliche Servern unter einer öffentlichen IP-Adresse betreiben werden. Für das Browser Fingerprinting ist dies jedoch nicht relevant.

[35] Vgl. Zivadinovic: *Tunnelbau – Router und Smartphones im VPN-Zusammenspiel*, Abschnitt „NAT-Probleme“; sowie Ford, Srisuresh und Kegel: *Peer-to-Peer Communication Across Network Address Translators*.

[36] Vgl. Ihlenfeld: *IPv4: Die letzten IP-Adressen in Europa werden vergeben.*

$2^{128} \approx 3,4 \cdot 10^{38}$ (IPv6) Adressen möglich.[37] Sowohl NAT als auch dynamische IP-Adressen sind daher nicht mehr notwendig, wenn lediglich die zur Verfügung stehenden IP-Adressen betrachtet werden. Für dynamische Adressen auch bei IPv6 spricht vor allem der Datenschutz und auch die Verhinderung des Access-Provider, dass Kunden nicht auf einfache Weise ihren Computer als Internetserver verwenden können.[38] Da der hintere Teil der IPv6-Adresse gerätespezifisch definiert wird (sog. *interface identifier*), sollte aus Gründen des Datenschutzes das verwendete Betriebssystem diesen Teil nicht aus dem Netzwerk-Interface herleiten, sondern zufällig generieren (dies wird als Privacy Extensions bezeichnet).[39] Auch NAT ist aus Sicht der Adressknappheit nicht mehr notwendig (ein Argument für IPv6 ist sogar, dass NAT überwunden werden kann, um eine klare Ende-zu-Ende-Kommunikation ohne Adressumwandlung sicherstellen zu können). Dennoch gibt es auch hier mögliche Formen der Umsetzung. So ist im Abstract von [RFC4864] zu lesen:

> Although there are many perceived benefits to Network Address Translation (NAT), its primary benefit of "amplifying" available address space is not needed in IPv6. In addition to NAT's many serious disadvantages, there is a perception that other benefits exist, such as a variety of management and security attributes that could be useful for an Internet Protocol site. IPv6 was designed with the intention of making NAT unnecessary, and this document shows how Local Network Protection (LNP) using IPv6 can provide the same or more benefits without the need for address translation.

Ob und wie dynamische Adressen und NAT bei IPv6 praktisch eingesetzt werden, ist momentan noch unklar. Die Aussagekraft von Browser Fingerprints

[37] Siehe [RFC2460].

[38] Die Deutsche Telekom verteilt seit September 2012 dynamische IPv6-Adressen an Privatkunden. Allerdings muss eine neue Adresse manuell angefordert werden (z. B. durch einen Neustart des Routers) und wird nicht nach einer bestimmten Zeit erneuert, siehe Ihlenfeld: „IPv6: Telekom verteilt IPv6-Adressen an Kunden"

[39] Vgl. [RFC4941].

mit Bezug zu IP-Adressen ist momentan gering, könnte aber ggf. in Zukunft zunehmen.

3.2.2 TCP-Quellport

Quell- und Zielport sind zwei Angaben im Kopfbereich des TCP-Segments. Das Betriebssystem des Geräts, das ein Datenpaket erhält, kann durch die Angabe im Zielport erkennen, an welche Anwendung die erhaltenen Daten weitergeleitet werden sollen. Im Falle von NAT werden die Portnummern zur Zuordnung der Geräte innerhalb des lokalen Netzwerks verwendet. Bei einem ausgehenden Paket stehen die vom Betriebssystem oder Router generierte Quellportangaben im TCP-Header; augenscheinlich sind dies willkürliche generierte Zahlen (nach [RFC6335], S. 10, Portnummern ≥ 1024). Zur Kommunikation mit bestimmten Diensten auf Servern gibt es jedoch fest vereinbarte Standardportnummern, die bei einem an den Server gehenden Paket als Zielport notiert werden (u a. Port 80 für HTTP / Webserver, siehe [RFC6335], S.6). Da für das Browser Fingerprinting nur Client-Quellports in Frage kommen, diese aber zufällig generiert werden, sind Portangaben vermutlich ohne große Relevanz.

3.2.3 HTTP-Kopfdaten

In den Steuerdaten des HTTP-Segments sind mehrere Einträge vorhanden, die für ein Browser Fingerprinting relevant sein können. Anders als bei den IP- und TCP-Kopfdaten hat der HTTP-Header keine feste Größe und kann auch benutzerdefinierte Einträge enthalten. Dennoch sind eine Vielzahl von standardisierten Einträge definiert.[40] Die Steuerdaten werden in der HTTP-Anfrage bzw. -Antwort der Nutzdaten als einfacher Text vorangestellt. Der Name und Wert des Kopfdatums wird durch einen Doppelpunkt, die gesamten Steuerda-

[40] Siehe hierzu [RFC2616], S. 100 und [RFC4229].

ten an sich durch Zeilenumbrüche getrennt. Zwischen Kopf- und Nutzdaten befindet sich eine leere Zeile.[41]

Folgende Kopfdaten können für einen Browser Fingerprint von Relevanz sein:

- **Aufrufende Seite („Referer“ [sic!])**
  Als Referer kann vom Webbrowser die Website angegeben sein, von der der Besucher auf die zu ladende Seite gelangt. Gelangt der Benutzer von Seite A durch einen Hyperlink auf Seite B, wird in der HTTP-Anfrage an den Server von Seite B die URL von Seite A als Referer übertragen.[42] Da bestimmte Nutzer u. U. immer von einer bestimmten Seite auf die Zielseite gelangen oder aber auch die URL-Angabe im Referer-Feld auch sog. *GET-Parameter* enthalten kann, kann dadurch eine Identifizierung bei einem Browser Fingerprint gegeben sein.[43]

- **Bezeichnung des Browsers („User-Agent")**
  Bei einer HTTP-Anfrage sendet der Webbrowser typischerweise seine Bezeichnung, Versions- und Plattforminformationen mit. Diese Informationen befinden sich im Eintrag „User-Agent“. Eine exakte Beschreibung des Formats der Zeichenfolge gibt es nicht, jedoch ähneln sich die Angaben bei unterschiedlichen Browsern.[44] [45] [46] Beispiel: *Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17.*

[41] Vgl. [RFC2616], S. 31 ff.

[42] Vgl. [RFC2616], S. 140 f. (Die korrekte englische Schreibweise wäre „Referrer“, jedoch hat sich der Tippfehler in die Festlegung des Standards geschlichen.)

[43] Als GET-Parameter werden die Angaben in einer URL nach einem Fragezeichen (?) bezeichnet.

[44] Vgl. [RFC2616].

[45] Eine humorvoll geschriebene Geschichte der Entwicklung der User-Agent-Zeichenfolge ist in Andersen: „History of the browser user-agent string“ zu lesen.

[46] Unter `http://www.useragentstring.com` können User-Agent-Zeichenfolgen analysiert werden (zuletzt am 6. Februar 2013 auf Verfügbarkeit geprüft).

- **Akzeptierende Dateitypen („Accept“)**
  In einem HTTP-Request sendet der Browser unter der Bezeichnung „Accept“ die von ihm akzeptierten Antwortformate zurück. Dabei handelt es sich meist um HTML, Extensible Hypertext Markup Language (XHTML) bzw. Extensible Markup Language (XML). Ebenso kann spezifiziert werden, welches Rückgabeformat präferiert wird. Genauere Informationen finden sich in [RFC2616], S. 100. Beispiel: *text/html,application/xhtml +xml,application/xml;q=0.9,*/*;q=0.8.*

- **Akzeptierende Zeichensätze („Accept-Charset“)**
  Der Browser kann ebenso definieren, welche Zeichencodierung er als Antwort vom Webserver wünscht und Prioritäten festlegen. Mögliche Zeichensätze sind u. a. UTF-8 oder ISO-8859-1.[47] Viele moderne Browser übertragen dieses Steuerdatum aber nicht mehr. Beispiel: *ISO-8859-1,utf-8;q=0.7,*;q=0.3.*

- **Akzeptierende Kompressionsformate („Accept-Encoding")**
  Um die Übertragungsgröße zu minimieren, können Webserver die Ausgabe vor dem Versand komprimieren. Der Client-Browser muss diese komprimierten Daten folglich auch entpacken können. Über das Headerfeld „Accept-Encoding“ kann der Browser mitteilen, welche Kompressionsformate er unterstützt.[48] Beispiel: *gzip, deflate.*

- **Akzeptierende Sprachen („Accept-Language“)**
  Falls ein Dokument in mehreren Sprachen vorliegt, kann dieses Steuerdatum dazu verwendet werden, die gewünschte Sprache des Benutzers auszugeben. Browser übertragen hier die aktuell verwendete Sprache des Browsers bzw. des Betriebssystems. Einige Browser erlauben auch die An-

[47] Vgl. [RFC2616], S. 102.
[48] Vgl. [RFC2616], S. 102.

gabe weiterer Sprachwünsche in den Einstellungen. Die Sprachen müssen wie in [RFC4646] spezifiziert angegeben werden und können auch gewichtet werden.[49] Beispiel: *de-de,de;q=0.8,en-us;q=0.5,en;q=0.3*.

3.3 Aktives Fingerprinting mit Javascript

Bei aktivem Fingerprinting müssen Daten und Eigenschaften abgefragt werden, die nicht direkt durch den Aufruf einer Internetressource mitgeschickt werden. Dies sind daher keine Merkmale, die sich aus dem TCP/IP-Protokollstack ergeben. Im Gegensatz zu passivem Fingerprinting muss auf der Seite des Clients eine Abfrage dieser Eigenschaften stattfinden. Dies kann z. B. durch JavaSript oder Plugins (Erweiterungen der Browserfunktionalität) erfolgen.

3.3.1 Vorbemerkung zu Fingerprints über JavaScript

Die folgenden Abschnitte beschreiben das Abrufen bestimmter Browser- oder Systemeigenschaften über JavaScript. Es existieren noch weitere Eigenschaften, die möglicherweise für ein Browser Fingerprint von Interesse sein könnten. Insbesondere bei der Abfrage von Plugin-Informationen kann hier nur eine Auswahl vorgestellt werden. Zum Verständnis sind Grundlagenkenntnisse des HTML Document Object Model (DOM) und von JavaScript erforderlich. Die Beispiele können mit einem leeren, wohlgeformten HTML-Dokument nachvollzogen werden:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Browser Fingerprinting</title>
```

[49] Vgl. [RFC2616], S. 104.

```
5     <script type="text/javascript">
6
7     </script>
8   </head>
9   <body>
10  </body>
11  </html>
```

Listing 1: Leeres HTML5-Dokument mit leerem JavaScript-Bereich (HTML)

Wenn nicht anders angegeben, soll der Code zwischen den Tags `<script>` und `</script>` eingefügt werden. In den vorgestellten Codebeispielen werden die ermittelten Werte, falls möglich, als Dialogfenster dargestellt (JavaScript: `alert()`). Sollte ein Browser Fingerprint erstellt werden sollen, wäre es sinnvoller, diese Daten an den Server zu übermitteln. Hier kommen Techniken wie Asynchronous JavaScript and XML (AJAX) zum Einsatz, die aber hier nicht weiter vertieft werden können.

3.3.2 Browserinformationen

Grundlegende Informationen über den vom Nutzer verwendeten Browser (und auch dem Betriebssystem) lassen sich über Eigenschaften des `navigator`-Objekts abfragen, welches wiederum dem `window`-Objekts angehört. Eine Auswahl an bestimmten Eigenschaften:

- `document.referrer`
  Enthält die URL der aufgerufenen Seite oder ist leer, falls die Seite direkt aufgerufen wurde. Entspricht damit dem „Referer“, der bereits beim passiven Fingerprinting benutzt wurde. Definiert in Le Hégaret, Le Hors und Stenback: *Document Object Model (DOM) Level 2 HTML Specification*. Beispiel: *http://example.org*.

- `navigator.appCodeName`

  Gibt die interne Bezeichnung des verwendeten Browsers zurück (meist „Mozilla“). Ist in keiner offiziellen Spezifikation notiert, wird aber u. a. in Mozilla Developer Network: *window.navigator.appCodeName* erwähnt. Beispiel: *Mozilla*.

- `navigator.appName`

  Gibt den Namen des Browser zurück. Wird bereits seit Jahren verwendet, ist offiziell aber das erste Mal in Hyatt und Hickson: *HTML 5* definiert. Beispiel: *Microsoft Internet Explorer*.

- `navigator.appVersion`

  Anders als die Bezeichnung der Eigenschaft vermuten lässt, werden hier nicht nur die Browser-Version, sondern u. a. auch weitergehende Informationen über das Betriebssystem oder gar den Prozessortyp zurückgegeben. Es existiert kein einheitliches Format und ähnelt häufig der Angabe des „User-Agent“ (HTTP-Kopfdaten). Die Eigenschaft ist seit Jahren verfügbar, findet aber das erste Mal in ebd. eine erste offizielle Definition. Beispiele: *5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17.* oder *Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0*

- `navigator.cookieEnabled`

  Gibt `true` zurück, falls der Browser das Ablegen von Cookies unterstützt und diese nicht vom Benutzer deaktiviert worden sind. Ist nicht offiziell definiert, wird aber praktisch von jedem Browser unterstützt (vgl. Mozilla Developer Network: *window.navigator.cookieEnabled*).

- `navigator.language`

  Gibt die vom Nutzer gewünschte Sprache an, in der Websites erscheinen sollen. Diese muss wie in [RFC4646] spezifiziert angegeben werden. Dabei

kann nur ein Sprachcode oder ein Sprach- und Regionalcode angegeben werden. Offiziell nicht spezifiziert, aber beschrieben in Mozilla Developer Network: *window.navigator.language*. Beispiele: *en-US* oder *de*.

- `navigator.platform`
  Gibt die vom Benutzer verwendete Plattform zurück. Offiziell definiert in Hyatt und Hickson: *HTML 5*, aber bereits länger verfügbar. Beispiele: *Win32* oder *MacIntel*.

- `navigator.userAgent`
  Gibt eine Kennung des verwendeten Browsers zurück. Dabei verwenden unterschiedliche Browser verschiedene Formate, die meist auch Informationen über das Betriebssystem des Nutzers enthalten. Diese Angabe ähnelt häufig der Angabe des „User-Agent“ (HTTP-Kopfdaten). Die Eigenschaft ist seit Jahren verfügbar, findet aber das erste Mal in ebd. eine erste offizielle Definition. Beispiel: *Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17*.

3.3.3 Bildschirminformationen

Informationen über den Bildschirm des Nutzers sind teilweise als direkte Eigenschaften des `screen`-Objekts abrufbar (ebenfalls Unterobjekt von `window`). Dies gilt für:

- `screen.width`
  Gibt die Breite des Bildschirms in Pixel zurück. Beim Microsoft Internet Explorer ist eine verlässliche Angabe nur bei einer Zoomstufe von 100 % zu erwarten (siehe Mozilla Developer Network: *window.screen.width*). Beispiel: *1024*.

- `screen.height`
  Gibt die Höhe des Bildschirms in Pixel zurück. Mögliche Einschränkungen sind bei Mozilla Developer Network: *window.screen.height* beschrieben. Beispiel: *768*.

- `screen.availWidth`
  Gibt die Anzahl der Pixel zurück, die für das aktuelle Fenster für die Breite zur Verfügung stehen. Weitere Informationen sind unter Mozilla Developer Network: *window.screen.availWidth* nachzulesen. Beispiel: *1024* (wenn das Fenster bei einer Bildschirmbreite von 1024 maximiert dargestellt wird).

- `screen.availHeight`
  Analog zu `screen.availHeight`, siehe auch Mozilla Developer Network: *window.screen.availHeight*. Beispiel: *940* (bedingt durch Taskleiste oder Dock).

- `screen.colorDepth`
  Gibt die Farbtiefe in Bits zurück, siehe auch: Mozilla Developer Network: *window.screen.colorDepth*. In manchen Browsern heißt die Eigenschaft `screen.pixelDepth`. Beispiel: *24*.

Moderne Webbrowser mit der WebKit-Engine unterstützen die Eigenschaft `window.devicePixelRatio`, die jedoch verlässlich nur unter Safari zu funktionieren scheint.[50] Die Eigenschaft gibt den Unterschied zwischen den physikalisch vorhanden und der geräteunabhängigen Pixel an.[51] Das Apple iPhone 4 hat eine physikalische Auflösung von 640x960 Pixel, obwohl tatsächlich nur Inhalte dargestellt werden, die typischerweise bei einer Auflösung von 320x480

[50] Vgl. Koch: *devicePixelRatio*.
[51] Die Bezeichnung Device Independent Pixels (DIPS) ist ebenfalls geläufig.

sichtbar sind. Die Pixeldichte ist also vervierfacht: sowohl in der Breite als auch Höhe wurde diese verdoppelt, `window.device.pixelRatio` hat hier somit den Wert 2.

3.3.4 Zeitzone

Über JavaSript lässt sich die Zeitzone ermitteln, in der sich der Client-Computer befindet. Dabei werden die Einstellungen des Betriebssystems abgefragt. Dadurch lässt sich ein ungefährer Aufenthaltsort des Computers ermitteln.

```
1 var d = new Date();
2 alert(d.getTimezoneOffset());
```

Listing 2: Abfrage der Zeitzone (JavaScript)

`getTimezoneOffset()` liefert den Unterschied zwischen der lokalen Zeit und der Greenwich Meridian Time (GMT) in Minuten zurück. Im Winter liefert die Funktion in der Zeitzone für Berlin z. B. -60 zurück.

3.3.5 Abfragen der Systemfarben

Betriebssysteme verwenden bestimmte Systemfarben, die die Optik der Standardkomponenten, wie z. B. Schaltflächen oder Fenster beeinflussen. Das World Wide Web Consortium (W3C) hat mit Cascading Style Sheets (CSS) Level 2 (1998) die Möglichkeit geschaffen, Elemente einer Website mit Systemfarben zu versehen.[52] So kann beispielsweise ein über HTML und CSS gestaltetes Fenster die Farben annehmen, die der Nutzer von seinem Betriebssystem gewohnt ist. Über die Funktion `getComputedStyle()` kann dann der tatsächliche Farbwert

[52] Çelik et al.: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*.

ermittelt werden.[53] CSS bietet insgesamt 28 Systemfarben zur Abfrage an, eine Auswahl zeigt die folgende Tabelle.[54]

`ActiveBorder`	Farbe des Rahmens des aktiven Fensters
`ActiveCaption`	Farbe des Titeltextes des aktiven Fensters
`Background`	Farbe des Desktophintergrunds
`ButtonText`	Die Farbe des Textes auf Schaltflächen
`ThreeDHighlight`	Farbe des helleren Randes an der Seite der Lichtquelle für 3D-Elemente

```
1  var farbe = 'ButtonText';  // Systemfarbe
2  var c;
3
4  var div = document.createElement('div');
5  div.style.backgroundColor = farbe;
6  document.getElementsByTagName('body')[0].appendChild(div);
7
8  if (div.currentStyle) {
9    c = div.currentStyle['backgroundColor'];
10 } else {
11   c = document.defaultView.getComputedStyle(div, null).
       getPropertyValue('background-color');
12 }
13
```

[53] Ob dies tatsächlich eine gewollte Funktionalität ist oder ob ein Zugriff darauf nicht vom Browser unterbunden werden sollte, ist nicht klar. Im Microsoft Internet Explorer ist solch ein Zugriff nicht möglich – hier wird statt eines Farbwertes die Bezeichnung der Systemfarbe (z. B. „ButtonText“) zurückgegeben.

[54] Weitere siehe Çelik et al.: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*.

```
14 alert(c);
```

Listing 3: Abfrage der Systemfarben am Beispiel „ButtonText“ (JavaScript)

Erläuterung: In Zeile 1 kann eine beliebige, vom W3C vorgegebene, Bezeichnung für eine Systemfarbe eingetragen werden. Im Anschluss wird ein leeres `<div />`-Element mit der gewünschten Hintergrundfarbe erstellt. Je nach Browser wird dann über eine entsprechende Funktion die tatsächlich dargestellte Farbe ermittelt.

3.4 Aktives Fingerprinting: Browsererweiterungen

Die Funktionalität von Internetbrowsern kann durch Erweiterungen, sog. Plugins, erhöht werden. Bestimmte Inhalte auf Websites erfordern solche Plugins, um u. a. Videos wiedergeben zu können. Mit JavaScript kann geprüft werden, ob und welche Plugins auf dem Client installiert sind.

3.4.1 Adobe Flash

Die Fähigkeiten eines Browsers können mit Plugins erweitert werden. So ist Adobe Flash eines der bekanntesten und weit verbreitetsten Plugins; nach eigenen Angaben wird es auf 99 % der verwendeten Desktop-Computer und Notebooks verwendet.[55] Im Mobilbereich ist der Anteil deutlich geringer, da u a. Apple in seinem mobilen Betriebssystem iOS auf die Verwendung von Adobe Flash verzichtet.[56] Ob das Adobe Flash Plugin verfügbar ist (und falls ja, in welche Version) kann also ein wichtiges Merkmal für den Browser Fingerprint werden:

```
1 try {
```

[55] Adobe: *Adobe Flash Platform runtimes*.
[56] Jobs: *Thoughts on Flash*.

```
2   var obj = new ActiveXObject('ShockwaveFlash.ShockwaveFlash.6'
        );
3   alert(new ActiveXObject('ShockwaveFlash.ShockwaveFlash').
        GetVariable('$version').replace(/\D+/g, '.').match
        (/^.?(.+),?$/)[1]);
4 } catch(e) {
5   try {
6     if(navigator.mimeTypes["application/x-shockwave-flash"].
          enabledPlugin) {
7       alert((navigator.plugins["Shockwave Flash 2.0"] ||
            navigator.plugins["Shockwave Flash"]).description.
            replace(/\D+/g, ".").match(/^.?(.+),?\$/)[1]);
8     }
9   } catch(e) {}
10 }
```

Listing 4: Ermitteln der Adobe Flash Plugin Version (JavaScript)

Erläuterung: In den ersten drei Zeilen wird versucht, ein ActiveX-Objekt des Flash-Plugins zu erstellen und von dieser die aktuelle Version zu ermitteln. Sollte der Versuch fehlschlagen, wird das Array `navigator.mimeTypes` durchlaufen, um zu prüfen, ob für den entsprechenden Multipurpose Internet Mail Extensions (MIME) Typ ein Plugin vorhanden ist. Falls ja, wird die entsprechende Version des Plugins (*Shockwave Flash 2.0* oder *Shockwave Flash*) zurückgegeben.[57]

3.4.2 Microsoft Silverlight

Ein Plugin mit ähnlicher Funktionalität ist Microsoft Silverlight. Die Verbreitung von Silverlight hat zwar in den letzten Jahren zugenommen, liegt aber

[57] Der Code basiert auf Sarvell: *Detecting Flash player version with JavaScript*.

hinter der von Flash.[58] Die generelle Zukunft von Silverlight ist ungewiss.[59] Dennoch kann eine Versionsabfrage für ein Fingerprint sinnvoll sein:

```
1  if (window.ActiveXObject) {
2    try {
3      var obj = new ActiveXObject('AgControl.AgControl');
4      var v = new Array('5.1.10411.0', '5.0.61118.0', '
           5.0.60818.0');
5      var i = -1;
6      var b = false;
7
8      do {
9        i++;
10       b = obj.isVersionSupported(v[i]);
11     } while (!b && i < v.length);
12
13     if (b) {
14       alert(v[i]);
15     }
16   } catch (e) {}
17 } else {
18   var b = false;
19   for (var i = 0; i < navigator.plugins.length; i++) {
20     if (navigator.plugins[i].name.indexOf('Silverlight') != -1)
           {
21       alert(navigator.plugins[i].description);
22       b = true;
23     }
24   }
25 }
```

[58] StatOwl: *Microsoft Silverlight Version Support*.
[59] Siehe Neumann: *Freier Silverlight-Klon Moonlight eingestellt*.

Listing 5: Ermitteln der Microsoft Silverlight Version (JavaScript)

Erläuterung: Falls der Browser ActiveX-Komponenten unterstützt, wird versucht ein Flash-Objekt zu instanziieren. In dem Array `v` werden alle bis jetzt veröffentlichten Versionen aufgelistet.[60] Die Funktion `isVersionSupported` liefert einen booleschen Wert, ob eine entsprechende Version von dem aktuell installierten Plugin unterstützt wird.[61] Falls die Nutzung von ActiveX-Komponenten nicht möglich ist, wird `navigator.plugins` durchlaufen, bis das Silverlight-Plugin gefunden wurde. Die Versionsangabe befindet sich direkt in der Beschreibung des Plugins.

3.4.3 Adobe Reader

Die am weitesten vebreitete Erweiterung zur Anzeige von Portable Document Format (PDF) Dateien ist der Adobe Reader (früher: Acrobat Reader). Die hohe Verbreitungsrate des Adobe Reader ist auch durch seine offenbar erscheinende Monopolstellung begründet. So wird auf Seiten, die PDF-Dokumente enthalten, fast ausschließlich auf den Adobe Reader verwiesen, falls der Nutzer die Inhalte nicht darstellen kann.[62]

```
1 if (window.ActiveXObject) {
2   var obj = null;
3   try {
4     obj = new ActiveXObject('AcroPDF.PDF');
```

[60] Aus Platzgründen hier nur die letzten drei, eine komplette Liste findet sich auf `http://www.microsoft.com/en-us/download/details.aspx?id=12121` (geprüft am 26. Januar 2013).

[61] Vgl. Microsoft Developer Network (MSDN): *IsVersionSupprted*.

[62] Vgl. Sawall: „FSFE findet 2.160-mal Werbung für Adobe Reader“.

```
5    } catch(e) {}
6
7    if (!obj) {
8      try {
9        obj = new ActiveXObject('PDF.PdfCtrl');
10     } catch (e) {}
11   }
12
13   if (obj) {
14     var version = obj.GetVersions().split(',');
15     version = version[0].split('=');
16     version = parseFloat(version[1]);
17     alert(version);
18   }
19 } else {
20   for (var i = 0; i < navigator.plugins.length; i++) {
21     if (navigator.plugins[i].name.indexOf('Adobe Acrobat') !=
         -1) {
22       alert(navigator.plugins[i].description.replace(/\D+/g, ".
           ").match(/^.?(.+),?$/)[1]);
23     }
24   }
25 }
```

Listing 6: Ermitteln der Adobe Reader Version (JavaScript)

Erläuterung: Wie auch in den vorigen Fällen wird zwischen der Abfrage über ActiveX-Komponenten oder über das `navigator.plugins`-Array unterschieden. Im ersten Fall wird versucht, eine ActiveX-Instanz zu erstellen (dabei ist „AcroPDF.PDF“ die aktuelle und „PDF.PdfCtrl“ eine alte Bezeichnung für das Plugin) und anschließend die Versionsnummer über `GetVersions()` zu erhalten. Falls die Nutzung von ActiveX-Komponenten nicht möglich ist, wird `navigator.plugins` durchlaufen, bis das Adobe-Reader-Plugin gefunden wur-

de. Die Versionsangabe befindet sich in der Beschreibung des Plugins.[63]

3.4.4 Alle installierten Plugins

Browser, die das `navigator.plugins`-Array unterstützen, bieten die Möglichkeit, eine gesamte Liste der nutzbaren Plugins im entsprechenden Browser auszugeben. Der Microsoft Internet Explorer unterstützt dies nicht, sodass hier nur spezielle Plugins (siehe oben) abgefragt werden können.

```
1 var a = new Array();
2
3 try {
4   for (var i = 0; i < navigator.plugins.length; i++) {
5     a.push(navigator.plugins[i].name + ': ' + navigator.plugins
        [i].description + ' (' + navigator.plugins[i].filename +
        ')');
6   }
7   alert (a.toString());
8 } catch (e) {}
```

Listing 7: Ermitteln aller verfügbaren Plugins (JavaScript)

Erläuterung: Falls der Browser das `navigator.plugins`-Array unterstützt, werden alle installierten Plugins mit Name, Beschreibung und dem Dateinamen des Plugins ausgegeben.

3.4.5 Alle unterstützten Datenarten

Internetdokumente können Texte, Grafiken, Audio- und Videoinhalte oder sonstige Daten enthalten. Damit der Webbrowser diese Daten entsprechend inter-

[63] Vgl. Ratzloff: *Detecting plugins in Internet Explorer (and a few hints for all the others)*.

pretieren kann oder die Verarbeitung an ein Plugin weiterreichen muss, werden die Datenarten in verschiedenen MIME-Typen kategorisiert.[64] Durch die installierten Plugins und Anwendungen auf dem Client-Computer unterscheiden sich die Angaben auf unterschiedlichen Geräten; die Auflistung ist damit für ein Browser Fingerprint interessant.

```
1 var a = new Array();
2
3 try {
4   for (var i = 0; i < navigator.mimeTypes.length; i++) {
5     a.push(navigator.mimeTypes[i].type + ': ' + navigator.
          mimeTypes[i].description);
6   }
7   alert(a.toString());
8 } catch (e) {}
```

Listing 8: Ermitteln unterstützter MIME-Typen (JavaScript)

Erläuterung: Analog zur Ermittlung aller verfügbaren Plugins.

3.5 Aktives Fingerprinting mit Adobe Flash: Ermitteln der installierten Schriftarten

Desktop-Betriebssysteme wie Microsoft Windows oder Mac OS X erlauben die Installation von zusätzlichen Schriftarten (englisch *fonts*). Diese können von bestimmten Software-Paketen wie Microsoft Office oder Adobe Creative Suite

[64] Vgl. [RFC2046].

installiert worden sein.[65][66] Andererseits ist es möglich, einzelne und spezielle Schriftarten je nach Bedarf zu installieren, die kostenlos oder gegen Lizenzgebühren im Internet angeboten werden. Ebenso können eigene Schriftarten (z. B. die persönliche Handschrift) durch Verwendung spezieller Software erstellt werden, die dann möglicherweise nur auf einem Gerät weltweit installiert ist. Ließe sich eine Liste der installierten, verfügbaren Schriftarten des Client-Computers erstellen, so kann diese als erheblicher Baustein eines Browser-Fingerprints verwendet werden. Es kann nicht nur ein Gerät wiedererkannt werden, es können ebenso Rückschlüsse über die potentiell installierte Software gezogen werden.[67] Die Installation bestimmter Fonts kann ebenso auf Interessen und Vorlieben hinweisen: Ist die Schriftart einer Partei installiert, ist der Nutzer des Geräts möglicherweise ein Sympathisant oder gar Mitglied. Ebenso kursieren im Internet viele Schriftarten, mit denen sich Logos oder Zeichensätze von Filmen nachstellen lassen. Lässt sich solch eine auf einem Gerät auffinden, könnte auch dies ein Hinweis auf den persönlichen Filmgeschmack liefern. Für Werbeanzeigen kann diese Information sehr hilfreich sein.

Das HTML DOM oder sonstige JavaScript Funktionen erlauben keinen Zugriff auf die installierten Schriftarten, anders als Plugins wie Adobe Flash. Mögliche Umsetzungen sind u. a. in Dugonjić: *Detect visitor's fonts with Flash* beschrieben und benötigen neben einem HTML-Dokument eine kompiliertes Flash-Projekt als Shockwave Flash (SWF) (hier: `bfp.swf`). Die Flash-Anwendung muss im Hauptteil des HTML-Dokument inkludiert werden (zwischen `<body>` und `</body>`):

[65] Siehe Microsoft Corporation: *Mit Office installierte Schriftarten (Knowledge Base 837463)*.

[66] Siehe Adobe Systems Incorporated: *Fonts included with Adobe's Creative Suite 5 and Creative Suite 5.5 Applications*.

[67] Hier sei auf die Installation von bestimmter Software verwiesen, die „eigene" Schriftarten installiert.

```
<object id="flashFont" name="flashFont" type="application/x-
    shockwave-flash" width="1" height="1" data="bfp.swf">
  <param name="movie" value="bfp.swf" />
</object>
```

Listing 9: Flash-Einbindung (HTML)

Die Flash-Anwendung muss lediglich folgendes ActionScript enthalten, das auf einem Layer abgelegt wird. Dabei werden alle installierten Schriftarten abgefragt und die Funktion `receiveFonts()` aufgerufen, die als JavaScript im HTML-Dokument hinterlegt ist.

```
var user_fonts = TextField.getFontList();
getURL('javascript:receiveFonts("' + escape(user_fonts) + '")',
    '_self');
```

Listing 10: Ermitteln der installierten Schriftarten (ActionScript)

Die Funktion `receiveFonts()` wird in dem HTML-Dokument im Kopfbereich (`<head>` bis `</head>`) definiert. Sie verarbeitet die kommagetrennte Liste an Schriftarten und gibt diese dann aus. In den meisten Browsern wird die Zeichenfolge direkt als Parameter übergeben. Im Microsoft Internet Explorer kann es jedoch zu Problemen kommen. Alternativ kann die Objektmethode `GetVariable()` benutzt werden, um auf Variablen innerhalb von Flash zuzugreifen.[68]

```
function receiveFonts(f) {
  var obj = document.getElementById('flashFont');
```

[68] Wie hier beschrieben: Adobe Systems Incorporated: *getVariable()*.

```
3     var fonts;
4
5     if (typeof(f) != 'undefined') {
6       fonts = unescape(f);
7     } else if (typeof(obj.GetVariable) != 'undefined') {
8       fonts = obj.GetVariable('/:user_fonts');
9     }
10
11    if(typeof(fonts) == 'string') {
12      alert(fonts);
13    }
14  }
```

Listing 11: Empfangen der ausgelesenen Schriftarten von Flash (JavaScript)

Ebenso könnte auch ein Java-Applet benutzt werden, um das gleiche Ergebnis zu erzielen. Dies wird u. a. in ALEXANDER: *Java fonts - how to list of all the available fonts* beschrieben.

Für einen Browser Fingerprint ist nicht nur die Liste der installierten Schriftarten relevant, sondern auch die Reihenfolge, in der sie durch das Plugin ermittelt werden. Adobe Flash liefert häufig keine alphabetisch sortierte Liste; die Einträge wirken auf den ersten Blick willkürlich angeordnet. Die Reihenfolge bleibt jedoch auch bei mehreren Aufrufen gleich.[69]

3.6 Aktives Fingerprinting: Mit CSS prüfen, ob bestimmte Schriften installiert sind

Die Liste aller installierten Schriftarten lässt sich nur über Plugins wie Flash oder Java abfragen. Wenn allerdings nur geprüft werden soll, ob auf dem Clientcomputer eine oder mehrere bestimmte Schriftarten installiert sind, genügt die

[69] Vgl. ECKERSLEY: „How Unique Is Your Web Browser?“, S. 15.

Verwendung von JavaScript und CSS.[70] Hierbei wird der Umstand ausgenutzt, dass über die CSS-Eigenschaft `font-family` mehrere Schriftarten angegeben werden können. Dabei versucht der Browser, die Erstgenannte zu verwenden. Ist diese nicht verfügbar, wird die zweite Angabe verwendet. Ist auch diese nicht verfügbar, folgt die Drittgenannte, usw. Neben tatsächlichen Schriftarten lassen sich ebenso generische Schriftfamilien angeben, wie z. B. `monospace` (Schriftart mit fester Zeichenbreite und -höhe).

```
1 span { font-size: 20px; }
2 #f0 { font-family: monospace; }
3 #f1 { font-family: "Comic Sans MS", monospace; }
4 #f2 { font-family: "DieSchriftDieEsNichtGibt", monospace;}
```

Listing 12: Eine Standardschrift und zwei zu prüfende Fonts (CSS)

Nun kann die Breite und Höhe eines Textes, der mit einer generischen Schriftfamilie dargestellt wird, ermittelt werden. Nachfolgend wird versucht, den gleichen Text mit einer zu überprüfenden Schriftart darzustellen. Als Zweitangabe fungiert dann erneut die generische Schriftfamilie. Werden beide Texte in ihrer dargestellten Größe gleich angezeigt, ist die Schriftart vermutlich nicht installiert. Weichen die Werte ab, liegt die Schriftart vor:

```
1 function testFonts() {
2   var d = document.getElementById('f0');
3   var dX = d.offsetWidth;
4   var dY = d.offsetHeight;
5   var m1 = document.getElementById('f1');
6   var m2 = document.getElementById('f2');
```

[70] Das Vorgehen wird u. a. bei Patel: *JavaScript/CSS Font Detector* ausführlich beschrieben.

```
7   var b1 = !(m1.offsetWidth == dX && m1.offsetHeight == dY);
8   var b2 = !(m2.offsetWidth == dX && m2.offsetHeight == dY);
9   alert ("Schrift 1 installiert: " + b1 + "\nSchrift 2
        installiert: " + b2);
10  }
```

Listing 13: Vorhandensein von Schriftarten prüfen (JavaScript)

Damit die Texte dargestellt werden, müssen sie noch in den Hauptteil des Dokuments eingebunden werden. Ebenso wird nach dem Laden der Seite die oben definierte JavaScript-Methode aufgerufen.

```
1  <body onload="testFonts()">
2    <span id="f0">mmmmmmmm</span>
3    <span id="f1">mmmmmmmm</span>
4    <span id="f2">mmmmmmmm</span>
5  </body>
```

Listing 14: Beispieltexte (HTML)

3.7 Aktives Fingerprinting: Login-Status bei sozialen Netzwerken

Websites, die eine Anmeldung (Login) erfordern, bieten bestimmte Inhalte, Grafiken oder Skripte nur an, wenn der Benutzer auch angemeldet ist. Dies betrifft insbesondere soziale Netzwerke wie Twitter, Facebook oder Google+. Prinzipiell gilt dies aber für jede Seite, die ein Authentifizierung verlangt und Inhalte für unbekannte Nutzer aussperrt. Die Idee hinter dem folgenden Verfahren ist es, eben solche nicht-öffentlichen Ressourcen anzufordern. Da der Webbrowser generell bei jedem HTTP-Request an einen Webserver die zu ihm gehörigen Cookies mitsendet, kann dieser ermitteln, ob der Client eingeloggt

ist oder nicht. Dementsprechend wird der Inhalt geschickt oder der Zugriff wird verweigert. Als Steuerdaten in der HTTP-Schicht wird als Antwort der „Status Code“ 200 („OK“) oder eben 403 („Forbidden“) bzw. 404 („Not Found“) gesendet.[71] Das HTML DOM kennt für verschiedene Elemente die Ereignisse `onload()` und `onerror()`, die ausgelöst werden, falls ein Element korrekt geladen wurde bzw. nicht geladen werden konnte. Ist also eine Ressource bekannt, die ein Webdienst nur angemeldeten Benutzern zur Verfügung stellt, kann geprüft werden, ob der Nutzer bei eben jenem Dienst zurzeit eingeloggt ist.[72] Es können verschiedene DOM-Elemente, wie `<img />` oder `<script />` verwendet werden.

Eine Kombination von verschiedenen Diensten kann wertvolle Informationen für einen Browser Fingerprint liefern.[73] Folgendes Beispiel (eingefügt zwischen `<body>` und `</body>` eines HTML-Dokuments) prüft, ob der Nutzer bei Twitter eingeloggt ist:[74]

```
1 <img src="https://twitter.com/login?redirect_after_login=%2
    Fimages%2Fspinner.gif"
2   onload="alert('Eingeloggt.')"
3   onerror="alert('Nicht eingeloggt.')"
4   style="visibility:hidden" />
```

Listing 15: Login-Status bei Twitter (HTML)

[71] Für eine Liste der Status Codes siehe [RFC2616], S. 57 ff.

[72] Vgl. Cardwell: *Abusing HTTP Status Codes to Expose Private Information*.

[73] In der im Rahmen dieser Diplomarbeit durchgeführten Feldstudie wird dieses Verfahren nicht angewendet.

[74] Zuletzt getestet unter OS X 10.8 mit Safari, Firefox und Opera am 17. Februar 2013. Es ist zu bedenken, dass sich die verwendete URL ändern kann.

3.8 Aktives Fingerprinting: Performancemessung

Informationen über den verwendeten Browser und das Betriebssystem lassen sich, wie beschrieben, über den HTTP-Steuerdateneintrag „User-Agent“ (siehe Seite 32) oder per JavaScript mit `navigator.appVersion` (siehe Seite 36) ermitteln. Auf diese Daten muss aber nicht zwangsläufig Verlass sein. Viele Browser, z. B. Safari oder Firefox, erlauben nativ oder mit einer Erweiterung, dass die „User-Agent“-Angabe vom Nutzer geändert werden kann. So kann der aufgerufenen Website ein Browser vorgetäuscht werden, der tatsächlich aber gar nicht verwendet wird. Ebenso ist es möglich, eine willkürliche Zeichenfolge anzugeben, der die Ermittlung des Browsers oder des Betriebssystems völlig verhindert. In einem anderen Ansatz, dokumentiert in Mowery et al.: „Fingerprinting Information in JavaScript Implementations“, S. 2 ff. werden die o. g. Zeichenfolgen ignoriert. Stattdessen werden verschiedene Performance-Tests über JavaScript durchgeführt. Da verschiedene Browser auch verschiedene JavaScript-Implementierungen enthalten, schneiden diese in unterschiedlichen Tests auch anders ab. In der 2011 mit 1.015 Personen durchgeführten Feldstudie wurden insgesamt 39 verschiedene JavaScript Benchmark Tests angewendet.[75] Gleichzeitig wurden die Teilnehmer gebeten, in einem Formular den im Computer arbeitenden Prozessor (inkl. Taktrate), Arbeitsspeicher und Betriebssystem anzugeben. Um Referenzwerte zu erhalten, wurde der Test im Vorfeld auf einem einzelnen Computer mit einer Vielzahl von verschiedenen Browsern durchgeführt. Durch dieses Verfahren konnten 79, 8 % der verwendeten Browser inkl. Browserversion korrekt klassifiziert werden.[76] Wenn nur die Browserfamilie (Firefox, Internet Explorer, etc.) ermittelt werden sollte, lag die

[75] Namentlich einige der „V8“- und „SunSpider“-JavaScript-Tests. Diese sind online unter `http://code.google.com/p/v8/`) bzw. `http://www.webkit.org/perf/sunspider/sunspider.html` abrufbar (beide zuletzt am 23. Februar 2013 auf Verfügbarkeit geprüft).

[76] Vgl. Mowery et al.: „Fingerprinting Information in JavaScript Implementations“, S. 4.

Rate sogar bei $98,2\,\%$.[77] Um die verwendeten Betriebssysteme erkennen zu können, wurde Firefox 3.6 auf verschiedenen Betriebssystemen für Referenzwerte verwendet. Da zu wenig Teilnehmer andere Betriebssysteme als Microsoft Windows verwendet haben, konnte lediglich für Windows eine Identifikationsrate bestimmt werden, die allerdings $98,5\,\%$ betrug.[78] Die Erkennungsrate für den verwendeten Prozessor lag aber mit $45,3\,\%$ deutlich niedriger.[79] Da das in der Studie verwendete Verfahren über drei Minuten benötigte, ist es für den praktischen Einsatz in dieser Form nicht geeignet.[80] Dennoch kommen die Autoren zu folgendem Schluss:

> Our implementations represent a lower bound on the effectiveness of our techniques. We have shown that it is possible to distinguish between browsers, along with the underlying system hardware and software, based solely on scripting benchmarks. We believe that a finer-grained approach to JavaScript performance fingerprinting can provide even more detailed information, such as hardware revisions within a processor family, clock speed, cache size, and the amount of RAM on the target system. Secondly, extending our technique to mobile devices should produce excellent results, given their unique and constant combination of mobile browser, operating system, and mobile hardware.[81]

3.9 Stand der Forschung

Die Electronic Frontier Foundation (EFF), eine nichtstaatliche Organisation, die sich für Bürgerrechte im Internet einsetzt, hat vom 27. Januar bis 15. Februar 2010 auf einer Website Browser Fingerprints gesammelt, ähnlich wie in der für diese Arbeit verwendeten Feldstudie. Die Ergebnisse sind in Eckersley:

[77] Vgl. ebd., S. 4.
[78] Vgl. ebd., S. 4.
[79] Vgl. ebd., S. 6.
[80] Vgl. ebd., S. 3.
[81] Ebd., S. 10.

„How Unique Is Your Web Browser?“ beschrieben und werden hier zusammengefasst. Dabei sammelte die EFF folgende Merkmale:

- Alle im Abschnitt 3.2.3, S. 31 ff., genannten Daten, mit Ausnahme von „Referer“,
- Erlaubnis zur Verwendung von Cookies (ja/nein),
- die vom Nutzer verwendete Bildschirmauflösung (siehe Abschnitt 3.3.3, S. 37),
- die vom Nutzer verwendete Zeitzone (siehe Abschnitt 3.3.4, S. 39),
- die im Browser aktiven Plugins und unterstützen MIME-Typen (siehe Abschnitt 3.4, S. 41 ff.),
- die installierten Schriftarten (siehe Abschnitt 3.5, S. 47 ff.),
- sowie ein Test, ob „DOM local storage“, „DOM session storage“ und „userData“ des Microsoft Internet Explorers unterstützt wird (mehr dazu in Abschnitt 4, S. 59 ff.).

Zusätzlich legte das als „Panopticlick“ benannte Projekt einen HTTP-Cookie mit einer Lebenszeit von drei Monaten auf dem Clientcomputer ab. In diesem Cookie wurde eine eindeutige ID vermerkt, die als Identifikationsquelle dienen konnte. Des Weiteren wurde die IP-Adresse des Teilnehmers serverseitig (als Hashwert) gespeichert.[82]

Das Projekt ist weiterhin unter `https://panopticlick.eff.org` nutzbar, die im o. g. Papier erfassten Ergebnisse betrachten aber lediglich 470.161 gesammelte Datensätze. Die EFF entfernte bei der Auswertung einige Datensätze,

[82] Vgl. Eckersley: „How Unique Is Your Web Browser?“, S. 7 f.

z. B. von Einträgen, die im Fingerprint, IP und einzigartiger ID übereinstimmten. So waren ursprünglich 1.043.426 Fingerprints vorhanden, die dann deutlich reduziert wurden. Von diesen endgültig verwendeten Fingerprints waren 83, 6 % einzigartig, wobei weitere 5, 3 % nur aus einer Doppelung bestanden. In der von der EFF durchgeführten Studie wurde vor allem die Bedeutung von Adobe Flash und der Java Virtual Machine klar – sind beide aktiviert, stieg die Einzigartigkeitsquote auf 94, 2 %.[83]

Neben der Aussagekraft von Plugins und die Möglichkeit der Erkennung der Schriftarten sowie der Analyse der „User-Agent"-Zeichenfolge sind generell von moderneren Desktop-Browsern aussagekräftigere Fingerabdrücke erwartbar (ca. 90 % sind einzigartig). Smartphones wie das iPhone, das sich kaum bis gar nicht konfigurieren lässt, haben bei einem Fingerprint am wenigsten Aussagekraft (vgl. ECKERSLEY: „How Unique Is Your Web Browser?", Abb. 2 auf S. 10).

In der Studie wurde auch die Veränderung eines Fingerprints untersucht. Da Cookies mit einzigartigen Identifikationsnummern abgelegt wurden, konnten Clients sicher wiedererkannt werden. Bei über 37 % der wiederkehrenden Besucher konnte mindestens eine Veränderung des Fingerprints festgestellt werden.[84]

In einer ähnlichen Studie, dokumentiert in BROENINK: „Using Browser Properties for Fingerprinting Purposes", wurden ebenfalls virtuelle Fingerabdrücke gesammelt und ausgewertet. Allerdings umfasste die Stichprobe lediglich 1.124 Personen, von denen 95,9 % einzigartige Fingerprints besaßen. Die abgefragten Merkmale gleichen denen des Panopticlick-Projekts. BROENINK beschreibt, dass die „Do-Not-Track"-Funktionen der Browser keinen Einfluss auf das Tracking über Fingerprints haben. Jedoch kann das Tor-Netzwerk und der

[83] Vgl. ebd., S. 2.
[84] Vgl. ebd., S. 12.

Tor-Browser die Wiedererkennung durch Fingerprints erschweren.[85] Da jedoch nur wenige Personen Tor nutzen, können diese wieder daran erkannt werden. Es ergibt sich ein Paradoxon: Je besser sich der Nutzer schützen möchte, desto einfacher kann er wiedererkannt werden:

> The Tor Browser does the best effort of remaining anonymous. The user agent is changed to a common browser version, all plugins are disabled and several properties are changed to some default value. Although the uniqueness of the browser is significantly decreased, the Tor Browser users can still be identified as a group. This is best expressed by an example; the available screen width and height are normally exposed through Javascript, as well as the total screen width and height. The Tor Browser modifies the latter to the same values as the first. This is quite unique, as most operating systems have a available height that is slightly less than the total screen height (due to a taskbar). Combined with the fact that most, if not all, Tor exit nodes are known, we can identify Tor users as a group. Going further, we know that Tor users are quite rare and Tor Browser users even rarer, so the chance we find such a user, is quite low. Therefore, we can with high probability re-identify a Tor user, which brings us back to the fingerprinting paradox; when a small subgroup of users is using a privacy-enhancing technology, they can be identified as being a member of that small subgroup.[86]

Unabhängig vom Browser Fingerprinting wird in DE MONTJOYE ET AL.: „Unique in the Crowd: The privacy bounds of human mobility“ gezeigt, dass Nutzer insbesondere im mobilen Bereich recht einfach durch einige Merkmale identifizierbar sind.

[85] Das Tor-Netzwerk (Akronym für The Union Routing) ist ein Netzwerk zur Anonymisierung des Datenverkehrs im Internet. Mehr Informationen auf `http://www.torproject.org` (abgerufen am 10. Mai 2013).

[86] BROENINK: “Using Browser Properties for Fingerprinting Purposes,” S. 6.

4 Weitere Trackingverfahren

Neben dem Browser Fingerprinting, bei dem die Daten des Nutzers typischerweise auf dem Server und nicht clientseitig gespeichert werden, gibt es weitere Trackingverfahren, die weiterhin Daten auf dem Computer des Clients ablegen und somit „Spuren“ hinterlassen. In diesem Kapitel wird die Möglichkeit der „Zweckentfremdung“ von Grafiken vertieft. Ebenso werden viele weitere Möglichkeiten des Trackings kurz erläutert.

4.1 Codierung von Informationen in Cache-Grafiken

Informationen können nicht nur in textbasierten Cookies abgelegt, sondern auch in Grafiken gespeichert werden. Ein im Internet populäres Format ist dabei Portable Network Graphics (PNG).[87] PNG erlaubt die Speicherung von Graustufen und Farbbildern mit bis zu 16 Bit pro Farbkanal (Farbanteil Rot, Grün und Blau (RGB)) und pro Pixel.[88] Ebenso unterstützt PNG auch Transparenzeinstellungen (Alpha-Kanal), die ebenso bis zu 16 Bit umfassen können.[89] Im Folgenden wird aber von einer 8 Bit Grafik ohne Transparenz ausgegangen (somit drei Angaben mit Werten von jeweils $0 - 255$). Folgende Tabelle illustriert beispielhaft einige Farbangaben:

[87] Ein weiteres Format ist u. a. Graphics Interchange Format (GIF). Die weiteren Angaben sind fast analog für eben dieses Format anwendbar.

[88] Vgl. Duce: *Portable Network Graphics (PNG) Specification (Second Edition)*, Abschnitt 4.1 „Images“.

[89] Vgl. ebd., Abschnitt 4.1 „Images“.

Rotwert	Grünwert	Blauwert	Resultierende Farbe
255	0	0	(rot)
255	255	0	(gelb)
0	0	255	(blau)
102	255	102	(hellgrün)
204	204	204	(grau)
255	255	255	(weiß)

Statt der Farbwerte könnten auch numerische Informationen oder Buchstaben kodiert werden. Da der American Standard Code for Information Interchange (ASCII) Standard lediglich 7 Bit umfasst, könnten bei einer 8 Bit PNG-Grafik ohne Transparenz drei Zeichen pro Pixel gespeichert werden.[90] Es könnte der Rotwert den ersten, der Grünwert den zweiten und der Blauwert den dritten Buchstaben repräsentieren, wobei jeweils das höchste Bit (für die Werte 128 bis 255) ignoriert wird. Folgende Beispielangaben in Verbindung mit der ASCII-Tabelle [91] illustrieren die Möglichkeit der Speicherung von Zeichen in Grafiken:

Rotwert	Grünwert	Blauwert	Farbe	ASCII-Zeichen
68	101	114		Der
83	79	83		SOS
65	122	33		Az!

[90] Vgl. [RFC20].

[91] Wie definiert in [RFC20]. Zur einfachen Übersicht u. a. auch unter `http://www.computer-masters.de/ascii-tabelle.php` (zuletzt geprüft am 9. Februar 2013) abrufbar.

Um solche oder ähnliche Angaben zu speichern, muss zunächst eine spezielle Grafik auf dem Server generiert werden. Diese wird bei dem Aufruf einer Website an den Client übertragen, der diese fortan im Puffer-Speicher (engl. *cache*) vorhält. Wird die Website erneut aufgerufen, kann ein Skript, welches mit der Website übertragen wird, die lokal im Cache gespeicherte Grafik auslesen. Dies ist mit JavaScript und dem HTML5 DOM Objekt `<canvas />` möglich. Wichtig ist hierbei, dass die lokal gespeicherte Grafik geladen wird. Nur wenn diese nicht existiert (beim ersten Aufruf oder wenn der Cache gelöscht wurde), diese vom Server (erneut) angefordert wird.

Im folgenden Beispielcode wird eine PNG-Datei (8 Bit pro Farbkanal RGB, keine Transparenz) auf dem Server erstellt, die einen Pixel breit und einen Pixel hoch ist. In ihr wird eine Zahl (max. $2^{(3 \cdot 8)} - 1 = 16777215$) abgelegt, die von dem clientseitigen JavaScript festgelegt wird. Die Einbindung der Grafik ist wie folgt:

```
1  var COOKIE_EXPIRE_PAST = 'Sun, 30 Sep 2012 00:00:00 GMT';
2  var COOKIE_EXPIRE_FUTURE = 'Sat, 03 Jan 2015 12:00:00 GMT';
3  var png_id = Math.floor(Math.random() * Math.pow(2, 24) - 1);
4  var b = false;
5
6  function addPng() {
7    var img = document.createElement('img');
8    img.onload = function() {
9      if (this.src != '') {
10       window.setTimeout('getPngId()', 50);
11       document.cookie = 'create_png=; expires=' +
             COOKIE_EXPIRE_PAST;
12     }
13   }
14   img.onerror = function() {
15     if (this.src != '') {
```

```
16        this.src = '';
17        document.cookie = 'create_png=' + png_id + '; expires=' +
              COOKIE_EXPIRE_FUTURE;
18        this.src = 'png.php';
19        if (!b) {
20          alert("Zufallszahl " + png_id + " an Server uebertragen
                . Erstelle PNG.");
21          b = true;
22        }
23      }
24    }
25    img.id = 'pngid';
26    img.src = 'png.php';
27    img.style.visibility = 'hidden';
28    document.getElementsByTagName('body')[0].appendChild(img);
29  }
```

Listing 16: Laden der PNG-Datei (JavaScript)

Erläuterung: Es wird ein leeres DOM-Element `<img />` erstellt und zwei anonyme Funktionen definiert. Anschließend wird die Grafik geladen, indem die `src`-Eigenschaft gesetzt wird (das Skript, das die Grafik erstellt, muss somit als „png.php“ gespeichert werden und im selben Verzeichnis wie diese HTML-Datei liegen). Die Grafik wird unsichtbar geschaltet und dem DOM hinzugefügt. Die Ereignisprozeduren werden aufgerufen, sobald das Event `onload()` (erfolgreiches Laden der Grafik, d. h. Grafik konnte aus dem Cache geladen werden) oder `onerror()` (Grafik konnte nicht geladen werden) eintritt. Im ersten Fall wird `getPngId()` aufgerufen und ein mögliches vorhandenes Cookie, das den zu speichernden Wert enthält, wird gelöscht. Im zweiten Fall, falls die Grafik also nicht geladen werden konnte, wird ein Cookie angelegt, das den zu speichernden Wert enthält (hier eine zufällig generierte Zahl, `png_value`). Dann wird der Browser aufgefordert, die Grafik erneut zu laden, indem der Verweis

auf die Grafik (`img.src`) neu gesetzt wird.

In demselben HTML-Dokument muss die Funktion `getPngId()` definiert werden, die das Auslesen der Zahl übernimmt:

```
1  function getPngId() {
2    var png = document.getElementById('pngid');
3    var canvas = document.createElement('canvas');
4    var ctx = canvas.getContext('2d');
5    ctx.drawImage(png, 0, 0);
6
7    var img_data = ctx.getImageData(0, 0, 1, 1);
8    var pix = img_data.data;
9
10   var dec = 0;
11   var r, g, b;
12
13   r = pix[0]; g = pix[1]; b = pix[2];
14   dec = r * Math.pow(2, 16) + g * Math.pow(2, 8) + b;
15   alert("Ausgelesen aus PNG: " + dec);
16 }
```

Listing 17: Auslesen der Zahl in der Grafik (JavaScript)

Erläuterung: Um die Grafik auszulesen, wird ein HTML5-Tag `<canvas />` benötigt, welches Grafikschnittstellen über das DOM zur Verfügung stellt. In jenes Objekt wird die geladene Grafik geladen (Zeile 5), um dann pixelweise über die Funktion `getImageData()` ausgelesen werden zu können (in diesem Fall wird nur ein Pixel ab Koordinate (0, 0) mit Breite 1 und Höhe 1 ausgelesen). [92] Im zurückgegebenen Objekt `data` sind alle Farbinformationen $(0-255)$,

[92] Mehr Informationen u. a. hier: `http://www.w3schools.com/tags/canvas_getimagedata.asp` (zuletzt abgerufen am 15. Februar 2013).

getrennt nach Rot-, Grün- und Blauwert (Indizes 0, 1 und 2), sowie ggf. Alphawert (Transparenz, Index 3) abgelegt. Sollten mehrere Pixel ausgelesen werden, sind die Informationen für den zweiten Pixel ab Index 4, für den dritten Pixel ab Index 8, usw. abrufbar. In Zeile 14 werden die Farbinformationen in den vorher gespeicherten Zahlenwert (siehe unten) zurückgewandelt, wobei der Blauwert die unteren 8 Bit belegt, der Grünwert den mittleren Bereich und der Rotwert die höchsten 8 Bit speichert.

Damit die Grafik zu dem HTML-Dokument zur Laufzeit auch hinzugefügt wird, muss die Methode `addPng()` nach dem vollständigen Laden der Seite durch den Browser aufgerufen werden. Dies kann z. B. durch das `onload()`-Ereignis ermöglicht werden:

```
1 <body onload="javascript:addPng()">
```

Listing 18: Event-Handler hinzugefügt (HTML)

Es fehlt nun noch die serverseitige Erstellung der Grafik, die im folgenden in der Skriptsprache PHP beschrieben wird. Das entsprechende Skript erstellt dynamisch zur Laufzeit eine 1x1-Pixel große PNG-Ausgabe (PHP bietet dafür entsprechende integrierte Methoden). Diese soll jedoch nur generiert werden, wenn dies explizit über ein Cookie gefordert wird. Andernfalls soll der Browser veranlasst werden, die Grafik aus dem Cache zu laden.

```
1 if (isset($_COOKIE['create_png']) && $_COOKIE['create_png'] >
     0) {
2   $x = 1; $y = 1;
3
4   $img = imagecreatetruecolor($x, $y);
5   $id = $_COOKIE['create_png'];
6
```

```
7    $hex = str_pad(dechex($id), 6, '0', STR_PAD_LEFT);
8
9    $color = imagecolorallocatealpha($img, hexdec(substr($hex, 0,
         2)), hexdec(substr($hex, 2, 2)), hexdec(substr($hex, 4,
        2)), 0);
10   imagesetpixel($img, 0, 0, $color);
11
12   header('Content-Type: image/png');
13   header('Last-Modified: Sun, 30 Sep 2012 00:00:00 GMT');
14   header('Expires: Sat, 03 Jan 2015 16:00:00 GMT');
15   header('Cache-Control: private, max-age=2592000');
16   imagepng($img, null, 0);
17 } else {
18   header("HTTP/1.1 304 Not Modified");
19 }
```

Listing 19: Erstellen der Grafik (PHP)

Erläuterung: Zu Beginn wird geprüft, ob ein Cookie „create_png“ vorliegt. Ist dies nicht der Fall, generiert dieses Skript keine Grafik, sondern sendet lediglich den HTTP-Statuscode 304 (Zeile 18), der den Browser darüber informiert, das keine Änderung am Dokument bzw. Grafik vorliegt.[93] Der Browser wird daraufhin versuchen, die Grafik lokal aus dem Cache zu laden. Sollte doch eine Grafik explizit angefordert worden sein (Listing 16 auf Seite 61, Zeile 17), wird diese ab Zeile 2 erstellt.[94] Die im Cookie abgelegte (dezimale) Zahl wird in Zeile 7 in eine sechsstellige hexadezimale Zahl umgewandelt und ggf. durch führende Nullen ergänzt. Diese hexadezimale Zahl wird dann erneut in drei Teile getrennt: die zwei Stellen links für den Rotwert, die mittleren Stellen

[93] Vgl. [RFC2616], S.63.

[94] Informationen zu den Grafikschnittstellen von PHP finden sich in der offiziellen Dokumentation: `http://www.php.net/manual/de/ref.image.php` (zuletzt abgerufen am 15. Februar 2013).

für den Grünwert und die rechten Stellen für den Blauwert. Bevor die Grafik tatsächlich ausgegeben wird, werden im HTTP-Header Steuerdaten für den Browser mitgeschickt; u. a., dass es sich bei den zu empfangenden Daten um eine PNG handelt (siehe MIME-Typ-Angabe in Zeile 12) und dass diese im Cache gehalten werden soll, damit diese später ausgelesen werden kann (Zeile 13 ff.).

Da Grafiken auch geladen werden können, ohne dass sie sichtbar im Browserfenster erscheinen (vgl. die CSS-Angabe `visibility: hidden`), können Grafiken zumindest in der Theorie unter HTML5 als Cookie-Ersatz dienen. Wie die Feldstudie zum Browser Fingerprinting allerdings zeigt, sind korrekte Ergebnisse beim Auslesen, insbesondere bei manchen Browsern, zurzeit nicht immer zu erwarten. Die vorgestellten Codebeispiele orientieren sich stark an dem 2010 vorgestellten „evercookie“, das zum ersten Mal im größeren Rahmen PNG-Dateien als Cookies zweckentfremdete. Weitergehende Informationen sind dazu auf `http://samy.pl/evercookie/` [95] zu finden.

4.2 Flash-Cookies (Local Shared Objects)

Anwendungen, die in Flash-Containern in Websites eingebunden sind, haben die Möglichkeit, Daten auf dem Client zu speichern. Adobe bezeichnet dies als Local Shared Object (LSO). Als Synonym wird auch der Begriff „Flash-Cookie“ verwendet.[96] Während HTTP-Cookies maximal 4 Kibibyte (KiB) groß sein können, gibt es für LSO praktisch kein Limit. Sobald bestimmte Größen erreicht worden sind (z. B. 100 KiB), muss der Benutzer die weitere Speicherung von Daten jedoch bestätigen.[97] Anders als HTTP-Cookies werden Flash-Cookies nicht vom Browser selbst, sondern von dem Plugin gespeichert und verwaltet.

[95] Zuletzt am 17. Februar 2013 abgerufen.
[96] Vgl. Adobe Systems Incorporated: *What are local shared objects?*
[97] Vgl. Adobe Systems Incorporated: *Gemeinsame Objekte*.

Dadurch sind Flash-Cookies browserübergreifend abrufbar. Falls ein Nutzer einen zweiten Browser verwendet, kann er dennoch direkt identifiziert werden, obwohl er dies vielleicht genau durch die Wahl des zweiten Browsers vermeiden möchte.[98] Bis zur Adobe Flash Version 10.1 bestand auch das Problem, dass ein Zugriff auf die Flash Cookies auch dann möglich war, wenn im verwendeten Browser explizit die Option „Private Browsing“ aktiviert wurde und das Speichern und Auslesen von Cookies somit nicht gewünscht war.[99] In einer Studie der University of California, Berkeley von 2009 konnte gezeigt werden, dass viele Websites Flash-Cookies als zusätzliche Speichermöglichkeit verwenden, ohne darauf konkret hinzuweisen. Es wurden häufig die gleichen Informationen in das Flash-Cookie wie auch in das HTTP-Cookie geschrieben, um die Informationen wiederherstellen zu können, falls letztgenanntes nicht mehr verfügbar ist.[100] Microsoft Silverlight bietet mit „Isolated Storage“ eine ähnliche Funktionalität, wie in Van Patten: „Isolierter Speicher in Silverlight 2“ nachzulesen ist.

4.3 Daten im Fensternamen (window.name)

Das HTML DOM bietet die Möglichkeit, in der Eigenschaft `name` des Objekts `window` eine Zeichenfolge zu speichern. Da Zeichenfolgen (Strings) qua Definition von JavaScript keine maximale Größe haben, hängt es von der Browserimplementierung ab, wie viele Daten gespeichert werden können. Dabei ist aber von mehreren Kilobyte oder gar Megabyte auszugehen. Die Eigenschaft wurde ursprünglich eingeführt, damit Hyperlinks auch in anderen Fenstern (oder Tabs) als in dem eigenen geöffnet und somit per Skript direkt über den Bezeichner

[98] Vgl. McKinley: *Cleaning Up After Cookies*, S.5.
[99] Vgl. Bachfeld: *Adobe Flash 10.1 unterstützt "Private Browsing"*.
[100] Siehe dazu Soltani et al.: *Flash Cookies and Privacy*.

angesprochen werden können.[101] Die gespeicherte Wert bleibt so lange für das aktive Fenster bzw. Tab erhalten bis dieses geschlossen wird. So kann in einer Datei der Wert per JavaScript gesetzt werden (`window.name = 'Irgendwas'`) und durch eine andere Datei, die auch auf einer anderen Domain liegen kann, wieder ausgelesen werden (`alert(window.name)`). Es kann somit ein Tracking über verschiedene Websites hinweg erfolgen, das enorme Sicherheitsprobleme implizieren kann. Andererseits wird es abrupt beendet, sobald der Nutzer seine Webseitenbesuche in einem anderen Fenster (Tab) fortsetzt bzw. im aktuellen Fenster beendet.

4.4 Web Storage

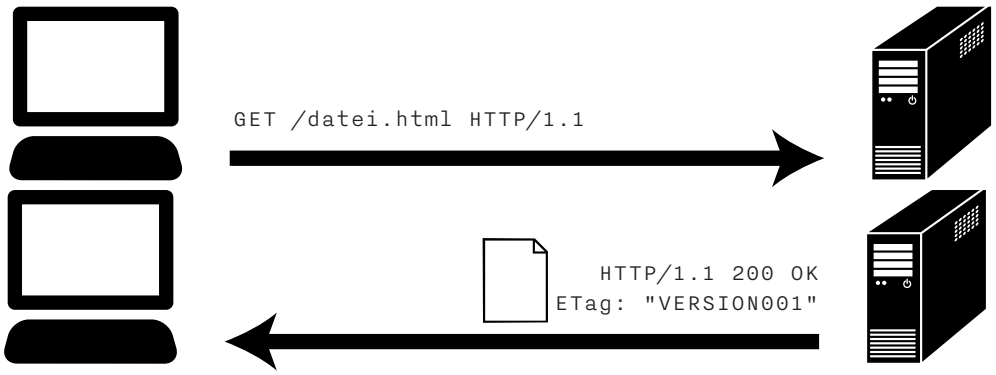
Das W3C hat mit dem Web Storage (auch DOM Storage) eine Möglichkeit geschaffen, ähnlich wie Cookies Daten lokal auf dem Client zu speichern.[102] Anders als Cookies können hier aber deutlich mehr Daten als 4 KiB gespeichert werden (beispielsweise bis zu 10 Megabytes beim Microsoft Internet Explorer).[103] Wie auch bei Cookies werden bei Web Storage Schlüssel-Werte-Paare zur Zuordnung und Speicherung verwendet. Darüber hinaus ist auch ein Standard zur Speicherung von lokalen Daten in Form von Structured Query Language (SQL) vorhanden, der jedoch momentan noch nicht von allen Browsern vollständig unterstützt wird.[104] Im Gegensatz zu Cookies wird der DOM Storage nicht automatisch bei einem HTTP-Request mitgeschickt. Um solche lokalen Daten an den Server zu übertragen, muss beispielsweise AJAX eingesetzt werden. Ebenso wird bei dem Web Storage zwischen „local“ und „session“ Web Storage unterschieden. In der ersten Variante bleiben die Daten erhalten, nach-

[101] Vgl. Mozilla Developer Network: *window.name*.
[102] Siehe Hickson: *Web Storage*.
[103] Siehe Microsoft Developer Network (MSDN): *Introduction to Web Storage*.
[104] Vgl. Hickson: *Web SQL Database*.

dem der Browser geschlossen wurde. Dabei haben Skripte einer Domain Zugriff auf alle Daten im DOM Storage, die von der Domain gespeichert wurden. Im Falle der zweiten Variante gehen die Daten verloren, sobald das Fenster bzw. das Tab geschlossen wird. Ebenso hat auch jede Domain in jedem Fenster (Tab) einen eigenen „session storage“, d. h. es kann nicht auf die Daten von anderen Fenstern (Tabs) zugreifen, auch wenn diese von derselben Domain stammen. Für das Tracking ist insbesondere „local storage“ von Interesse.

4.5 ETag im HTTP-Header

Im HTTP-Standard 1.1 ist bei Serverantworten die Angabe eines ETag-Eintrags („Entity Tag“) möglich. Als Wert für das ETag kann der Server eine beliebige Zeichenfolge angeben, z. B. eine Prüfsumme der angeforderten Ressource (durch eine Hashfunktion). Andererseits können auch lediglich eine Revisionsnummer, eine Zeitmarke oder sonstige Information angegeben werden.[105] Der Browser kann die erhaltene Datei, inkl. der ETag-Angabe im Kopfbereich, im Cache ablegen. Folgendes Beispiel illustriert die Verhaltensweise von Client (links) und Server (rechts):

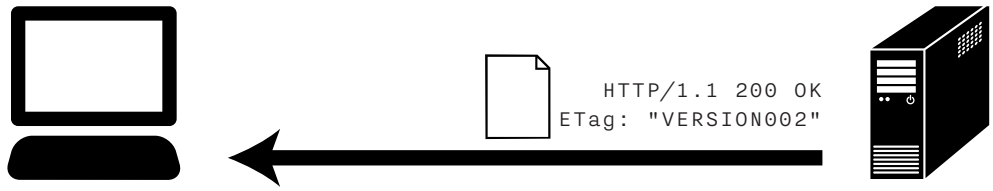


[105] Vgl. [RFC2616], S. 126.

Fordert der Nutzer dieselbe Ressource erneut an, ermöglicht HTTP 1.1 dem Browser das Mitsenden des gespeicherten ETags in der HTTP-Anfrage. Das entsprechende Feld in den Steuerdaten nennt sich beim HTTP-Request „If-None-Match“.

Der Server prüft nun, ob dieser Wert in der Anfrage der aktuellen Entitätmarke (ETag) entspricht. Falls dies zutrifft, sendet der Server lediglich eine kurze Antwort mit dem HTTP-Status 304 („Not Modified“, nicht geändert) zurück. Somit müssen nicht alle Daten erneut übertragen werden und der Browser kann die Datei aus dem Cache verwenden.

Falls der übertragende Wert „If-None-Match“ in der Anfrage nicht dem aktuellen ETag entspricht, wird die Datei komplett mit dem neuem ETag übertragen.



Um das ETag zum Tracking von Nutzern zu verwenden, wird statt einer Versionsinformation für das angeforderte Dokument eine (generierte) Identifikationsnummer des Nutzers mitgesendet. Diese wird beispielsweise beim ersten Aufruf willkürlich angelegt und als ETag mitgeschickt. Ruft der Nutzer die entsprechende Ressource erneut auf, so wird diese im „If-None-Match“-Feld vermerkt. Der Server erhält die Information, dass der entsprechende Nutzer erneut aktiv ist und übermittelt die (möglicherweise aktualisierte) Datei erneut, behält das ETag jedoch bei. Diese Identifikationsnummer wird bei allen Ressourcen verwendet, die dieser Nutzer anfordert. Somit dient das ETag nicht mehr der Versionskontrolle einer Datei, sondern dem Tracking eines Clients über mehrere Ressourcen.

5 Browser Fingerprinting Feldstudie

Im Rahmen der Diplomarbeit wurde vom 15. November bis 15. Dezember 2012 eine Feldstudie zum Browser Fingerprinting durchgeführt. Diese war online unter `http://bfp.henning-tillmann.de` abrufbar. In diesem Kapitel wird die Feldstudie beschrieben und ausgewertet.

5.1 Ziel der Feldstudie

Während des Zeitraums der Datenerfassung sollten möglichst viele Fingerprints von unterschiedlichen Nutzern gesammelt werden. Wenn möglich, sollten die Teilnehmer nicht nur einmal, sondern bestenfalls mehrmals über eine größere Zeitspanne teilnehmen, um Änderungen an den Merkmalen feststellen zu können. Der Feldstudie geht die Hypothese voraus, dass viele, wenn auch nicht alle, Nutzer über ihren Browser Fingerprint zu identifizieren sind, d. h., dass diese virtuellen Fingerabdrücke in der Strichprobe einzigartig sind. Ebenso wird angenommen, dass mit einem hinreichend effizienten Algorithmus Änderungen an bestimmten Merkmalen erkannt werden können.

5.2 Methodik

Die folgenden Abschnitte beschreiben die Funktionsweise des Projekts und wie die Daten ausgewertet werden.

5.2.1 Funktionsweise des Projekts

Die Feldstudie sollte sowohl passive als auch aktive Fingerprintmerkmale (vgl. Kapitel 3) sammeln. Die Website war sowohl in Deutsch als auch in Englisch aufrufbar. Bevor Daten gesammelt wurden, musste jeder Teilnehmer einer Datenschutzerklärung zustimmen, die über das Sammeln der Daten und das Ablegen eines Cookies informierte. Nach der Zustimmung und einem Klick auf

„Teilnehmen“ wurde beim Aufruf von `bfp.php` die für ein passiven Browser Fingerprint relevanten Daten gespeichert. Im Fall des ersten Aufrufs der Seite, wurde eine eindeutige Kennung (*uid*) generiert und in einem Cookie abgelegt. Neben dieser eindeutigen Nummer wurde auch eine neue Datenbankzeile mit einer eindeutigen Zeilennummer (*id*) erstellt, die in das dynamisch generierte HTML-Dokument geschrieben wurde. Beide Daten (*uid* und *id*) waren fortan für die Kommunikation zwischen Client und Server notwendig. Nachdem die Seite vom Browser vollständig geladen wurde und das Event `onload()` eintrat, begann das aktive Fingerprinting (siehe `bfp.js`). Für die Liste der Schriftarten wurde ein Flash-Objekt eingesetzt, das die entsprechenden Daten sammelte und eine JavaScript-Methode aufrief, die diese Daten, sowie auch alle weiteren aktiven Fingerprint-Merkmale, per AJAX an den Server sendete. Jede Kommunikation enthielt das Tupel (id, uid). Obwohl die *id* theoretisch als Primärschlüssel zur eindeutigen Identifikation eines Datensatzes ausgereicht hätte, sollte durch die zusätzliche Angabe der *uid* Manipulationen in der Übertragung verhindert werden.

Nach der Übermittlung der Daten konnte der Nutzer seine ermittelten Merkmale ansehen oder aber auch das Projekt über soziale Netzwerke weiterempfehlen. Ebenso hatte der Nutzer jederzeit die Möglichkeit, die Teilnahme an dem Projekt zu beenden und das Cookie über die Website zu löschen. In diesem Fall hätte er bei einer erneuten Teilnahme eine neue *uid* erhalten (dies motiviert die Definition der Menge F_i, siehe Seite 78).

5.2.2 Struktur der Datenbank

Das Projekt verwendet eine MySQL-Datenbank. Die genaue Struktur ist dem elektronischen Anhang zu entnehmen. Da Schriftarten, installierte Plugins und akzeptierende MIME-Typen nur anonymisiert als Hashwerte im Rahmen der Veröffentlichung dieser Diplomarbeit weitergegeben werden, wurden zur Lauf-

Abbildung 1: Bildschirmfoto nach Übermittlung des Fingerprints

zeit per Message-Digest Algorithm 5 (MD5)[106] 128-Bit Hashwerte erzeugt. Es liegen weder bei den Schriftarten*1, noch bei den Plugins*2 oder den akzeptierenden MIME-Typen*3 Kollisionen vor. Die Relation verwendet die *id* als Surrogat- und Primärschlüssel. Sollten Merkmale nicht ermittelt werden können, erhalten sie den Nullwert `null` (nicht zu verwechseln mit der Zahl 0 oder einer leeren Zeichenfolge „“). Die Fingerprint-Merkmale werden durch eine Zeitmarke *dt* und die ermittelte PNG-ID ergänzt (siehe Kapitel 5.7, S. 98 ff.).

5.2.3 Teilnehmer

Auf `http://bfp.henning-tillmann.de` konnte jeder zwischen dem 15. November und dem 15. Dezember 2012 teilnehmen. Einzige Zugangsvoraussetzung war aktiviertes JavaScript, um Spambots und Webcrawler nicht als aktive Teilneh-

[106] Vgl. [RFC1231].

mer zu werten. Ebenso lag der Fokus der Studie auf aktivem Fingerprinting, das ohne clientseitige Skriptsprache nicht durchführbar ist. Da keine Formulare ausgefüllt werden und der Teilnehmer lediglich die zwei Klicks tätigen musste (Häkchen zur Zustimmung der Datenschutzerklärung und ein weiterer Klick auf „Teilnehmen“), waren die Barrieren zur Teilnahme sehr gering. Insgesamt wurden 23.709 Datensätze gesammelt.*[4] Dabei wurden 18.692 einzigartige Kennungen (*uid*) generiert, d. h. dass dies die obere Schranke an teilgenommen Geräten mit einem jeweiligen Browser markiert.*[5] Der Höchstwert an Teilnahmen einer eindeutigen *uid* beträgt 57, durchschnittlich wurde eine *uid* $1,268$-mal verwendet.*[6]

5.2.4 Merkmale und Fingerprint-Funktion

In den folgenden Abschnitten wird die Menge aller Fingerprints als X bezeichnet, für einzelne Fingerprints gilt $x \in X$. Die Menge der Merkmale wird mit M bezeichnet. Einzelne Merkmale $m \in M$ werden als Index angegeben, somit als x_m für ein Merkmal m zum bestimmten Fingerprint x bzw. für alle möglichen Ausprägungen des Merkmals bei X_m. Die Merkmale wurden in Kapitel 3 vorgestellt und sind auf Seite 76 aufgelistet.
In der Feldstudie wurde kein Login-Status bei sozialen Netzwerken geprüft, keine Performance-Tests durchgeführt und Schriftarten wurden nur per Flash abgerufen (keine Einzelprüfung mit CSS/JavaScript).[107]

In den folgenden Abschnitten werden folgende in der Datenbank vermerkten Eigenschaften nicht als Fingerprint Merkmale gezählt – es sei denn, sie werden explizit erwähnt.

- Die IP-Adresse wird nicht als Fingerprint Merkmal betrachtet, da diese typischerweise nicht durch das System oder den Nutzer, sondern vom

[107] Vgl. Abschnitte 3.7, 3.8 und 3.6.

$M := \{$ http_user_agent, http_accept, http_accept_charset, http_accept_encoding, http_accept_language, navigator_appCodeName, navigator_appName, navigator_cookieEnabled, navigator_language, navigator_platform, color_depth, devicePixelRatio, timezone_offset, java_enabled, color_activeborder, color_activecaption, color_appworkspace, color_background, color_buttonface, color_buttonhighlight, color_buttonshadow, color_buttontext, color_captiontext, color_graytext, color_highlight, color_highlighttext, color_inactiveborder, color_inactivecaption, color_inactivecaptiontext, color_infobackground, color_infotext, color_menu, color_menutext, color_scrollbar, color_threeddarkshadow, color_threedface, color_threedhighlight, color_threedlightshadow, color_threedshadow, color_window, color_windowframe, color_windowtext, plugin_flash, plugin_adobe_acrobat, plugin_silverlight, plugins, mimetypes, fonts $\}$.

Access Provider festgelegt wird. Eine IP-Adresse kann jedoch, wie in Kapitel 5.4.2 beschrieben, vor allem innerhalb eines kurzen Zeitfensters aussagekräftig sein.

- Die TCP-Portnummer kann als Merkmal ausgeschlossen werden, da der Wert nicht stabil bleibt und nach nur kurzer Zeit keine identifizierenden Informationen mehr bereitstellt (vgl. Tabelle 3, Seite 92).

- Die aufrufende Adresse (HTTP-Referer) ist in dieser Feldstudie nicht von Bedeutung, da das Projekt in zu wenig anderen Websites eingebunden wurde. Bei Werbenetzwerken kann dieses Merkmal jedoch dazu dienen, Nutzer in gewisse Gruppen einzuteilen (z. B. Sprachen der Websites, die den Code eingebunden haben und vom Nutzer aufgerufen wurden).

- Die (verfügbare) Bildschirmgröße wurde hier nicht betrachtet, da ins-

besondere beim Vergleich minimale Abweichung als anderer Fingerprint gewertet würden. Es müsste daher ein entsprechender Bewertungsalgorithmus erstellt werden, der jedoch aus Platzgründen nicht in dieser Diplomarbeit eingeführt werden kann.

Es existiert neben den o. g. Merkmalen X_{id}, der bei der Feldstudie die Datensatz-ID angibt, sowie X_{uid} für die eindeutige Geräte-/Browserkennung. Ebenso sei eine Fingerprint-Funktion $f(\cdot)$ gegeben, für die gilt:

$$f : \{x \mid x \in X\} \mapsto fp,$$

wobei fp eine eindeutige Kennung ist. Gleiche Fingerprints bilden auf identische Kennungen ab:

$$f(x) = f(y), \quad \text{falls} \quad \forall m \in M : x_m = y_m.$$

5.2.5 Rohdaten und Bereinigung der Datensätze

Die gesammelten Daten der Feldstudie werden in einer Menge F zusammengefasst. Es handelt sich hierbei um alle Daten, die weder modifiziert noch nach bestimmten Kriterien aussortiert worden sind. Es ergibt sich:

$$|F| = 23.709.$$

In den folgenden Abschnitten wird häufig auf diese Gesamtmenge verwiesen. Dennoch macht es Sinn, die Anzahl der Datensätze nach bestimmten Vorgaben zu verringern. Fingerprints, bei denen keine aktiven Merkmale gespeichert werden konnten (d. h. die AJAX-Kommunikation schlug fehl), werden als **fehlerhafte Datensätze** bezeichnet. Als Kriterium für einen fehlerhaften Datensatz wird geprüft, ob der Eintrag für die Spalte `navigator_appCodeName` keinen Wert (`null`) enthält. Da diese Datensätze nur passive Fingerprintmerkmale

enthalten, erhöhen Sie die Gesamtmenge, ohne dass sie effektiv für Vergleiche der Merkmale genutzt werden können. F' bezeichnet alle nicht-fehlerhaften Datensätze:

$$\begin{aligned} F_e &:= \left\{x \mid (x \in F) \wedge \left(x_{navigator\_appCodeName} \neq null\right)\right\} \\ F' &:= F_e \\ \implies |F'| &= 21.504 \end{aligned}$$

In den Rohdaten in F erscheinen Datensätze, die den selben Fingerprint und die selbe Browser/Gerätekennung (uid) enthalten. Dies geschieht beispielsweise dann, wenn eine Person mehrfach an der Studie teilnimmt, ohne dass sich die Konfiguration seines Computers bzw. Browsers ändert. Um die Anzahl der Datensätze zu verringern, wird bei identischen Kombinationen von Fingerprint und uid nur das erste Vorkommen gespeichert und redundantes Vorkommen verworfen. Diese Werte sind in F_u enthalten. Aus der Schnittmenge mit F' ergibt sich F''.

$$\begin{aligned} F_u &:= F \setminus \{x \mid (x \in F) \wedge \\ &\quad (\exists y : (y \in F) \wedge (y_{id} < x_{id}) \wedge (y_{uid} = x_{uid}) \wedge (f(y) = f(x)))\} \\ \implies |F_u| &= 20.062 \\ F'' &:= F' \cap F_u \\ \implies |F''| &= 18.352 \end{aligned}$$

In einem letzten Schritt werden alle Datensätze, die den selben Fingerprint und die selbe IP-Adresse besitzen, nur als ein Datensatz gezählt. Das Vorgehen ist analog zum vorherigen Schritt. Da vor allem technikaffine Personen an der Feldstudie teilgenommen haben, ist davon auszugehen, dass diese u. U. zwischen ihren Teilnahmen ihre Cookies gelöscht haben könnten. Ein Nachteil

dieses Vorgehens ist jedoch, dass unterschiedliche Geräte mit der gleichen Konfiguration hinter einer IP-Adresse als ein Gerät erkannt werden (vgl. Hinweise zu NAT auf Seite 29). Dies ist jedoch bei Privatcomputern höchst unwahrscheinlich und nur bei Arbeitsplatzcomputern, die identisch konfiguriert sind, zu erwarten.

$$
\begin{aligned}
F_i &:= F \setminus \{x \mid (x \in F) \wedge \\
&\qquad (\exists y : (y \in F) \wedge (y_{id} < x_{id}) \wedge (y_{ip} = x_{ip}) \wedge (f(y) = f(x)))\} \\
\Longrightarrow |F_i| &= 20.629 \\
F''' &:= F'' \cap F_i \\
\Longrightarrow |F'''| &= 17.937
\end{aligned}
$$

Um diese Mengen in der SQL-Tabelle abzubilden, wurde nachträglich eine Spalte `ignorelevel` hinzugefügt. Die Werte werden durch das Skript `auswertung/set_ignorelevel.php` gesetzt. Ist `ignorelevel` $= 0$, so ist $x \in F'''$. Enthält `ignorelevel` das Bit 1, dann ist $x \notin F_e$, bei Bit 2 gilt $x \notin F_u$ und bei Bit 4 folglich $x \notin F_i$. Die Gesamtmenge F erhält man daher durch die WHERE-Bedingung `ignorelevel >= 0`.

5.3 Auswertung der User-Agent-Zeichenfolge

Für die folgenden Analysen von der Fingerprints und der Cache-Grafiken können weitergehende Informationen über den verwendeten Browser und das Betriebssystem nützlich sein. Diese sind in unterschiedlichen Formaten in der Zeichenfolge „User-Agent“ in den HTTP-Steuerdaten vermerkt. Um strukturierte Browser-/Systemdaten zu jedem Fingerprint-Datensatz nutzen zu können, wird eine Hilfstabelle `user_agents` erstellt, die zu jeder Datensatz-ID die verwendete Browserfamilie (z. B. „Safari“, „Chrome“), die verwendete Version (getrennt nach Major- und Minorangaben), die verwendete Plattform (z. B. „iOS“,

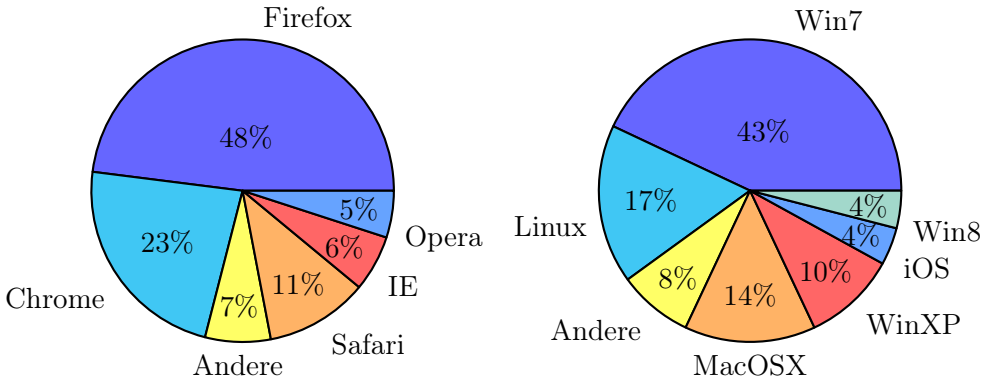


Abbildung 2: Verwendete Browser und Betriebssysteme in F

„Win7“) und weitere Informationen auflistet. Dabei ist jede Datensatz-ID dieser Hilfstabelle der entsprechenden *id* aus der Tabelle `bfp` zugeordnet (somit haben beide Tabellen die gleiche Anzahl an Zeilen). Diese Tabelle wird durch das Skript `auswertung/user_agents.php` erstellt und verwendet die PHP-Funktion `get_browser()`, die anhand einer „User-Agent“-Zeichenfolge entsprechende Daten extrahieren kann.[108]

In folgenden Abschnitten kann durch folgende Abbildung auf die Browserbezeichnung zugegriffen werden:

$$browser : \{s \mid s \in X_{id}\} \mapsto String$$

Es sei angemerkt, dass sich die „User-Agent“-Zeichenfolge durch den Nutzer anpassen lässt, die gesammelten Angaben somit nicht vollständig die verwendeten Betriebssysteme und Browser widerspiegelt. Wie Abbildung 2 zeigt, enthalten fast die Hälfte aller auswertbaren „User-Agent“-Zeichenfolgen Hinweise auf den Firefox-Browser.[*7] Den größten Anteil unter den Betriebssystemen stellt Windows 7.[*8] Mobile Browser konnten nur bei 1.754 Datensätzen festgestellt

[108] Informationen zur verwendeten Funktion auf `http://php.net/manual/de/function.get-browser.php` (besucht am 15. März 2013).

werden.*9

5.4 Unterschiedlich- und Einzigartigkeit der Fingerprints

Um Clients durch Browser Fingerprints wiedererkennen zu können, sind möglichst seltene Fingerprintmerkmale wünschenswert, beispielsweise das Vorhandensein kaum verwendeter Schriftarten, seltene Browserplugins oder auch die Verwendung von untypischen Systemfarben. Ein **Fingerprintmerkmal ist einzigartig**, wenn dessen Wert genau einmal in der betrachteten Menge an Fingerprints vorkommt. Ein **Fingerprint ist einzigartig**, wenn eins seiner Merkmale einzigartig ist oder wenn eine Kombination von n Merkmalen einzigartig ist. Dies gilt für eine durch die Feldstudie relativ kleine Menge F (bzw. die daraus abgeleiteten Mengen F', F'' und F'''), für globale Aussagen aller verwendeten Browser weltweit ist dies kaum möglich (vgl. ECKERSLEY: „How Unique Is Your Web Browser?“, S. 9 f.).

5.4.1 Einzigartigkeit einzelner Merkmale

Die Merkmale mit der höchsten Anzahl an verschiedenen Werten sind die Liste der Plugins, die Liste der unterstützen MIME-Typen und die installierten Schriftarten. Die höchste Anzahl an unterschiedlichen Werten bei passiven Fingerprint-Merkmalen ist in der „User-Agent“-Zeichenfolge zu finden. Auffällig ist ebenso, dass die aktive Variante der „User-Agent“-Zeichenfolge (JavaScript: `navigator.userAgent`) in exakt 1.000 Fällen nicht mit der Angabe im HTTP-Header übereinstimmt und daraus auch unterschiedliche Werte für die Anzahl der unterschiedlichen Datensätze resultieren.*10 Dies ist vor allem auf vom Benutzer modifizierte Einträge zum „User-Agent“, die bei der JavaScript-Abfrage nicht berücksichtigt werden,[109] oder aber auf bestimmte Versionen des

[109] Vgl. z. B. Datensatz mit id = 10727.

Microsoft Internet Explorers oder Opera, bei dem sich die Angaben unterscheiden, zurückzuführen.[110] Bei den Systemfarben ergeben sich je nach Eigenschaft verschiedene Maximalwerte für die Anzahl an unterschiedlichen Werten: Die Hintergrundfarbe des Desktops wird von mehr Benutzern individuell festgelegt (429) als die dunkle Farbe für den Schatten eines 3D-Steuerelements (63). Wird allerdings eben letztgenannte Eigenschaft vom Nutzer individuell festgelegt, so kann er möglicherweise nur daran identifiziert werden – die Einzigartigkeitsquote der Werte liegt bei fast 59 %. Bei booleschen Werten, wie der Abfrage, ob Java aktiviert ist, oder ob Cookies unterstützt werden, kann es folglich nur zwei Werte geben. Ebenso existiert auch für `navigator.appCodeName` nur der Wert „Mozilla“ oder `null`, falls dieser – möglicherweise durch einen AJAX-Fehler bedingt – nicht abgerufen werden konnte.

Bei Plugins, MIME-Typen und Schriftarten ist der Anteil der einzigartigen Datensätze sehr hoch. Dies bedeutet, dass es im Falle der Plugins 11.636 Datensätze gibt, die bei diesem Merkmal eindeutig sind. Wiederum 649 Fingerprints haben bei diesem Merkmal einen identische Wert. Dieser identische Wert stammt vom iPhone, da in der Liste der installierten Plugins lediglich „QuickTime“ aufgeführt wird.[111] Da das iPhone keine Installation von weiteren Plugins zulässt, sind diese Geräte anhand des Plugin-Auflistung nicht unterscheidbar.

Einen Überblick über die Anzahl der unterschiedlichen und einzigartigen Werte liefert Tabelle 1.

[110] Vgl. z. B. Datensätze mit id = 9783 oder id = 70.

[111] Dies betrifft iOS 6, das aktuelle Betriebssystem des iPhones. In der Version 5 wurde zusätzlich noch das „YouTube Plug-in“ aufgelistet.

Merkmal	*Verschiedene F*[1]	*Verschiedene F'''*[1]	*in %*[2]	*Max*[3]	*Einzigartig*[4]	*in %*[5]
uid	18692	16968	94,6	13	16363	96,43
plugins_hash	12501	12500	69,69	649	11636	93,09
mimetypes_hash	12231	12231	68,19	649	11207	91,63
fonts_hash	11004	10954	61,07	181	9877	90,17
navigator_userAgent	2497	2497	13,92	1458	1670	66,88
http_user_agent	2236	2196	12,24	1440	1402	63,84
navigator_appVersion	1355	1355	7,55	6200	842	62,14
http_accept_language	611	606	3,38	6291	402	66,34
color_background	429	429	2,39	6643	271	63,17
screen_height	323	322	1,8	3305	176	54,66
screen_width	223	222	1,24	5205	114	51,35
http_accept	141	136	0,76	15417	72	52,94
plugin_flash	117	117	0,65	5639	33	28,21
document_referrer	121	73	0,41	16247	44	60,27
color_threeddarkshadow	63	63	0,35	7019	37	58,73
navigator_language	57	57	0,32	8721	27	47,37
plugin_adobe_acrobat	53	53	0,3	4011	7	13,21
navigator_platform	49	49	0,27	10844	23	46,94
plugin_silverlight	31	31	0,17	4777	4	12,9
timezone_offset	25	25	0,14	17294	4	16
http_accept_encoding	19	19	0,11	12091	4	21,05
devicePixelRatio	18	18	0,1	9625	7	38,89
navigator_appName	9	9	0,05	15647	4	44,44
java_enabled	2	2	0,01	11590	0	0
navigator_appCodeName	2	1	0,01	17937	0	0

[1] Die Anzahl der verschiedenen Merkmale in F bzw. F'''.
[2] Die Anzahl der verschiedenen Merkmale (relativ zu $|F'''|$).
[3] Anzahl d. Auftretens eines bestimmten Wertes mit größter Häufigkeit in F'''.
[4] Anzahl der Werte, die exakt einmal in F''' vorkommen.
[5] Die Anzahl der einzigartigen Werte relativ zu „Verschiedene F'''“.

Tabelle 1: Auftreten unterschiedlicher/einzigartiger Merkmale

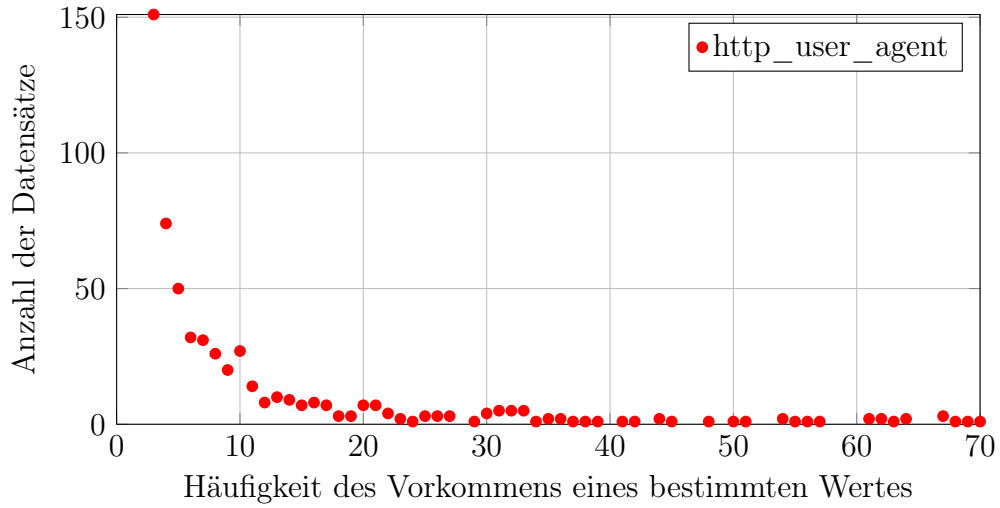


Abbildung 3: Werteanzahl/Häufigkeit-Relation (Ausschnitt)

5.4.2 Einzigartigkeit der passiven Fingerprintmerkmale

Da für die Betrachtung der passiven Merkmale die Menge F_e nicht relevant ist, werden im Folgenden Schlüsse für F_p gezogen.

$$\begin{aligned} F_p &:= F_i \cap F_u \\ \Longrightarrow |F_p| &= 18.861 \end{aligned}$$

Dies entspricht der Menge aller Fingerprints, die nicht aufgrund ihrer IP oder der *uid* mehrfach vorkommen.[112] Werden alle passiven Fingerprintmerkmale, gespeichert unter den Spaltenbezeichnungen „http_user_agent“, „http_accept“, „http_accept_charset“, „http_accept_encoding“, „http_accept_language“, gruppiert, existieren 4.352 verschiedene Einträge.*[11] Dabei kommt eine Kombination sogar 1.168-mal vor. Insgesamt existieren 197 Konfigurationen, die mehr als 10-mal in F erscheinen.*[12] 3.082 (16,3 % von F_p) Konfiguratio-

[112] Es handelt sich damit um die Datensätze, bei denen die Bits 2 und 4 in `ignorelevel` nicht gesetzt sind.

nen sind einzigartig und existieren genau einmal.*[13] Wird nur nach der „User-Agent“-Zeichenfolge betrachtet, liegt der Wert bei 1.403.*[14]

5.4.3 Einzigartigkeit durch Kombination mehrerer Merkmale

Allein durch die Auswertung der Plugin-Informationen sind 11.636 Datensätze in F''' (64,9 %) eindeutig zu identifizieren. Dieser Wert kann durch die Hinzunahme weiterer Merkmale erhöht werden. Dabei können die Angaben aus Tabelle 1 selbstverständlich nicht addiert werden, da die einzelnen Merkmale untereinander abhängig sind. In Tabelle 2 werden die Merkmale, ausgehend von Tabelle 1 kombiniert, bis schließlich nach allen Merkmale (inkl. Bildschirmauflösung) gruppiert wird. Diese Vorgehensweise ist mitunter nicht optimal. So ist es möglich, dass eine Kombination von n Merkmalen in anderer Reihenfolge höhere Werte liefern könnte.

Es existieren bei der Feldstudie **16.605 einzigartige Fingerprints**. Da sich diese Werte auf die Menge F''' beziehen (zu erkennen an `ignorelevel = 0`, Tabelle 2), liegt die Quote bei **92,57 %**. Bei der Verwendung von vier Merkmalen (Liste der Plugins, Unterstütze MIME-Typen, Liste der Schriftarten und die „User-Agent“-Zeichenfolge) lassen sich 15.556 einzigartige Datensätze ermitteln. Dies entspricht einer Quote von 86,73 % mit Bezug zu F'''.

Rechnet man die cookiebasierten Gerätekennungen (*uid*) bei der Gruppierung hinzu, erhöht sich die Anzahl der einzigartigen Datensätze mit Bezug zu den Plugin-Informationen auf 16.913. Dies bedeutet, dass identische Plugin-Informationen nicht nur als ein Vorkommen berechnet werden, sondern pro *uid* einmal. Unter Bezugnahme aller Fingerprint-Merkmale, kombiniert mit der *uid*, ergeben sich folglich 17.937 Datensätze (100 % von F''').

Der Internet Explorer unterstützt die Eigenschaften `navigator.plugins` und `navigator.mimetypes` nicht. Da diese sich jedoch als sehr aussagekräftige Merkmale zur Identifizierung von Fingerprints herausgestellt haben, wird im

SELECT COUNT(*) AS c FROM bfp WHERE ignorelevel = 0 GROUP BY ... HAVING c = 1	
plugins_hash	11636
..., mimetypes_hash	12081
..., fonts_hash	14247
..., navigator_userAgent	15556
..., http_user_agent	15620
..., http_accept_language	15786
..., color_background	15841
..., color_highlight	15851
..., screen_height	16335
..., color_activeborder	16351
..., color_menu, color_activecaption, color_inactivecaption, color_buttonshadow, color_graytext, color_threedshadow, color_buttonface, color_threedface	16377
..., screen_width, color_threedlightshadow, color_scrollbar, color_inactivecaptiontext, color_inactiveborder, color_appworkspace	16407
..., color_infobackground, color_window	16409
..., http_accept, color_windowframe, color_buttonhighlight, color_threedhighlight, color_windowtext	16423
..., color_captiontext, color_menutext, color_infotext, color_buttontext, color_highlighttext, color_threeddarkshadow	16427
..., plugin_flash	16452
..., navigator_language	16466
..., navigator_platform	16469
..., http_accept_charset	16473
..., plugin_adobe_acrobat	16475
..., devicePixelRatio, plugin_silverlight	16497
..., timezone_offset, navigator_appName	16540
..., http_accept_encoding, color_depth, navigator_appCodeName	16560
..., navigator_cookieEnabled, java_enabled	16605

Tabelle 2: Einzigartigkeit von Fingerprints durch Kombinationen

Folgenden die Menge der Fingerprints betrachtet, die von dem Internet Explorer stammen. Damit Einträge mit geänderten „User-Agent“-Zeichenfolgen ausgeschlossen werden, muss sichergestellt werden, dass die beiden Merkmale tatsächlich `null` sind.[*15]

$$\begin{aligned} F_{ie} &:= \{x \mid (x \in F''') \wedge (browser\,(x_{id}) = \text{„IE“}) \wedge \\ &\qquad (x_{plugins} = null) \wedge (x_{mimetypes} = null)\} \\ \Longrightarrow |F_{ie}| &= 1.122 \end{aligned}$$

Werden als Merkmal lediglich die Schriftarten berücksichtigt, so haben 804 Datensätze, also 71,66 % von F_{ie}, einzigartige Werte. Da in Abschnitt 5.4.2 gezeigt wurde, dass die passiven Merkmale kaum aussagekräftig sind, könnte die Schlussfolgerung entstehen, dass der Internet Explorer kaum eindeutigen Fingerprints liefert, sofern auch das Auslesen der Schriftarten durch Deaktivierung von Java, Flash, etc. verhindert würde. Dies ist jedoch nicht der Fall. Die „User-Agent“-Zeichenfolge beim Internet Explorer ist, verglichen mit anderen Browsern, sehr unterschiedlich und aussagekräftig. 539 Datensätze, 48,04 % von F_{ie}, haben ein einzigartiges „User-Agent“-Merkmal, wenn die JavaScript-Variante `navigator.userAgent` ausgewertet wird.[*16] Der Internet Explorer überträgt in der Zeichenfolge Plugin-Informationen, die diese Zeichenfolge sehr schnell einzigartig werden lässt. Bei der passiven Angabe im HTTP-Header sind es 309 Fingerprints.

Werden Schriftarten und `navigator_userAgent` kombiniert, sind 994 Datensätze (88,6 %) einzigartig. Vorteile bzgl. des Datenschutzes liefert der Internet Explorer daher nicht. Im Gegenteil: Kein anderer Browser liefert bei Ausnutzung dieser beiden Merkmale eine so hohe Einzigartigkeitsquote.

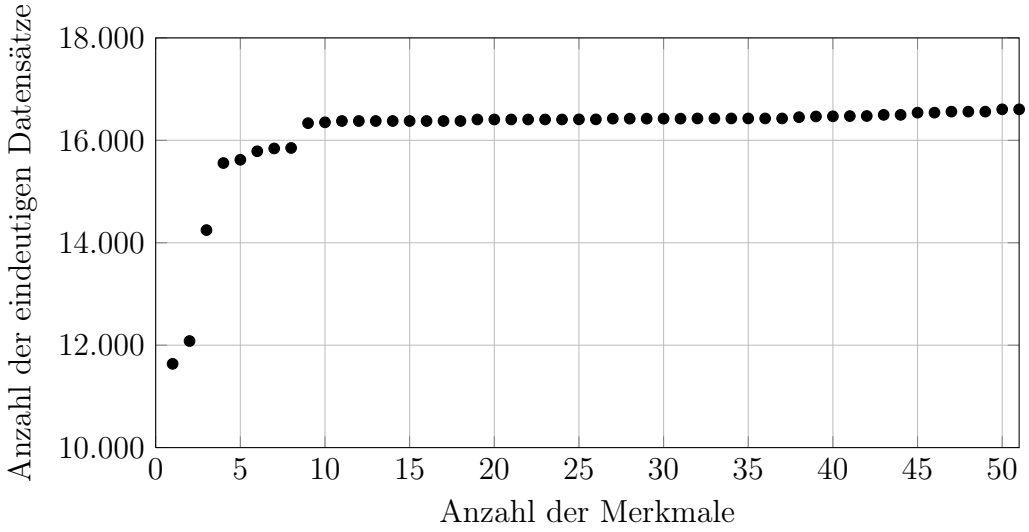


Abbildung 4: Einzigartigkeit durch Kombinationen

5.5 Veränderung des Fingerprints

Ein Ziel der Feldstudie ist die Ermittlung der Veränderung der Fingerprints (also Veränderungen der Konfiguration der Clientcomputer). Da 30 Tage lang daran teilgenommen werden konnte, war das Zeitfenster im besten Fall (Nutzer nimmt am ersten und letzten Tag teil) fast um das Doppelte länger als bei „Panopticlick“ der EFF (siehe Kapitel 3.9).

5.5.1 Vorbemerkungen

Um die Veränderung der Fingerprints über Zeit besser analysieren zu können, wird eine Hilfstabelle `uid_changes` erstellt. Dies geschieht durch das Skript `auswertung/erstelle_uid_changes.php`. Dieses Skript geht wie folgt vor (als Grundlage dient die Menge F):

1. Ermittle alle eindeutigen UIDs, die mindestens zweimal vorkommen.

2. Für all diese UIDs: Ermittle alle Fingerprints mit entsprechender UID.

2.1. Falls es sich um den ersten nicht-fehlerhaften Datensatz handelt, setze diesen Datensatz als Referenzdatensatz und springe zum nächsten.

2.2. Falls es bereits einen Datensatz an diesem Kalendertag gab, überspringe diesen Datensatz.

2.3. Falls es sich um einen fehlerhaften oder manipulierten Datensatz handelt, überspringe diesen Datensatz.

2.4. Vergleiche die Fingerprintwerte des aktuellen Datensatzes mit dem Referenzdatensatz und vermerke mögliche Änderungen.

2.5. Setze den Referenzdatensatz auf den aktuellen Datensatz.

Als **fehlerhafte Datensätze** werden Fingerprints verstanden, bei dem die AJAX-Kommunikation nicht funktionierte (in diesem Fall, wenn `navigator.appCodeName` nicht ermittelt werden konnte und in der Tabelle `null` vermerkt ist, vgl. die Menge F_e). Als **manipulierter Datensatz** werden Fingerprints identifiziert, die von unterschiedlichen Plattformen ermittelt worden sind (z. B. „Win32“ und „Linux x86_64“).[113] Manuelle Änderungen der „User-Agent“-Zeichenfolge, wie es manche Browser nativ oder über Plugins ermöglichen, werden nicht als manipulierte Datensätze interpretiert, obwohl es in der tatsächlichen Nutzung im Internet vermutlich selten vorkommt, dass ein Nutzer eine Seite mehrfach über einen Browser mit grundlegend verschiedenen „UserAgent“-Zeichenfolgen besucht.

Eine weitere Tabelle `uid_changes_multiple` dokumentiert ebenso die Änderungen der Fingerprints mit dem Unterschied, dass mehrfache Einträge pro Kalendertag nicht ignoriert, sondern ebenfalls einbezogen werden. Diese Daten

[113] Hier sei u. a. die UID „50bf6b4fd833a3.61083197“ erwähnt. Vermutlich wurde hier das Cookie von einen Betriebssystem/Browser in einem anderen Betriebssystem/Browser erneut verwendet. Eine einfache, manuelle Änderung der „User-Agent“-Zeichenfolge ist aufgrund weiterer Änderungen äußerst unwahrscheinlich.

werden hier aber nicht weiter betrachtet, da die Änderungen am selben Kalendertag vermutlich eher der Experimentierfreudigkeit der Teilnehmer geschuldet sind und keine alltagsüblichen Fingerprintveränderungen dokumentieren. Generell ist davon auszugehen, dass technisch versierte Teilnehmer der Studie die Werte in ihrem Browser bzw. Betriebssystem geändert haben, um unterschiedliche Fingerprints zu erzeugen. Daher ist im alltäglichen Einsatz eher von weniger Änderungen auszugehen.

In beiden Tabellen werden die Veränderungen über alle Spalten der Fingerprints zu einer bestimmten UID gesammelt, sowie

- das Datum des ersten und letzten Fingerprints der UID,
- die Anzahl des Vorkommens der UID,
- die Anzahl der fehlerhaften Datensätze,
- bei Schriftarten, Liste der Plugins und unterstützen MIME-Typen sowohl alle Veränderungen (inkl. `null`, tritt bei Fehlern auf oder bei Nichtverfügbarkeit von Flash), als auch nur Veränderungen ohne `null`-Datensätze. In der Auswertung werden nur die Datensätze betrachtet, bei denen kein `null` auftritt.

Wenn das Fingerprint-Merkmal einen anderen Wert trägt als beim Referenzdatensatz, so gilt dies immer als Veränderung. Diese werden hier als vergesslich angesehen, d. h. dass die Spalteneinträge „A“, „B“, „A“ als zwei Veränderungen angesehen werden, obwohl im dritten Fall wieder der erste Wert angenommen wird.

5.5.2 Auswertung

Insgesamt wurden 1.189 UIDs erkannt, die nach den o. g. Bedingungen mindestens an zwei unterschiedlichen Kalendertagen genutzt wurden. Davon wurden

bei 704 keine Veränderung eines der Fingerprint-Merkmale erkannt, wenn erneut an der Feldstudie teilgenommen wurde. Bei 156 Datensätzen wurde genau eine Veränderung festgestellt. Ab 8 Veränderungen ist die Anzahl der Datensätze nur noch einstellig. Weitere Werte sind in Abbildung 5 dargestellt.[114] Wird die Bildschirmgröße mitgerechnet, verbleiben 656 Einträge, bei denen im Laufe der Zeit keine Veränderung des Fingerprints eintritt.*[17].

Wie zu erwarten war, ändern sich vor allem die IP-Adresse und die Port-Nummer häufig. Bei fast durchschnittlich jedem dritten (29,36 %) erneuten Vorkommen eines Datensatz mit bekannter *uid* wechselte die IP-Adresse. Da diese beiden Werte jedoch nicht zu den Fingerprintmerkmalen gerechnet werden, sind deren Änderungen nicht relevant.

Mit weitem Abstand folgen tatsächliche Merkmale, beginnend mit der „User-Agent"-Zeichenfolge, der Liste der Plugins, der MIME-Typen und der Bildschirmauflösung. Bei der Liste der Plugins ist zu beachten, dass die gespeicherte Übersicht auch Versionsnummern der Plugins enthält. Bei einer Aktualisierung eines Plugins gilt dies als eine Änderung der gesamten Liste.

Die Liste der Schriftarten ändert sich im Durchschnitt bei nur 2,15 % der Datensätze mit gleicher *uid*.

Innerhalb des relativ kurzen Beobachtungszeitraum lässt sich feststellen, dass bei knapp 60 % der wiederkehrenden Teilnehmer keine Veränderung der Konfiguration festzustellen war. Bei knapp 90 % wurden maximal drei Veränderungen verzeichnet.

5.6 Vergleich von Fingerprints

Im Rahmen dieser Diplomarbeit kann aus Platzgründen lediglich ein naiver Algorithmus zum Vergleich von Fingerprints vorgestellt werden. Dieser ist nicht

[114] Da die Eigenschaften `navigator.appVersion` und `navigator.userAgent` vom HTTP-Header „UserAgent" abhängen, werden diese nicht mitgezählt.

Merkmal	AVG[1]	SD[2]	MAX[3]	SUM[4]
port*	52,59 %	27,76 %	96,15 %	3375
ip*	29,36 %	30,77 %	95,83 %	1998
http_user_agent	6,29 %	14,49 %	80,00 %	311
plugins_notnull	5,98 %	13,94 %	100,00 %	324
mimetypes_notnull	4,25 %	12,24 %	100,00 %	215
screen_height	3,58 %	12,22 %	75,00 %	218
screen_width	3,51 %	12,13 %	75,00 %	216
navigator_appVersion	3,29 %	11,14 %	66,67 %	162
fonts_notnull	2,63 %	9,84 %	72,73 %	141
plugin_flash	2,15 %	8,21 %	50,00 %	122
java_enabled	0,96 %	5,29 %	50,00 %	52
plugin_silverlight	0,85 %	5,36 %	50,00 %	50
plugin_adobe_acrobat	0,55 %	4,02 %	50,00 %	40
color_menu	0,49 %	5,48 %	80,00 %	27
http_accept_encoding	0,38 %	4,19 %	66,67 %	25
http_accept_language	0,29 %	3,44 %	66,67 %	14
http_accept	0,26 %	3,23 %	50,00 %	9
color_inactivecaptiontext	0,18 %	2,94 %	50,00 %	5
color_background	0,17 %	2,90 %	50,00 %	4
http_accept_charset	0,15 %	2,61 %	50,00 %	4
devicePixelRatio	0,10 %	1,57 %	33,33 %	7
color_windowtext	0,04 %	1,45 %	50,00 %	1
timezone_offset	0,01 %	0,48 %	16,67 %	1
navigator_appCodeName	0,00 %	0,00 %	0,00 %	0

[1] Durchschnittliche Veränderungen zwischen zwei Fingerprints
[2] Standardabweichung
[3] Höchstwert von Veränderungen, die zu einer *uid* gehören
[4] Summe aller verzeichneten Veränderungen in `uid_changes`
[*] Zählen nicht zu den Fingerprintmerkmalen

Tabelle 3: Veränderungen der Merkmale (Auswahl)

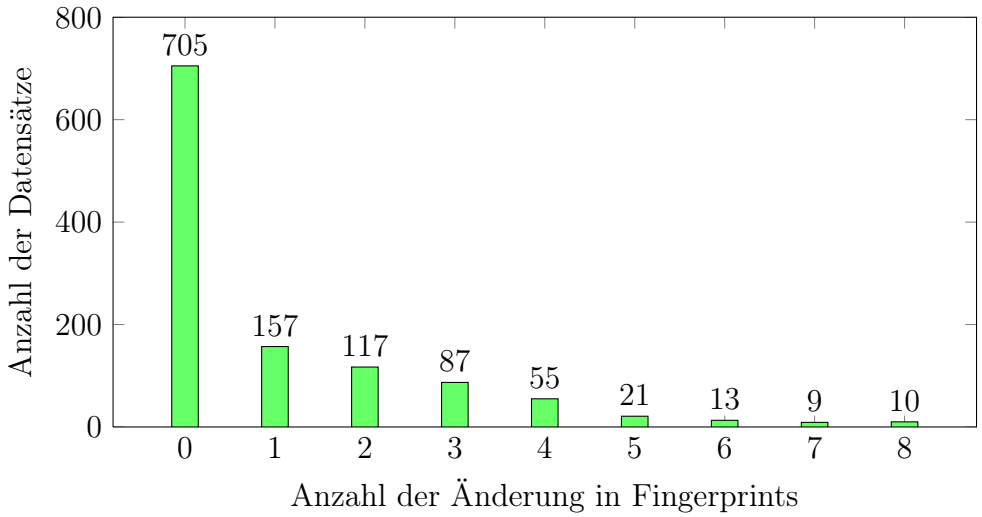


Abbildung 5: Durchschnittliche Anzahl an Änderungen der Fingerprints

effizient und geht nicht auf besondere Merkmale ein, die sich stärker oder schwächer als andere ändern.

$$compareFingerprints : \{(X, Y) \mid X, Y \in F\} \mapsto \{z \mid 0 \leq z \leq 1\}$$

```
1  function compareFingerprints(X, Y):
2      z := 1;
3      for (i := 0 to sizeof(X)):
4          x := X_i; y := Y_i;
5          if (x ≠ null ∧ y ≠ null):
6              if (typeof(x) == String):
7                  z := z * compare_strings(x, y);
8              else:
9                  if (x ≠ y): z := z * 0.95;
10     return z;
```

Listing 20: Vergleich zweier Fingerprints (Pseudocode)

Der Algorithmus nutzt die Pseudo-Funktion `compare_strings`, die zwei Zeichenfolgen miteinander vergleicht und den Grad der Gleichheit zwischen 0 (keine Ähnlichkeit) und 1 (identischer Text) wiedergibt. Sollten alle Merkmale identisch sein, behält z seinen Anfangswert 1. Sollte eine Zeichenfolge abweichen, wird z mit dem Rückgabewert von `compare_strings` multipliziert. Bei anderen Werten wird im Falle einer Abweichung z mit $0,95$ multipliziert. Dieser Wert ist frei gewählt und dient lediglich der Verkleinerung von z, falls ein Datum abweicht.

5.6.1 Vergleich einiger Fingerprints

Theoretisch könnte jeder gesammelte Fingerprint aus F mit jedem anderen verglichen werden. Durch die Mächtigkeit von $|F| = 23.709$ ergibt sich somit, dass der erste Datensatz mit 23.708, der zweite mit 23.707, usw. verglichen werden muss. Insgesamt wären dies 281.022.778 Vergleiche. Da der vorgestellte Algorithmus durch den Vergleich der Zeichenfolge nicht performant ist, wird hier die Komplexität zunächst minimiert.

Interessant sind insbesondere Datensätze, die den Kriterien für F''' genügen und dessen *uid* mindestens ein weiteres Mal erscheint:

$$\begin{aligned} F_c &:= \{x \mid (x \in F''') \wedge (\exists y \in F''' : (x \neq y) \wedge (x_{uid} = y_{uid}))\} \\ \Longrightarrow |F_c| &= 1.986 \end{aligned}$$

Auch bei F_c wären immerhin noch 1.969.120 Vergleiche notwendig, weswegen hier nur für die ersten 50 Datensätze ein Vergleich mit allen anderen aus F_c durchgeführt wird. Die Tabelle „comparison“ umfasst 98.025 Einträge. Neben den Rückgabewerten von `compareFingerprints()` (mit den Fingerprints x und y) wurde ebenso vermerkt, ob die verglichenen Fingerprints die identische UID oder IP-Adresse besitzen. Ebenso wurde gespeichert, ob die PNG-ID (siehe Abschnitt 5.7) übereinstimmt. In einer weiteren Spalte wurde die Differenz der

`null`-Einträge vermerkt. Dies bedeutet, falls entweder x oder y bei einem der Merkmale einen `null`-Wert enthält, wurde dieser Zähler um eins erhöht.

Insgesamt wurden 65 Vergleiche von Fingerprints durchgeführt, dessen *uid* identisch war.*[18] Die folgende Tabelle bezieht sich auf die Rückgabewerte von `compareFingerprints` und listet den kleinsten, größten und durchschnittlichen Rückgabewert der Funktion, unterteilt in Vergleiche mit identischer oder unterschiedlicher *uid*. Die letzte Spalte gibt die Standardabweichung an:

uid	**Anzahl**	**MIN**	**MAX**	**AVG**	**SD**
Identisch	65	0,0274	1	0,8278	0,2419
Unterschiedlich	97.960	0	1	0,0211	0,0797

Tabelle 4: Fingerprint-Vergleiche

Ein Rückgabewert von 0,7 entspricht ungefähr dem Durchschnittswert von Vergleichen mit identischer *uid* abzüglich der Standardabweichung und sei hiermit der Schwellwert: Liefert `compareFingerprints` für die Fingerprints x und y einen Wert $\geq 0,7$, so wird angenommen, dass es sich um einen identischen Nutzer handeln könnte, sonst um unterschiedliche. Durch diesen Schwellwert werden 293 Fingerprint-Paare als von einem identischen Client stammend gewertet, obwohl die *uid* unterschiedlich sind.*[19] Damit könnten 0,003 % sog. „false-positives“ sein. Zwei dieser möglichen „false-positive“ Vergleiche haben jedoch einen einen Rückgabewert von 1 (identische Fingerprints) oder 0,966 und eine identische IP-Adresse. Es ist daher davon auszugehen, dass hier lediglich der Cookie gelöscht wurden.

93 verglichene Fingerprints sind identisch (Rückgabewert 1), haben jedoch keine gemeinsame *uid* oder IP-Adresse. Hier ist zu prüfen, ob es tatsächlich unterschiedliche Clients sind oder auch hier der Cookie gelöscht wurde. Insbesondere bei Geräten, die wenig bis nicht zu konfigurieren sind (Smartphones oder Tablets) kann keine Auskunft getroffen werden.

Plattform	Browser	# Vergleiche	Unterscheidung?
Android	Android Browser 4.0	2	Nein
Android	Chrome 18.0	1	Nein
iOS	Mobile Safari 5.1	5	Nein
iOS	Mobile Safari 6.0	7	Nein
WinXP	IE 8.0	1	Ggf. möglich
MacOSX	Safari 5.1	8	Ggf. möglich
MacOSX	Safari 6.0	69	Ggf. möglich

Tabelle 5: Identischen Merkmale aber unterschiedlicher *uid*

Bei den Fingerprints mit Desktop-Betriebssystemen ist nun zu prüfen, wie „individuell“ die entsprechenden Fingerprint-Merkmale sind. Handelt es sich bei den Fingerprints um Geräte mit Standardkonfiguration, so sind diese nicht unterscheidbar. Sind die Fingerprints jedoch relativ speziell (z. B. weil eine seltene Schriftart installiert ist), ist es möglicherweise der gleiche Nutzer mit neuer *uid* und anderer IP-Adresse. Aus Platzgründen wird das Vorgehen hier nicht weiter vertieft, sondern soll lediglich als Ansatz zum Identifizieren von identischen Clients dienen.

14 von 65 Fingerprintvergleiche (21,5 %) haben einen Rückgabewert $< 0,7$, obwohl die *uid* übereinstimmt. Von den 14 haben immerhin sechs einen Vergleichswert von $> 0,5$. Die starken Abweichungen hängen vermutlich mit dem einfachen Vergleich der Zeichenfolgen zusammen.

5.6.2 Ansatz zur Optimierung des Algorithmus

Dass bei knapp 90 % der Nutzer höchstens drei Änderungen der Konfiguration festzustellen sind und man davon ausgehen kann, dass bei der Nutzung von Fingerprints außerhalb dieser Feldstudie dieser Wert im gleichen Zeitraum noch höher ist (da es vermutlich weniger technikaffine und experimentierfreudige Nutzer gibt), könnte ein Algorithmus zur Wiedererkennung bestimmter Nutzer wie folgt beschrieben werden: Angenommen es existiert ein aktueller

Fingerprint x^* und die Menge bekannter Fingerprints F^*. Ferner existiert eine Funktion $V(x)$ mit $x \in F^*$, der zu einem gegebenem Fingerprint angibt, wie häufig ein Auftreten zu erwarten ist. Für $V(x) \to 0$ wird angenommen, dass der Fingerprint einzigartig ist. $V_m(x)$ funktioniert für ein bestimmtes Merkmal m aus der Menge der Merkmale M analog. Ebenso sei eine Funktion $A(\cdot, \cdot)$ gegeben, die entscheidet, ob ein Merkmal m_2 ausgehend von m_1 erreicht werden kann:

$$A : \{(x_{m_1}, x_{m_2}) \mid x \in F^*; m_1, m_2 \in M\} \mapsto \{0, 1\}$$

Beispiel: Wenn zwei „User-Agent"-Zeichenfolgen mit $A(\cdot, \cdot)$ verglichen werden, gibt die Funktion eine 1 zurück, wenn beide von einem Firefox-Browser stammen, die zweite jedoch eine höhere Versionsnummer trägt. Die Funktion gibt 0 zurück, wenn die erste von einem Mac OS X, die zweite jedoch von einem Windows 7 System stammt.

Prüfe, ob x^* in F^* vorkommt:

(a) Falls ja, *kann* es sich um einen wiederkehrenden Client handeln, wenn $V(x^*)$ hinreichend klein.

(b) Falls nein, suche einen ähnlichen Fingerprint y^*. Sei $M' \subseteq M$ die Menge der Merkmale, in der sich x^* und y^* unterscheiden. Es handelt sich um einen neuen Nutzer wenn $\exists m \in M' : A(y_m, x_m) = 0$. Ansonsten kann es sich um einen wiederkehrenden Client handeln, der bereits unter y^* bekannt war, insbesondere wenn $\forall m \in M' : V_m(x^*)$ hinreichend groß ist.

5.7 Verlässlichkeit der Nutzung von Cache-Grafiken zur Speicherung von Daten

Neben Fingerprinting-Daten, die nur serverseitig gespeichert werden, wurde in der Feldstudie auch eine Cache-Grafik, wie in Abschnitt 4.1 (Seite 59) beschrieben, auf dem Client abgelegt. Da diese Grafik im Cache und unabhängig von den HTTP-Cookies gespeichert wird, kann hier ein unterschiedliches Tracking durchgeführt werden. Das Vorgehen bei der Feldstudie war wie folgt:

(a) Lege beim ersten Aufruf eine Cache-Grafik an, die die aktuelle Datensatz-ID (x_{id}) als Farbinformation enthält.

(b) Lade beim erneuten Aufruf die Datensatz-ID des ersten Aufrufs aus der Cache-Grafik. Sollte dies fehlschlagen, erstelle eine neue Cache-Grafik mit der aktuellen Datensatz-ID.

Somit gilt für eine entsprechende Funktion $G(id, uid)$ mit $x_{(id,uid)} \in F$:

$$G(id, uid) = \begin{cases} \min(z), & \text{falls } \exists z : G(z, uid) = z \wedge (z, uid) \in F \wedge z < id \\ id, & \text{sonst} \end{cases}$$

5.7.1 Datenbestand

Es existieren 2.893 Datensätze bei der die ID aus der Cache-Grafik kleiner als die aktuelle Datensatz-ID ist.[*20] Bei 2.426 ($\approx 84\,\%$) Datensätzen entspricht $G(id, uid) < id$, d. h., dass der Datensatz, auf den die Cache-Grafik verweist, die selben UID trägt.[*21]

5.7.2 Auswertung anhand fester UID-Werte

Um eine verlässliche Aussagekraft zur Trackingmöglichkeit von Cache-Grafiken zu erhalten, kann der Referenzwert UID verwendet werden. In einer genauen

Analyse (siehe `auswertung/cache-grafiken.php`) wird dabei jeder Datensatz betrachtet und wie folgt klassifiziert:

- **Initiierende PNG-ID:** Es handelt sich bei dem Datensatz um das erste Vorkommen der UID, daher entspricht die PNG-ID der aktuellen Datensatz-ID.
- **Erstreferenzierende PNG-ID:** Es handelt sich um ein wiederholtes Vorkommen der UID, die PNG-ID verweist auf den ersten Datensatz, der die UID trägt. Dies ist das bestmögliche Verhalten bei einem erneuten Aufruf.
- **Referenzierende PNG-ID:** Es handelt sich um ein wiederholtes Vorkommen der UID, die PNG-ID verweist auf einen, jedoch nicht den ersten, Datensatz, der die UID trägt. Dies kann u a. eintreten, wenn die Browsercache, nicht aber die Cookies, gelöscht worden sind.
- **Erneuernde PNG-ID:** Die UID tritt nicht zum ersten Mal auf, die PNG-ID verweist aber auf den aktuellen Datensatz. Ein Tracking über die Cache-Grafik ist somit erst wieder ab diesem Zeitpunkt möglich.
- **Fehlerhafte PNG-ID:** Die PNG-ID ist größer als die aktuelle Datensatz-ID. Dies darf theoretisch nicht vorkommen und weist auf einen Fehler bei dem Auslesen der Grafik hin.
- **Leere PNG-ID:** Es wurde keine PNG-ID gespeichert, der entsprechende Eintrag ist somit `null`. Andere über AJAX übermittelte Daten, hier insbesondere `navigator.appCodeName`, liegen jedoch vor.
- **Fremdreferenzierende PNG-ID:** Die PNG-ID verweist auf einen Datensatz, der eine andere UID enthält. Es kann sich dabei um einen Fehler handeln oder der Nutzer kann möglicherweise die Cookies, nicht aber den

Cache gelöscht haben. Im letztgenannten Fall wurde dann eine neue UID erzeugt. Hier kann ein Vergleich der Fingerprints helfen, um diese Fälle zu bestimmen.

Bei der Auswertung wird zunächst die Menge der UID ermittelt, die mindestens zweimal verwendet wurden und dessen zugehöriger Fingerprint in F'''' liegt.*[22] Danach werden für jede UID die zugehörigen Fingerprints aus der Gesamtmenge F betrachtet und in die o. g. Kategorien eingeteilt:

Datensatzklassifizierung	**Anzahl**
Initieerend	570
Erstreferenzierend	982
Referenzierend	390
Erneuernd	608
Fehlerhaft	17
Leer	534
Nicht betrachtet (AJAX-Fehler, nicht in F')	295
Fremdreferenzierend	162

Enthält ein Datensatz eine PNG-ID, die auf einen Datensatz mit unterschiedlicher UID verweist, so wurden die Fingerprints mit `compareFingerprints()` verglichen. Dabei hatten 140 einen Wert von ≥ 0.7 und 22 einen kleineren Wert. Der größte Anteil stammt vermutlich von Teilnehmern, die ihre Cookies gelöscht haben, aber durch die PNG-ID dennoch als wiederkehrende Teilnehmer ermittelt werden könnten.

Werden die initiierenden Werte und die Datensätze mit AJAX-Fehlern abgezogen, verbleiben 2.693 Datensätze. Von diesen verweisen 1.372 (50,9 %) auf einen älteren Datensatz mit identischer UID – in diesem Fall konnte somit ein

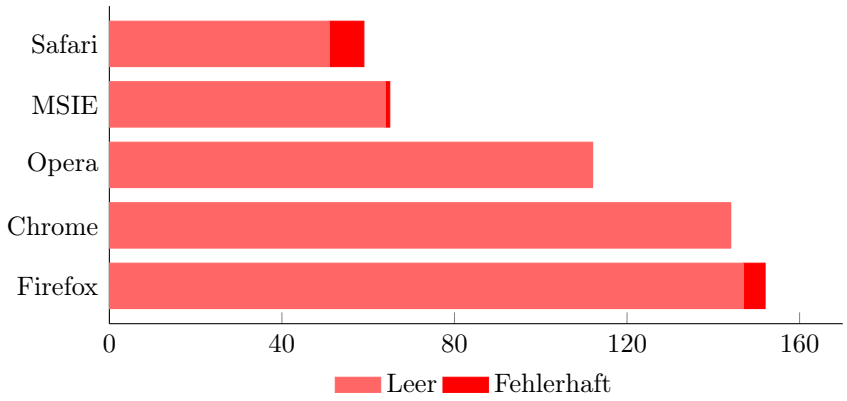


Abbildung 6: Leere und fehlerhafte Datensätze nach Browser (absolut)

erfolgreiches Tracking durchgeführt werden. Werden die Datensätze mit ähnlichen Fingerprints hinzugezogen, liegt die Quote sogar bei 56,1 %. Dennoch konnte in 43 % der Fälle die PNG-ID nicht ausgelesen werden, war fehlerhaft oder musste neu gesetzt werden.

Chrome 23 hatte relativ und absolut gesehen am häufigsten Probleme, die Grafik auszulesen. Insgesamt 124 Datensätze haben in der Spalte ein `null` vorliegen, obwohl die anderen über AJAX übermittelten Daten existieren. Die leeren und fehlerhaften Datensätze sind in Abbildung 6 nach Browserfamilie gruppiert. Da Firefox in fast jeden zweiten Fingerprint in F benutzt wurde (vgl. Abbildung 2), ist die Fehlerrate relativ gering. Zum Vergleich: Opera konnte nur in 5 % der Fingerprints nachgewiesen werden, hat aber absolut nur etwas weniger Datensätze mit `null`-Werten für die PNG-ID ergeben.

5.7.3 Bewertung

Die Verlässlichkeit des Trackings über eine Cache-Grafik ist nicht sonderlich hoch. In etwa der Hälfte der Fällte konnte die Grafik nicht oder nur fehlerhaft ausgelesen werden. Möglicherweise kann der verwendete Code optimiert werden. Vermutlich liegt das Problem aber eher bei der Grafikverarbeitung im

Browser, da das Auslesen der Grafik je nach Browserfamilie und -version unterschiedlich gut funktioniert. Das Tracking per Cache-Grafik könnte aber als zusätzlicher Mechanismus verwendet werden: Wurden HTTP-Cookies gelöscht und ist der abgefragte Fingerprint uneindeutig, könnte die gespeicherte Grafik Rückschlüsse auf den Nutzer geben.

6 Fazit: Auswirkungen und Schutzmaßnahmen

Die grundlegende These, dass viele – wenn auch nicht alle – Browser Fingerprints der Feldstudie einzigartig sind, konnte bestätigt werden. Nach Abzug der fehlerhaften und doppelten Einträge sind 92,57 % der gesammelten Fingerprints mit all ihren Merkmalen einzigartig. Als aussagekräftigste Eigenschaften stellen sich die Listen der Plugins, unterstützten MIME-Typen und Schriftarten dar. Wird dazu noch die „User-Agent“-Zeichenfolge hinzugezogen, können nur durch diese vier Merkmale 86,73 % der gesammelten Fingerprints eindeutig bestimmt werden.

Die aktiven Merkmale, die JavaScript und Flash benötigen, sind am aussagekräftigsten. Werden nur passive Merkmale betrachtet, die in jedem HTTP-Request vorkommen, so sinkt die Anzahl der einzigartigen Datensätze deutlich.

Bei 60 % der wiederkehrenden Teilnehmern (an unterschiedlichen Kalendertagen) wurde keine Veränderung der Konfiguration festgestellt. Am häufigsten änderten sich die „User-Agent“-Zeichenfolge und die Liste der installierten Plugins. Mit einem entsprechenden Algorithmus können auch diese Änderungen erkannt und abweichende Fingerprints von einem Gerät verknüpft werden.

Die eingesetzten Cache-Grafiken funktionieren nicht immer als verlässliche Trackingmethode und sind vom verwendeten Browser abhängig. Werden aber mehrere Trackingverfahren kombiniert, können Cache-Grafiken eine gute Zweit- oder Drittlösung sein.

6.1 Personenbeziehbarkeit von Browser Fingerprints

Ob Browser Fingerprints personenbeziehbare oder gar personenbezogene Daten sind, kann hier juristisch nicht abschließend geklärt werden. Es kann jedoch ein Vergleich zu bestehenden Datenschutzdiskursen hergestellt und daraus mögliche Schlüsse gezogen werden.

In der Diskussion, ob IP-Adressen personenbezogene Daten sind, wird ge-

nerell unterschieden, ob es sich um statische oder dynamische IP-Adressen handelt. Sind diese statisch, wird der Personenbezug bejaht. Bei dynamischen Adressen werden die handelnden Akteure unterschiedlich beurteilt: Für Access-Provider, also Internetzugangsanbieter, sind auch dynamische IP-Adressen personenbezogen, insofern die IP-Adresse aus dem eigenen Adresspool stammt. Dem Provider ist die Verteilung seiner aktuell vergebenen – im Falle einer Speicherung auf Vorrat auch ehemals vergebenen – IP-Adressen an einen Anschluss bekannt.[115] Inhalteanbieter haben diese Information generell nicht, können aber u. U. einen Personenbezug herstellen. Dies kann durch eine Zusammenführung der Daten geschehen oder wenn der Content-Provider weitere personenbezogene Daten sammelt und diese mit der IP-Adresse verknüpft.[116]

Obwohl das Thema mitunter kontrovers diskutiert wird, sind auch dynamische IP-Adressen u. a. für den Landesdatenschutzbeauftragten Niedersachsen personenbeziehbare Daten.[117]

Wenn IP-Adressen personenbeziehbar sind, müssten folglich auch Browser Fingerprints personenbeziehbar sein. Dies ist insbesondere dann der Fall, wenn ein Fingerprint einmalig einer personenbezogenen Information oder gar einer Person zugeordnet wird. Fortan kann, ein eindeutiger Fingerprint und/oder ein entsprechender Algorithmus, der Veränderungen an der Konfiguration erkennt, vorausgesetzt, durch den Fingerprint stets ein Personenbezug ermöglicht werden.

---

[115] Fingerprints können immer nur einer Computer-Browser-Kombination zugeordnet, nicht aber einer direkten Person zugeordnet werden, da sich mehrere Personen einen Computer teilen könnten, ohne unterschiedliche Benutzeroberflächen oder Browser zu verwenden. Ein ähnliches Problem ergibt sich auch hier: IP-Adressen können nur Anschlussinhabern zugeordnet werden, die den Zugang möglicherweise weiteren Personen zur Verfügung stellen.

[116] Vgl. Koch: *Internet-Recht: Praxishandbuch zu Dienstenutzung, Verträgen, Rechtsschutz und Wettbewerb, Haftung, Arbeitsrecht und Datenschutz im Internet, zu Links, Peer-to-Peer-Nutzern und Domain-Recht, mit Musterverträgen*, S. 950 ff.

[117] Vgl. Der Landesbeauftragte für den Datenschutz Niedersachsen: *Sind IP-Adressen personenbeziehbare Daten?*

Es ist jedoch zu vermuten, dass ein Fingerprinting-Algorithmus niemals eindeutig sein kann. Insbesondere mehrere gleiche Computer, die also nicht individuell konfiguriert sind oder sich nicht individuell konfigurieren lassen, haben identische Fingerprints (vgl. NAT). Wie die Feldstudie jedoch gezeigt hat, ist ein extrem hoher Anteil an Fingerprints eindeutig. Fingerprints sind mitunter nicht nur personenbeziehbar, sondern je nach Merkmal möglicherweise direkt personenbezogen (vgl. die Informationen, die aus der Liste der Schriftarten hervorgehen).

Werden Fingerprints, vor allem die passiven Merkmale, als personenbezogen oder -beziehbar eingestuft, enthielte jeder Webseitenaufruf Daten, bei dem das „Speichern, Verändern oder Nutzen“ nur möglich ist, wenn „der Betroffene eingewilligt hat“.[118] Greifen die anderen Punkte von § 14 Absatz 2 BDSG nicht, was besonders im Falle von Punkt 3 und 5 zu prüfen wäre, könnte in diesem Fall ein Aufruf von Webseiten, so wie bisher üblich, nicht mehr möglich sein. Selbst beim Vorschalten einer Einverständniserklärung bei jeder Website, würden bereits beim Aufruf dieser Erklärung passive Fingerprint-Merkmale mitgeschickt. Es wäre daher zu diskutieren, ob der Nutzer, allein durch die Verwendung von Webbrowsern, ein Einverständnis an den Inhalteanbieter der aufzurufenden Seite gibt, die Verarbeitung von automatisch übertragenden Informationen (passive Merkmale), die technisch durch den TCP/IP-Protokollstack und durch das HTTP-Protokoll übermittelt werden, durchzuführen. Ferner wäre zu klären, ob dies auch die Verarbeitung von aktiven Fingerprint-Merkmalen beinhaltet. Möglicherweise könnte eine Lösung darin bestehen, bestimmte Merkmale als personenbeziehbar oder gar personenbezogen (Schriftarten, u. U. auch Plugin- und MIME-Liste) zu deklarieren, anderer hingegen nicht. Doch auch hier könnte sich ein Dilemma ergeben, da der „User-Agent“ beim Microsoft Internet Explorer, verglichen mit anderen Browsern, deutlich detailliertere Informationen

[118] Vgl. § 14 Absatz 2 BDSG.

liefert.

6.2 Einsatzgebiete des Browser Fingerprintings

Browser Fingerprinting muss für den Nutzer generell nicht mit Nachteilen verbunden sein. So kann das Ermitteln und Archivieren bestimmter Merkmale der Steigerung der Sicherheit dienen. Verlangt eine Website, beispielsweise beim Online-Banking, eine Authentifizierung mit Benutzername und Kennwort, könnte jeder Login auch mit vorigen Fingerprintmerkmalen verglichen werden. Unterscheiden sich bei einem Login die aktuellen Fingerprintmerkmale deutlich von denen aus vorherigen Anmeldungen, so kann eine weitere Sicherheitsabfrage stattfinden. Selbst der in der Diplomarbeit vorgestellte naive Algorithmus zum Vergleich von Fingerprints ist dafür ausreichend.

Verwendet eine Website Cookies zur erneuten Authentifizierung (um die Eingabe von Benutzername und Kennwort zu vermeiden), so kann durch Fingerprints ein sog. „Session Hijacking“ abgefangen werden. Ein „Session Hijacking“ kann eintreten, wenn Authentifizierungsinformationen, die in einem Cookie abgelegt sind und bei einem Aufruf einer Website mitgesendet werden, von Dritten ausgelesen und verwendet werden. Entweder, weil ein Angreifer physikalischen Zugriff auf einen Computer erhalten hat oder weil durch einen „Man-In-The-Middle“-Angriff der Datenverkehr zwischen Client und Server mitgeschnitten wurde.[119] Findet ein Abgleich des Browser-Fingerprints statt, kann ein solcher Angriff möglicherweise unterbunden werden.

Für den Nutzer im Sinne der Privatsphäre eher unvorteilhaft, für Werbenetzwerke nützlich, ist es durchaus realistisch, Browser Fingerprints zum Tracking einzusetzen. Insbesondere bei der Ansammlung von vielen einzigartigen Merkmalen, sind – wie die Feldstudie gezeigt hat – Browser Fingerprints

[119] Vgl. Nagamalai et al.: „Security Threats and Countermeasures in WLAN“, S. 171 ff.

durchaus verlässlich. Mit einem ausgefeilteren Algorithmus, der Veränderungen der Merkmale effizient erkennt, kann die Bedeutung des Fingerprintings weiter erhöht werden. Je kürzer ein Zeitfenster zwischen zwei Aktionen ist, desto geeigneter sind die virtuellen Fingerabdrücke. Beispiel: Ein Nutzer sieht einen Werbebanner für ein Produkt, das er jedoch nicht anklickt. Später ruft er die Website des Produkts manuell (oder durch andere Verlinkungen) auf und kauft dieses Produkt. Vermutlich hat der Werbebanner seine Kaufentscheidung beeinflusst. Gleichen sich die Fingerprint, kann die Werbeeinblendung und der Kaufprozess verknüpft werden.

6.3 Schutzmaßnahmen

Ziel einer Schutzmaßnahme muss es sein, möglichst wenig Fingerprintmerkmale offenbaren zu können oder diese so generisch wie möglich zu halten. Das Sammeln von passiven Merkmalen lässt sich ohnehin nicht verhindern. Die „User-Agent“-Zeichenfolge kann aber bei vielen Browsern entweder nativ oder über Erweiterungen angepasst werden. Es empfiehlt sich aber, die Zeichenfolge nicht manuell zu ändern, da gerade bei einzigartigen Werten („Mein Browser 1.0“) ein eindeutiges Indentifizierungsmerkmal entsteht. Allerdings könnte, insbesondere bei Verwendung von eher untypischen Browsern, der „User-Agent“ auf einen weit verbreiteten Wert gesetzt werden. Bei der Feldstudie kam die Zeichenfolge „Mozilla/5.0 (Windows NT 6.1; WOW64; rv:17.0) Gecko/20100101 Firefox/17.0“ am häufigsten vor (1932-mal in F).[*23] Hier ist aber zu bedenken, dass Webseiten, die durch sog. „CSS-Hacks“ oder andere Anpassungen für bestimmte Browser optimiert sind, möglicherweise nicht mehr fehlerfrei dargestellt werden. Auch ist die tatsächliche Browserbezeichnung möglicherweise noch über JavaScript ermittelbar.

Aktives Fingerprinting lässt sich durch das Deaktivieren von Plugins und vor allem durch JavaScript verhindern. Dadurch kann ein erheblicher Anteil an

identifizieren Merkmalen verschleiert werden. Die Wiedererkennungsrate sinkt deutlich. Dennoch ist das Deaktivieren von JavaScript vermutlich für viele Nutzer keine Option, da eine große Anzahl an Websites, insbesondere sog. „Rich Internet Applications“, [120] clientseitiges Scripting benötigen. Systemeinstellungen, wie beispielsweise Systemfarben, sollten möglichst nicht geändert und auf Werkseinstellungen belassen werden. Insbesondere außergewöhnliche Farbkombinationen (rosa Hintergrundfarbe und gelber Text) können mitunter als vollständig identifizierbare Merkmale ausreichen.

Der Internet Explorer bietet keine Möglichkeit, die Liste aller installierten Plugins und unterstützten MIME-Typen abzufragen. Da es sich um die zwei aussagekräftigsten Merkmale handelt, kann hierdurch die Privatsphäre des Nutzers – zumindest in der Theorie – erhöht werden. Dennoch ist die Einzigartigkeitsrate bei dem Internet Explorer sehr hoch (siehe Seite 85). Die Ermittlung der Schriftarten erfordert ein Plugin, beispielsweise Java oder Flash.[121] Der Nutzer kann hier also jene Plugins deaktivieren, um die Ermittlung der Schriftarten zu verhindern. Da aber auch Flash äußerst populär und für bestimmte Websites zwingend erforderlich ist, sind Browser-Erweiterungen wie ClickToFlash[122] zu empfehlen. Hierbei wird für jede eingebundene Flashressource nur ein Platzhalter angezeigt. Durch einen Klick kann dann der tatsächliche Flashinhalt geladen und angezeigt werden. Eine weitere Variante zum Schutz bietet die Software „Fluxfonts“: der Systemdienst generiert regelmäßig zufällige Schriftarten, so dass sich das Fingerprintmerkmal stetig ändert.[123]

[120] Es handelt sich dabei um Webseiten, die ähnliche Funktionen wie „klassische“ Anwendungen anbieten und wie eigenständige Desktopprogramme aussehen und sich bedienen lassen. Hier sei als Beispiel Drag & Drop erwähnt. Siehe auch Linder: „Die Usability von Rich Internet Applications“.

[121] Zwar können über JavaScript und CSS auch bestimmte Schriftarten auf Existenz geprüft werden, eine gesamte Liste lässt sich über dieses Verfahren jedoch nicht ermitteln.

[122] Siehe `http://clicktoflash.com` (besucht am 22. April 2013).

[123] Siehe `http://web.aeyoun.priv.no/p/fluxfonts/intro.en` (besucht am 22. April 2013).

FireGloves, ein Browsererweiterung für Firefox (wird mittlerweile nicht mehr weiterentwickelt), bietet ebenfalls Schutzmöglichkeiten vor Fingerprints.[124] So kann JavaScript u. a. nicht mehr die Liste der Plugins oder unterstützten MIME-Typen auslesen.

Paradoxerweise ergeben sich aber durch solche Schutzmaßnahmen erneut Aspekte, die einen Rechner einzigartig machen. Beispiel: Wenn eine Website die Flash-Version einerseits über JavaScript als auch über Flash selbst abfragt, so gleichen sich in der Regel beide Angaben. Ist aber ein Flash-Blocker installiert, so wird eine Abfrage der Version über JavaScript möglich sein, über Flash selbst hingegen fehlschlagen. Diese Information, dass ein Flash-Blocker installiert ist, kann mit weiteren Fingerprint-Merkmalen einen Nutzer identifizierbarer machen.

6.4 Ausblick

Spätestens mit der Ankündigung Mozillas, ab Firefox 22 Drittanbieter-Cookies in den Standardeinstellung abzuweisen,[125] wird sich die Art des globalen Trackings zwangsläufig verändern. Ob dabei, so wie von der „EU-Cookie-Richtlinie“ gefordert, stets auch die Privatsphäre des Nutzers geachtet wird, bleibt abzuwarten. Problematisch bei Browser Fingerprints ist, dass diese entweder gar nicht (passive Merkmale oder binärer Code für Plugins) oder nur durch genaue Inspektion des Quellcodes (aktive Merkmale mit clientseitiger Skriptsprache) nachgewiesen werden können. Der durchschnittliche Nutzer ist, wenn er sich nicht ausgiebig mit Schutz- und Verschleierungsmechanismen auseinandersetzt, mit hoher Wahrscheinlichkeit über Browser Fingerprints wiedererkennbar. Dies betrifft vor allem Desktop-Computer. Für Betreiber von Werbenetzwerke kann

[124] Verfügbar auf: `https://addons.mozilla.org/de/firefox/addon/firegloves/` (zuletzt abgerufen am 20. Mai 2013).

[125] Vgl. Zlotos: *Firefox 22 soll Tracking-Cookies unterdrücken*.

ein Browser Fingerprint die erste Wahl sein, sollte der Client keine Speicherung von Cookies ermöglichen. Unter der Annahme, dass 20 % der Nutzer Cookies verweigern oder regelmäßig löschen, ist ein virtueller Fingerabdruck einem Cookie sogar vorzuziehen. Werden mehrere Verfahren (z. B. Cookie, Fingerprint, Cache-Grafik und ETag) kombiniert, kann die Verlässlichkeit des Trackings möglicherweise perfektioniert werden. Eine Vermutung, die den Nutzer alles andere als erfreuen wird.

7 Request For Comments (RFC) Quellenverzeichnis

[RFC1231] Ron Rivest: The MD5 Message-Digest Algorithm. April 1992. URL: http://tools.ietf.org/html/rfc1321 (besucht am 28. Februar 2013).

[RFC1945] Tim Berners-Lee, Roy T. Fielding und Henrik Frystyk: Hypertext Transfer Protocol – HTTP/1.0 (RFC 1945). Internet Engineering Steering Group. Mai 1996.

[RFC2046] Ned Freed und Nathaniel S. Borenstein: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types (RFC 2046). The Internet Society. November 1996.

[RFC20] Vint Cerf: ASCII format for Network Interchange. Oktober 1969.

[RFC2109] David M. Kristol und Lou Montulli: HTTP State Management Mechanism (RFC 2109). Februar 1997. URL: http://tools.ietf.org/html/rfc2109 (besucht am 18. April 2013).

[RFC2460] Stephen E. Deering und Robert Hinden: Internet Protocol, Version 6 (IPv6) Specification (RFC 2460). Dezember 1998.

[RFC2616] Paul J. Leach, Tim Berners-Lee, Jeffrey C. Mogul, Larry Masinter, Roy T. Fielding und James Gettys: Hypertext Transfer Protocol – HTTP/1.1 (RFC 2616). Juni 1999.

[RFC2965] David M. Kristol und Lou Montulli: HTTP State Management Mechanism (RFC 2965). Oktober 2000. URL: http://tools.ietf.org/html/rfc2965 (besucht am 18. April 2013).

[RFC4229] Jeffrey C. Mogul und Mark Nottingham: HTTP Header Field Registrations. Dezember 2005.

[RFC4646] Addison Phillips und Mark Davis: Tags for Identifying Languages (IETF RFC 4646). The Internet Society. September 2006.

[RFC4864] Brian Carpenter, Eric Klein, Tony Hain, Gunter Van de Velde und Ralph Droms: Local Network Protection for IPv6. Mai 2007.

[RFC4941] Thomas Narten, Richard Draves und Suresh Krishnan: Privacy Extensions for Stateless Address Autoconfiguration in IPv6 (RFC 4941). September 2007.

[RFC6265] Adam Barth: HTTP State Management Mechanism (RFC 6265). April 2011. URL: `http://tools.ietf.org/html/rfc6265` (besucht am 18. April 2013).

[RFC6335] Magnus Westerlund, Lars Eggert, Joseph Touch, Michelle Cotton und Stuart Cheshire: Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry (RFC 6335). August 2011.

8 Quellenverzeichnis

ADOBE: *Adobe Flash Platform runtimes*. Juli 2011. URL: http://www.adobe.com/de/products/flashplatformruntimes/statistics.html (besucht am 7. Dezember 2012).

ADOBE SYSTEMS INCORPORATED: *Fonts included with Adobe's Creative Suite 5 and Creative Suite 5.5 Applications*. URL: http://www.adobe.com/type/browser/fontinstall/cs5installedfonts.html (besucht am 23. Januar 2013).

— *Gemeinsame Objekte*. URL: http://help.adobe.com/de_DE/as3/dev/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d80.html (besucht am 17. Februar 2013).

— *getVariable()*. URL: http://help.adobe.com/en_US/Director/11.5/UsingScripting/WSc3ff6d0ea77859461172e0811d64c1a1b3-7d9f.html (besucht am 26. Januar 2013).

— *What are local shared objects?* URL: http://www.adobe.com/security/flashplayer/articles/lso/ (besucht am 17. Februar 2013).

AFP: *Americans not fans of online targeted ads: survey*. März 2012. URL: Americansnotfansofonlinetargetedads:surveyhttp://news.yahoo.com/americans-not-fans-online-targeted-ads-survey-201105738.html (besucht am 11. Mai 2013).

ALEXANDER, ALVIN: *Java fonts - how to list of all the available fonts*. August 2011. URL: http://alvinalexander.com/blog/post/jfc-swing/swing-faq-list-fonts-current-platform (besucht am 26. Januar 2013).

ANDERSEN, AARON: „History of the browser user-agent string". In: *WebAIM* (August 2008). URL: http://webaim.org/blog/user-agent-string-history/ (besucht am 6. Februar 2013).

ANGWIN, JULIA und VALENTINO-DEVRIES, JENNIFER: *Race Is On to 'Fingerprint' Phones, PCs*. November 2010. URL: http://online.wsj.com/

article/SB10001424052748704679204575646704100959546.html# (besucht am 11. Mai 2013).

Ayenson, Mika, Wambach, Dietrich James, Soltani, Ashkan, Good, Nathan und Hoofnagle, Chris Jay: *Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning*. Juli 2011. URL: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1898390.

Bachfeld, Daniel: *Adobe Flash 10.1 unterstützt "Private Browsing"*. Mai 2010. URL: http://www.heise.de/security/meldung/Adobe-Flash-10-1-unterstuetzt-Private-Browsing-994551.html (besucht am 17. Februar 2013).

Baron, L. David, Lilley, Chris und Çelik, Tantek: *CSS Color Module Level 3*. Last Call WD. World Wide Web Consortium (W3C), Juli 2008. URL: http://www.w3.org/TR/2008/WD-css3-color-20080721 (besucht am 6. Dezember 2012).

Berners-Lee, Tim: Basic HTTP as defined in 1992. W3 Project/CERN. 1991. URL: http://www.w3.org/Protocols/HTTP/HTTP2.html (besucht am 15. April 2013).

Beuth, Patrick: *Der Privatsphäre-Revolution fehlen die Revolutionäre*. Februar 2013. URL: http://www.zeit.de/digital/datenschutz/2013-02/online-tracking-studien (besucht am 11. Mai 2013).

Beverly, Robert: „A Robust Classifier for Passive TCP/IP Fingerprinting“. In: *Passive and Active Network Measurement*. Hrsg. von Chadi Barakat und Ian Pratt. Bd. 3015. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, S. 158–167. ISBN: 978-3-540-21492-2.

Böhm, Markus: *Fotofilter-App: Sicherheitslücke bei Instagram*. Dezember 2012. URL: http://www.spiegel.de/netzwelt/web/instagram-sicherheitsluecke-der-fotofilter-app-ist-das-cookie-a-870818.html (besucht am 18. April 2013).

Broenink, Ralph: „Using Browser Properties for Fingerprinting Purposes“. In: *16th biannual Twente Student Conference on IT*. Twente University Press. 2012, S. 169–176. URL: http://referaat.cs.utwente.nl/conference/16/paper/7306/using-browser-properties-for-fingerprinting-purposes.pdf (besucht am 23. Februar 2013).

Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. (BITKOM): *Internet: Mehrheit will lieber Werbung als Gebühren (Presseinformation)*. Mai 2012. URL: http://www.bitkom.org/files/documents/BITKOM_Presseinfo_Online-Werbung_14_05_2012.pdf (besucht am 11. Mai 2013).

Cardwell, Mike: *Abusing HTTP Status Codes to Expose Private Information*. Januar 2011. URL: https://grepular.com/Abusing_HTTP_Status_Codes_to_Expose_Private_Information (besucht am 17. Februar 2013).

Çelik, Tantek, Bos, Bert, Hickson, Ian und Lie, Håkon Wium: *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Candidate Recommendation. World Wide Web Consortium (W3C), September 2009. URL: http://www.w3.org/TR/2009/CR-CSS2-20090908 (besucht am 6. Dezember 2012).

de Montjoye, Yves-Alexandre, Hidalgo, Cesar A., Verleysen, Michel und Blondel, Vincent D.: „Unique in the Crowd: The privacy bounds of human mobility“. In: *Scientific Reports* (März 2013).

Der Landesbeauftragte für den Datenschutz Niedersachsen: *Sind IP-Adressen personenbeziehbare Daten?* Juni 2012. URL: http://www.lfd.niedersachsen.de/portal/live.php?navigation_id=13025&article_id=55991&_psmand=48 (besucht am 10. Mai 2013).

Duce, David: *Portable Network Graphics (PNG) Specification (Second Edition)*. W3C Recommendation. W3C, November 2003. URL: http://www.w3.org/TR/2003/REC-PNG-20031110 (besucht am 9. Februar 2013).

Dugonjić, Marko: *Detect visitor's fonts with Flash*. August 2006. URL: http://www.maratz.com/blog/archives/2006/08/18/detect-visitors-fonts-with-flash/ (besucht am 26. Januar 2013).

Eckersley, Peter: „How Unique Is Your Web Browser?“ In: *Privacy Enhancing Technologies*. Hrsg. von MikhailJ. Atallah und NicholasJ. Hopper. Bd. 6205. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, S. 1–18. ISBN: 978-3-642-14526-1.

Ford, Bryan, Srisuresh, Pyda und Kegel, Dan: *Peer-to-Peer Communication Across Network Address Translators*. Februar 2005. URL: http://www.bford.info/pub/net/p2pnat/ (besucht am 29. Januar 2013).

Grossman, Jeremiah: *I know where you've been*. August 2006. URL: http://jeremiahgrossman.blogspot.de/2006/08/i-know-where-youve-been.html (besucht am 11. Mai 2013).

Hickson, Ian: *Web SQL Database*. Techn. Ber. World Wide Web Consortium (W3C), November 2010. URL: http://dev.w3.org/html5/webdatabase/.

— *Web Storage*. Techn. Ber. World Wide Web Consortium (W3C), Dezember 2011. URL: http://www.w3.org/TR/webstorage/.

Hyatt, David und Hickson, Ian: *HTML 5*. W3C Working Draft. W3C, August 2009. URL: http://www.w3.org/TR/2009/WD-html5-20090825/ (besucht am 3. Dezember 2012).

Ihlenfeld, Jens: *IPv4: Die letzten IP-Adressen in Europa werden vergeben*. September 2012. URL: http://www.golem.de/news/ipv4-die-letzten-ip-adressen-in-europa-werden-vergeben-1209-94574.html (besucht am 29. Januar 2013).

— „IPv6: Telekom verteilt IPv6-Adressen an Kunden“. In: *golem.de* (November 2012). URL: http://www.golem.de/news/ipv6-telekom-verteilt-ipv6-adressen-an-kunden-1211-96035.html (besucht am 4. Februar 2013).

Jobs, Steve: *Thoughts on Flash*. April 2010. URL: http://www.apple.com/hotnews/thoughts-on-flash/ (besucht am 7. Dezember 2012).

Kirsch, Christian: *Firefox-Entwickler stopfen altes CSS-Leck*. März 2010. URL: http://www.heise.de/ix/meldung/Firefox-Entwickler-stopfen-altes-CSS-Leck-968556.html (besucht am 11. Mai 2013).

Koch, F.A.: *Internet-Recht: Praxishandbuch zu Dienstenutzung, Verträgen, Rechtsschutz und Wettbewerb, Haftung, Arbeitsrecht und Datenschutz im Internet, zu Links, Peer-to-Peer-Nutzern und Domain-Recht, mit Musterverträgen*. Oldenbourg Wissensch.Vlg, 2005. ISBN: 9783486578010.

Koch, Peter-Paul: *devicePixelRatio*. Juni 2012. URL: http://www.quirksmode.org/blog/archives/2012/06/devicepixelrati.html (besucht am 3. Dezember 2012).

Lammenett, Erwin: *Praxiswissen Online-Marketing*. Gabler Verlag, 2012.

Le Hégaret, Philippe, Le Hors, Arnaud und Stenback, Johnny: *Document Object Model (DOM) Level 2 HTML Specification*. W3C Recommendation. W3C, Januar 2003. URL: http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109 (besucht am 3. Dezember 2012).

Lehofer, Hans Peter: *Internetreferenzierungsdienst: EuGH beschreibt Google im Google-Urteil mit einem Wort, das Google (noch) nicht kennt*. März 2010. URL: http://blog.lehofer.at/2010/03/internetreferenzierungsdienst-eugh.html (besucht am 25. April 2013).

Linder, Jörg: „Die Usability von Rich Internet Applications“. German. In: *Social Semantic Web*. Hrsg. von Andreas Blumauer und Tassilo Pellegrini. X.media.press. Springer Berlin Heidelberg, 2009, S. 83–97. ISBN: 978-3-540-72215-1.

Mahlmann, Peter und Schindelhauer, Christian: „Das Internet — unter dem Overlay“. German. In: *Peer-to-Peer-Netzwerke*. eXamen.press. Springer Berlin Heidelberg, 2007, S. 11–54. ISBN: 978-3-540-33991-5.

MAYER, JONATHAN R.: „Any person... a pamphleteer – Internet Anonymity in the Age of Web 2.0“. Magisterarb. Stanford University, April 2009. URL: http://www.stanford.edu/~jmayer/papers/thesis09.pdf (besucht am 23. Februar 2013).

MCKINLEY, KATHERINE: *Cleaning Up After Cookies*. Dezember 2008. URL: https://www.isecpartners.com/media/11976/iSEC_Cleaning_Up_After_Cookies.pdf (besucht am 17. Februar 2013).

MICROSOFT CORPORATION: *Mit Office installierte Schriftarten (Knowledge Base 837463)*. Dezember 2007. URL: http://support.microsoft.com/kb/837463/de (besucht am 23. Januar 2013).

— *Technical Details on Microsoft Product Activation for Windows XP*. August 2001. URL: http://technet.microsoft.com/en-us/library/bb457054.aspx (besucht am 4. Februar 2013).

MICROSOFT DEVELOPER NETWORK (MSDN): *Introduction to Web Storage*. Oktober 2012. URL: http://msdn.microsoft.com/en-us/library/cc197062(VS.85).aspx#_dom (besucht am 18. Februar 2013).

— *IsVersionSupprted*. URL: http://msdn.microsoft.com/en-us/library/bb980107(v=vs.95).aspx (besucht am 22. Januar 2013).

MOWERY, KEATON, BOGENREIF, DILLON, YILEK, SCOTT und SHACHAM, HOVAV: „Fingerprinting Information in JavaScript Implementations“. In: *Proceedings of W2SP 2011*. Hrsg. von HELEN WANG. IEEE Computer Society. Mai 2011. URL: http://cseweb.ucsd.edu/~hovav/papers/mbys11.html (besucht am 23. Februar 2013).

MOZILLA DEVELOPER NETWORK: *window.name*. Mai 2009. URL: https://developer.mozilla.org/en-US/docs/DOM/window.name (besucht am 17. Februar 2013).

— *window.navigator.appCodeName*. April 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.navigator.appCodeName (besucht am 3. Dezember 2012).

— *window.navigator.cookieEnabled*. April 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.navigator.cookieEnabled (besucht am 3. Dezember 2012).
— *window.navigator.language*. August 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.navigator.language (besucht am 3. Dezember 2012).
— *window.screen.availHeight*. September 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.screen.availHeight (besucht am 3. Dezember 2012).
— *window.screen.availWidth*. September 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.screen.availWidth (besucht am 3. Dezember 2012).
— *window.screen.colorDepth*. Dezember 2007. URL: https://developer.mozilla.org/en-US/docs/DOM/window.screen.colorDepth (besucht am 3. Dezember 2012).
— *window.screen.height*. August 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.screen.height (besucht am 3. Dezember 2012).
— *window.screen.width*. August 2012. URL: https://developer.mozilla.org/en-US/docs/DOM/window.screen.width (besucht am 3. Dezember 2012).

Nagamalai, Dhinaharan, Dhinakaran, BeatriceCynthia, Sasikala, P., Lee, Seoung-Hyeon und Lee, Jae-Kwang: „Security Threats and Countermeasures in WLAN“. In: *Technologies for Advanced Heterogeneous Networks*. Hrsg. von Kenjiro Cho und Philippe Jacquet. Bd. 3837. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, S. 168–182. ISBN: 978-3-540-30884-3.

Neumann, Alexander: *Freier Silverlight-Klon Moonlight eingestellt*. Mai 2012. URL: http://www.heise.de/developer/meldung/Freier-

Silverlight-Klon-Moonlight-eingestellt-1586174.html (besucht am 22. Januar 2013).

Patel, Lalit: *JavaScript/CSS Font Detector*. März 2007. URL: http://www.lalit.org/lab/javascript-css-font-detect/ (besucht am 10. Mai 2013).

ProDevTipps.com: *Detecting Flash player version with JavaScript*. November 2008. URL: http://www.prodevtips.com/2008/11/20/detecting-flash-player-version-with-javascript/ (besucht am 21. Januar 2013).

Ratzloff, Matthew: *Detecting plugins in Internet Explorer (and a few hints for all the others)*. Juni 2007. URL: http://www.matthewratzloff.com/blog/2007/06/26/detecting-plugins-in-internet-explorer-and-a-few-hints-for-all-the-others/ (besucht am 26. Januar 2013).

Rouse, Margaret: *Transient cookie (session cookie)*. September 2005. URL: http://searchsoa.techtarget.com/definition/transient-cookie (besucht am 18. April 2013).

Sarvell, Henrik: *Detecting Flash player version with JavaScript*. November 2008. URL: http://www.prodevtips.com/2008/11/20/detecting-flash-player-version-with-javascript/ (besucht am 9. Februar 2013).

Sawall, Achim: „FSFE findet 2.160-mal Werbung für Adobe Reader“. In: (November 2010). URL: http://www.golem.de/1011/79126.html (besucht am 26. Januar 2013).

Schoen, Seth: *New Cookie Technologies: Harder to See and Remove, Widely Used to Track You*. September 2009. URL: https://www.eff.org/deeplinks/2009/09/new-cookie-technologies-harder-see-and-remove-wide (besucht am 17. Februar 2013).

Schwenke, Matthias Christoph: „Datenschutzrechtliche Gestaltungsanforderungen an die Individualisierung“. German. In: *Individualisierung und Datenschutz*. DUV, 2006, S. 59–237. ISBN: 978-3-8350-0394-1.

Soltani, Ashkan, Canty, Shannon, Mayo, Quentin, Thomas, Lauren und Hoofnagle, Chris Jay: *Flash Cookies and Privacy*. August 2009. url: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1446862 (besucht am 17. Februar 2013).

StatOwl: *Microsoft Silverlight Version Support*. Dezember 2012. url: http://www.statowl.com/silverlight.php (besucht am 22. Januar 2013).

Tanenbaum, Andrew S.: *Computernetzwerke*. 4., überarb. A. Pearson Studium, 2003. isbn: 3827370469.

Uecker, Philip: „Host-Provider, Content-Provider, Access-Provider oder was?“ In: *DFN Infobrief 06/09 (Institut für Informations-, Telekommunikations- und Medienrecht der Westfälischen Wilhelms-Universität Münster)* 06/09 (2009), 5 f.

Van Patten, Justin: „Isolierter Speicher in Silverlight 2“. In: *MSDN Magazin* 03/2009 (März 2009). url: http://msdn.microsoft.com/de-de/magazine/dd458794.aspx.

Zivadinovic, Dusan: *Tunnelbau – Router und Smartphones im VPN-Zusammenspiel*. März 2010. url: http://www.heise.de/mobil/artikel/NAT-Probleme-954958.html (besucht am 29. Januar 2013).

Zlotos, Ragni: *Firefox 22 soll Tracking-Cookies unterdrücken*. Februar 2013. url: http://heise.de/-1809872 (besucht am 10. Mai 2013).

9 SQL-Abfragen (Queries)

Hier sind die wichtigsten für die Auswertung verwendeten MySQL-Abfragen aufgelistet. Damit diese einzeln verwendbar sind, wurde u. a. auf VIEWs verzichtet.

*1 SELECT a.id, b.id FROM bfp AS a JOIN bfp AS b WHERE a.fonts_hash = b.fonts_hash AND NOT (a.fonts = b.fonts);

*2 SELECT a.id, b.id FROM bfp AS a JOIN bfp AS b WHERE a.plugins_hash = b.plugins_hash AND NOT (a.plugins = b.plugins);

*3 SELECT a.id, b.id FROM bfp AS a JOIN bfp AS b WHERE a.mimetypes_hash = b.mimetypes_hash AND NOT (a.mimetypes = b.mimetypes);

*4 SELECT COUNT(*) FROM bfp;

*5 SELECT COUNT(*) FROM bfp GROUP BY uid;

*6 SELECT AVG(t.c) FROM (SELECT COUNT(*) AS c FROM bfp GROUP BY uid) AS t;

*7 SELECT browser, COUNT(*) FROM user_agents GROUP BY browser;

*8 SELECT platform, COUNT(*) FROM user_agents GROUP BY platform;

*9 SELECT ismobiledevice, COUNT(*) FROM user_agents GROUP BY ismobiledevice;

*10 SELECT navigator_userAgent, http_user_agent FROM bfp WHERE NOT navigator_userAgent IS NULL AND NOT navigator_userAgent = http_user_agent;

*11 SELECT COUNT(*) FROM bfp WHERE NOT (ignorelevel & 2) AND NOT (ignorelevel & 4) GROUP BY http_user_agent, http_accept, http_accept_charset, http_accept_encoding, http_accept_language;

*12 SELECT COUNT(*) AS c FROM bfp WHERE NOT (ignorelevel & 2) AND NOT (ignorelevel & 4) GROUP BY http_user_agent, http_accept, http_accept_charset, http_accept_encoding, http_accept_language HAVING c > 10;

*13 SELECT COUNT(*) AS c FROM bfp WHERE NOT (ignorelevel & 2) AND NOT (ignorelevel & 4) GROUP BY http_user_agent, http_accept, http_accept_charset, http_accept_encoding, http_accept_language HAVING c = 1;

*14 SELECT COUNT(*) AS c FROM bfp WHERE NOT (ignorelevel & 2) AND NOT (ignorelevel & 4) GROUP BY http_user_agent HAVING c = 1;

15 SELECT b. FROM bfp AS b LEFT JOIN user_agents AS u ON u.id = b.id WHERE b.ignorelevel = 0 AND u.browser LIKE 'IE' AND b.plugins IS NULL AND b.mimetypes IS NULL;

*16 SELECT COUNT(*) AS c FROM bfp AS b LEFT JOIN user_agents AS u ON u.id = b.id WHERE b.ignorelevel = 0 AND u.browser LIKE 'IE' AND b.plugins IS NULL AND b.mimetypes IS NULL GROUP BY b.navigator_userAgent HAVING c = 1;

*17 SELECT * FROM uid_changes WHERE http_user_agent = 0 AND http_accept = 0 AND http_accept_charset = 0 AND http_accept_encoding = 0 AND http_accept_language = 0 AND navigator_appCodeName = 0 AND navigator_appName = 0 AND navigator_appVersion = 0 AND navigator_cookieEnabled = 0 AND navigator_language = 0 AND navigator_platform = 0 AND navigator_userAgent = 0 AND screen_width = 0 AND screen_height = 0 AND color_depth = 0 AND devicePixelRatio = 0 AND timezone_offset = 0 AND java_enabled = 0 AND color_activeborder = 0 AND color_activecaption = 0 AND color_appworkspace = 0 AND color_background = 0 AND color_buttonface = 0 AND color_buttonhighlight = 0 AND color_buttonshadow = 0 AND color_buttontext = 0 AND color_captiontext = 0 AND color_graytext = 0 AND color_highlight = 0 AND color_highlighttext = 0 AND color_inactiveborder = 0 AND color_inactivecaption = 0 AND co-

lor_inactivecaptiontext = 0 AND color_infobackground = 0 AND color_infotext = 0 AND color_menu = 0 AND color_menutext = 0 AND color_scrollbar = 0 AND color_threeddarkshadow = 0 AND color_threedface = 0 AND color_threedhighlight = 0 AND color_threedlightshadow = 0 AND color_threedshadow = 0 AND color_window = 0 AND color_windowframe = 0 AND color_windowtext = 0 AND plugin_flash = 0 AND plugin_adobe_acrobat = 0 AND plugin_silverlight = 0 AND plugins_notnull = 0 AND mimetypes_notnull = 0 AND fonts_notnull = 0;

*18 SELECT COUNT(*) FROM comparison WHERE uid = 1;

*19 SELECT COUNT(*) FROM comparison WHERE fp >= 0.7 AND uid = 0;

*20 SELECT id, uid, png_id FROM bfp WHERE png_id < id;

*21 SELECT a.id, a.uid, a.png_id, b.id, b.png_id FROM bfp AS a LEFT JOIN bfp AS b ON (a.png_id = b.id) WHERE a.png_id < a.id AND a.uid = b.uid;

*22 SELECT uid FROM bfp WHERE ignorelevel = 0 GROUP BY uid HAVING COUNT(uid) > 1;

*23 SELECT http_user_agent, COUNT(*) AS c FROM bfp GROUP BY http_user_agent ORDER BY c DESC LIMIT 1;