UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Täuschungsstrategien gegen Fingerprinting im Webbrowser

Camouflage strategies to protect against Browser Fingerprinting

Bachelorarbeit

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
Malte Luttermann

ausgegeben und betreut von
Prof. Dr. Maciej Liśkiewicz

Lübeck, den 12. September 2019

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

(Malte Luttermann)
Lübeck, den 12. September 2019

Kurzfassung Internetseiten haben oft ein großes Interesse daran, Daten von ihren Besuchern zu sammeln, um ihren Gewinn zu steigern – beispielsweise durch personalisierte Werbeanzeigen. Um an die Daten der Besucher zu gelangen, setzen Internetseiten verschiedenste Web Tracking Techniken ein – unter anderen das Browser Fingerprinting. Bei dem Browser Fingerprinting bekommt der Besucher selbst in der Regel nicht mit, dass Daten über ihn gesammelt werden. Diese Arbeit stellt einige Täuschungsstrategien vor, wie Nutzer vor Browser Fingerprinting geschützt werden können. Zudem wurden diese Täuschungsstrategien in Form einer Google Chrome Browser Erweiterung implementiert und evaluiert, um ihre praktische Wirksamkeit zu testen.

Abstract Websites show great interest in collecting personal data from their visitors in order to increase their profits, for example by using personalized advertisement. In order to receive personal data from visitors, websites apply many different web tracking techniques – among others the so called browser fingerprinting. With the use of browser fingerprinting, the visitor does generally not recognize that his personal data is being collected. This thesis presents some camouflage strategies to protect users from browser fingerprinting. Additionally, a Google Chrome browser extension was developed to implement and evaluate these camouflage strategies in the real world.

Inhaltsverzeichnis

1. Einleitung **1**

2. Grundlagen **3**
2.1. Web Tracking . . . 4
2.1.1. First-party Web Tracking . . . 4
2.1.2. Third-party Web Tracking . . . 5
2.2. Fingerprinting . . . 5
2.2.1. Funktionsweise von Fingerprinting . . . 6
2.2.2. Fingerprinting Techniken . . . 6
2.2.3. Arten von Fingerprinting . . . 12
2.2.4. Genauigkeit von Fingerprinting . . . 13
2.2.5. Abwehrmaßnahmen gegen Fingerprinting . . . 16
2.2.6. Fazit . . . 18

3. Unsere Strategien **19**
3.1. Bestimmung der zu besuchenden Seiten . . . 20
3.1.1. Analyse des Browserverlaufes . . . 20
3.1.2. Analyse von Third-party Requests . . . 21
3.1.3. Zeitpunkt der Verbindungen . . . 22
3.2. Strategien basierend auf Suchbegriffen . . . 23

4. Implementierung **27**
4.1. Aufbau einer Google Chrome Browser Erweiterung . . . 27
4.2. Bestimmung der zu besuchenden Seiten . . . 27
4.3. Vortäuschen von Verbindungen . . . 28
4.3.1. Häufigkeit von Verbindungen . . . 28
4.4. Bestimmung der Suchbegriffe . . . 29
4.4.1. Analyse von URLs, um Suchbegriffe zu erhalten . . . 30
4.5. Nach Suchbegriffen suchen . . . 32
4.5.1. Suchfelder auf Seiten finden . . . 32

5. Evaluation **35**
5.1. Aufbau der Experimente . . . 35
5.1.1. Konfiguration der Testumgebung . . . 35
5.1.2. Ablauf der Experimente . . . 36
5.2. Auswertung der Experimente . . . 37
5.2.1. Fazit . . . 43

6. Zusammenfassung und Ausblick **44**

1. Einleitung

Unzählige Internetseiten sammeln Daten von Nutzern. Oftmals geben Benutzer ihre Daten freiwillig an (zum Beispiel bei der Registrierung), aber es gibt auch viele Fälle, bei denen die Nutzer selbst gar nicht bemerken, dass ihre Daten gespeichert werden. Dementsprechend sind sich Nutzer auch nicht immer darüber bewusst, wozu die von ihnen gesammelten Daten verwendet werden. Die vorliegende Arbeit beschäftigt sich ausschließlich mit Fällen, in denen Nutzer ihre Daten *nicht* freiwillig übermitteln.

Es werden unterschiedlichste Daten der Nutzer gespeichert. Darunter fallen persönliche Informationen wie beispielsweise Vorname, Name, Adresse, Alter, Geschlecht, in letzter Zeit besuchte Internetseiten, zuletzt gekaufte Produkte und kürzlich kontaktierte Personen; geografische Daten wie der Wohnort sowie technische Daten wie zum Beispiel das verwendete Endgerät, der verwendete Internetbrowser und die IP Adresse [SES14]. Dies waren lediglich einige Beispiele; die Liste ließe sich auch noch fortsetzen.

Im Rahmen dieser Bachelorarbeit wurde die Browser Erweiterung *FPFool* für den Internetbrowser Google Chrome entwickelt [FPF19]. Diese Erweiterung soll die Privatsphäre ihrer Nutzer vor Fingerprinting im Webbrowser schützen, indem sie falsches Nutzerverhalten vortäuscht. Browser Fingerprinting ist eine Technik, die von Internetseiten eingesetzt werden kann, um Daten ihrer Besucher zu sammeln. Dabei bekommen die betroffenen Personen meistens nicht mit, dass Informationen über sie gespeichert werden. Aktuell verbreitete Abwehrmaßnahmen versuchen in der Regel dafür zu sorgen, dass Nutzer im Internet unerkannt bleiben. Das bedeutet, dass die Aktionen der Benutzer keiner Identität zugeordnet werden können sollen. Strategien, welche versuchen, die Anonymität zu wahren, sind aber leider nicht immer erfolgreich, weshalb wir in dieser Arbeit alternative Strategien vorstellen möchten. Die in dieser Arbeit vorgestellten Strategien haben nicht zum Ziel, die Aktionen des Nutzers geheim zu halten. Stattdessen sollen zusätzliche, gefälschte Aktionen durch die Software FPFool produziert werden, sodass Internetseiten nicht zwischen echten Aktionen des Nutzers und den vorgetäuschten Aktionen der Software unterscheiden können. Auf diese Art und Weise sollen die Interessen des Nutzers auch dann verschleiert werden, wenn der Versuch seine Identität geheim zu halten, fehlschlagen sollte.

Die Täuschungsstrategien wurden in Form einer Browser Erweiterung implementiert, weil eine solche Erweiterung direkten Zugriff auf wichtige Schnittstellen des Internetbrowsers besitzt. Beispielsweise haben wir einen direkten Zugriff auf den Browserverlauf und können beliebig Fenster sowie Tabs öffnen und schließen. Weiter wurde sich für den Internetbrowser Google Chrome entschieden, da dieser laut des Statistikportals Statista der meistbenutzte Internetbrowser ist [Sta19].

Um zu überprüfen, wie praxistauglich die vorgestellten Täuschungsstrategien tatsächlich sind, wurden einige Experimente durchgeführt, bei denen die entwickelte Browser Erweiterung FPFool zum Einsatz kam.

Neben dieser Einleitung ist diese Arbeit folgendermaßen aufgebaut: Kapitel 2 beschreibt alle für diese Arbeit wichtigen Grundlagen. Zunächst sehen wir uns an, wie Web Tracking allgemein funktioniert, welche Parteien beteiligt sind und welche Interessen von den jeweiligen Parteien vertreten werden. Dann wird ein Überblick darüber geliefert, wie Web Tracking basierend auf Fingerprinting Methoden umgesetzt wird. Dabei werden unterschiedlichste Fingerprinting Techniken vorgestellt. Wir werden lernen, welche Techniken es gibt, was sie auszeichnet und wie man versuchen kann, sich dagegen zu wehren, wenn man seine Informationen nicht preisgeben möchte.

In Kapitel 3 werden einige Ansätze für Algorithmen beschrieben, um Täuschungsstrategien gegen Fingerprinting in Internetbrowsern umzusetzen. Wir beschreiben, wie die zu besuchenden Seiten bestimmt werden sollen und stellen Täuschungsstrategien basierend auf Suchbegriffen vor.

Anschließend liefert Kapitel 4 einen Einblick in die Implementierung der Software FPFool. Zu Beginn wird der Aufbau einer Google Chrome Erweiterung erläutert. Danach wird erklärt, wie Verbindungen gezielt vorgetäuscht werden können und was dabei beachtet werden sollte. Außerdem wird beschrieben, wie URLs analysiert werden können, um gesuchte Begriffe zu ermitteln und wie automatisiert nach Begriffen auf Internetseiten gesucht werden kann.

Danach werden in Kapitel 5 einige Experimente vorgestellt, um die Effektivität der zuvor entwickelten Algorithmen zu testen. In diesem Kapitel werden wir überprüfen, inwiefern die zuvor beschriebenen Ideen praxistauglich sind. Dafür werden zunächst der Versuchsaufbau sowie die Testumgebung beschrieben. Abschließend sollen die Experimente ausgewertet werden.

Am Ende sollen die Erkenntnisse aus dieser Arbeit in Kapitel 6 zusammengefasst werden. Darüber hinaus geben wir einen Ausblick über mögliche Weiterentwicklungsmöglichkeiten der Software FPFool.

2. Grundlagen

In diesem Kapitel wird zunächst beschrieben, was unter dem Begriff Web Tracking zu verstehen ist und zwischen welchen Arten von Web Tracking unterschieden wird. Web Tracking kann nicht nur in Internetbrowsern, sondern zum Beispiel auch in E-Mail Programmen, mobilen Anwendungen oder anderen Anwendungsprogrammen vorkommen. Wir wollen uns in dieser Arbeit jedoch auf Tracking im Internetbrowser beschränken. Anschließend wird eine spezielle Web Tracking Technik – das Fingerprinting im Webbrowser – erklärt. Dabei gehen wir zuerst auf die allgemeine Funktionsweise ein und sehen uns danach unterschiedlichste Varianten von Fingerprinting im Webbrowser an. Abschließend wollen wir die Wirksamkeit von Fingerprinting im Webbrowser sowie Abwehrmaßnahmen dagegen betrachten.

Oftmals werden die Daten der Nutzer gesammelt, um personalisierte Werbeanzeigen zu schalten. Einige Nutzer befürworten personalisierte Werbung, andere hingegen möchten nicht, dass ihre Daten dafür verwendet werden. Das Sammeln der Nutzerdaten kann aber auch größere Nachteile für die betroffenen Personen mit sich bringen. Die Autoren von [MGEL12] und [HSL+14] haben unter anderen die folgenden Gefahren beschrieben:

Price discrimination beschreibt die Ausnutzung der Nutzerdaten zur Festlegung von Preisen im Internet. Zum Beispiel können Online-Shops ihre Preise abhängig von Charaktereigenschaften der Nutzer auswählen, um den größtmöglichen Gewinn herauszuholen. Die gespeicherten Nutzerdaten werden benutzt, um die finanzielle Situation des Nutzers einzuschätzen. Anhand dieser wird der Preis dann festgelegt. Personen, die wirtschaftlich besser dastehen, könnten einen höheren Preis angezeigt bekommen als Personen, denen es finanziell nicht so gut geht.

Search discrimination bezeichnet die Manipulation von Suchergebnissen anhand von Nutzerdaten. Beispielsweise können zwei Nutzer bei derselben Suchanfrage unterschiedliche Ergebnisse erhalten oder die Ergebnisse in einer anderen Reihenfolge angezeigt werden. Somit können auch hier beispielshalber teure Produkte für wohlhabende Nutzer zuerst angezeigt werden, um den Gewinn zu erhöhen.

Das Sammeln der Nutzerdaten kann auch noch schwerwiegendere Bedrohungen für die Privatsphäre der betroffenen Personen darstellen. Etwa besteht die Gefahr, dass die gesammelten Daten für Identitätsdiebstahl missbraucht werden [Woo12].

Wir haben nun anhand von einigen Beispielen gesehen, dass die Privatsphäre von Nutzern im Internet ernsthaften Gefahren ausgesetzt ist. Es existieren noch weitere Bedrohungen für Nutzer, die aus der Speicherung von personenbezogenen Daten resultieren. In [BCELBR17] und [SES14] berichten die Autoren ausführlich über weitere Gefahren.

2.1. Web Tracking

Das Sammeln und Speichern der persönlichen Informationen im Internet bezeichnen wir im Folgenden als *Web Tracking*. Die Internetseite, welche versucht, Informationen über den Besucher herauszufinden und abzuspeichern, nennen wir *Tracker*. Grundsätzlich unterscheiden wir zwischen zwei verschiedenen Arten von Web Tracking: *First-party Web Tracking* und *Third-party Web Tracking*, auf die wir im nächsten Abschnitt genauer eingehen wollen.

Es gibt eine Vielzahl an verschiedenen Web Tracking Techniken. Am bekanntesten sind wohl die, die auf *Cookies* basieren. Ein Cookie besteht aus einem Schlüssel sowie einem Wert für diesen Schlüssel. Internetseiten können Cookies auf dem Gerät des Benutzers abspeichern, wenn der Nutzer der Verwendung von Cookies zugestimmt hat. Diese Arbeit beschäftigt sich jedoch nur mit einer speziellen Web Tracking Variante, nämlich dem *Browser Fingerprinting*. Dementsprechend gehen wir ab sofort davon aus, dass ein Tracker Browser Fingerprinting einsetzt, wenn wir von Web Tracking sprechen. Ein Überblick über weitere häufig eingesetzte Web Tracking Techniken befindet sich in [BCELBR17].

2.1.1. First-party Web Tracking

First-party Web Tracking beschreibt eine Art von Web Tracking, bei der die vom Benutzer besuchte Internetseite selbst der Tracker ist. Das bedeutet, dass eine Internetseite die Daten ihrer Besucher sammelt und speichert. First-party Web Tracking läuft also so ab,

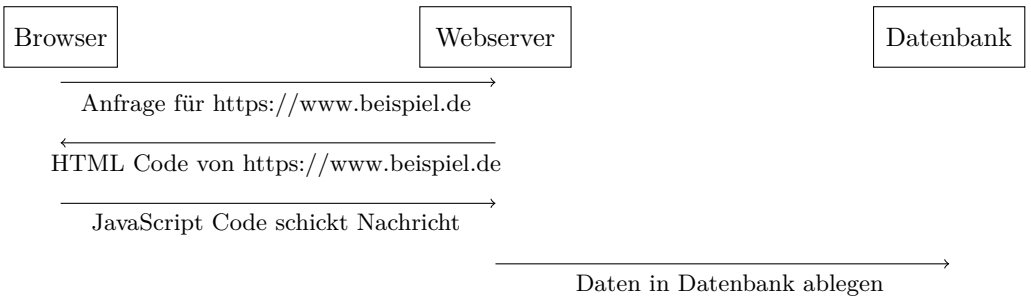


Abbildung 2.1.: Beispielszenario für First-party Web Tracking.

wie in Abbildung 2.1 dargestellt: Gibt ein Nutzer die Adresse https://www.beispiel.de in seinem Internetbrowser ein, so fragt der Browser den zu der Seite zugehörigen Webserver nach dem Inhalt der angefragten Seite. Als Antwort enthält der Browser des Nutzers dann den Inhalt der Seite in Form eines HTML Dokumentes. Der Browser lädt dieses Dokument nun, damit dem Nutzer die gewünschte Seite angezeigt wird. Falls sich außerdem JavaScript Code in dem HTML Dokument der angefragten Seite befindet, dann wird dieser in dem Browser des Nutzers ausgeführt, sofern der Nutzer der Ausführung von JavaScript zugestimmt hat. Dieser JavaScript Code kann nun beispielsweise die Browserkonfiguration des Nutzers abfragen und dann an den Webserver der besuchten Seite schicken, welcher diese Informationen dann in seiner Datenbank abspeichern kann.

2.1.2. Third-party Web Tracking

Third-party Web Tracking hingegen bezeichnet eine Art von Web Tracking, bei der nicht die vom Benutzer besuchte Internetseite seine Daten sammelt, sondern eine dritte Internetseite, welche auf der besuchten Seite in irgendeiner Form eingebunden ist. Ein Beispiel dafür ist der sogenannte »Like-Button« von Facebook, der auf anderen Internetseiten eingebunden werden kann. Wenn eine Internetseite den Like-Button einbindet, dann wird ein Script von Facebook geladen, welches in dem Browser des Nutzers ausgeführt wird. Über dieses Script kann Facebook nun Daten des Nutzers erhalten – auch wenn sich dieser eigentlich auf einer anderen Internetseite befindet. Diese Form des Web Tracking ist weit

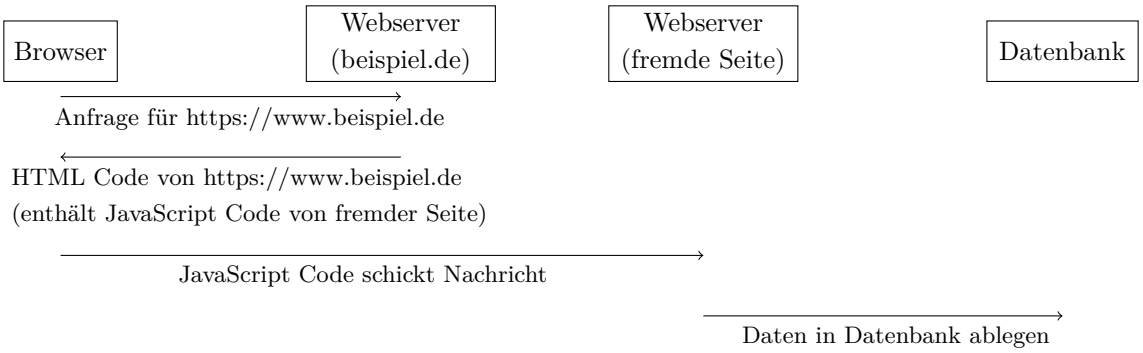


Abbildung 2.2.: Beispielszenario für Third-party Web Tracking.

verbreitet, wobei Internetnutzer oftmals nicht mitbekommen, dass ihre Daten an dritte Parteien gesendet werden.

2.2. Fingerprinting

Wie der Name *Fingerprinting* schon vermuten lässt, ist das Ziel bei diesem Verfahren einen digitalen Fingerprint bzw. Fingerabdruck von Internetseitenbesuchern zu erstellen, um jeden Besucher eindeutig identifizieren zu können. Damit dies gelingt, sammeln Tracker Informationen über die Systemkonfiguration der Nutzer. Dazu gehören beispielsweise die IP-Adresse, die Browserversion, die verwendeten Plugins und das zugehörige Betriebssystem. Für eine eindeutige Identifikation reichen diese Merkmale in der Regel noch nicht aus, weshalb der Tracker noch weitere Merkmale sammelt.

Ein *Fingerprint* bzw. *Fingerabdruck* ist dann einfach die Konkatenation aller gesammelten Merkmale. Ebenfalls möglich wäre es, aus dem Ergebnis dieser Konkatenation mit Hilfe einer Hashfunktion einen Hashwert zu erzeugen und diesen dann als Fingerabdruck abzuspeichern.

2.2.1. Funktionsweise von Fingerprinting

Besucht ein Nutzer eine Internetseite, welche Fingerprinting einsetzt, so erstellt diese Seite einen Fingerabdruck für den Nutzer. Der Fingerabdruck wird dann mit bereits existierenden Fingerabdrücken verglichen, um herauszufinden, ob dieser Nutzer die Seite schon einmal besucht hat. Der Nutzer bekommt davon nichts mit.

Nun kann die Seite die Aktionen des Nutzers auf dieser Seite protokollieren. Beispielsweise kann gespeichert werden, nach welchen Begriffen der Benutzer gesucht hat. Dies lässt bereits einige Rückschlüsse auf die Interessen des Nutzers zu. Im Gegensatz zu einigen anderen Web Tracking Techniken werden beim Fingerprinting keine Informationen auf dem System des Benutzers, sondern direkt auf dem Server des Trackers gespeichert. Folglich hat der Nutzer keine Möglichkeit, über ihn gesammelte Daten selbst zu löschen, wie es zum Beispiel bei Cookies der Fall ist.

Eine serverseitige Speicherung ermöglicht es außerdem, Nutzer über mehrere Besuche hinweg zu verfolgen, da der erstellte Fingerabdruck zwischen zwei Besuchen nicht verloren geht. Darüber hinaus wird *Cross-Domain Tracking* durch eine serverseitige Speicherung ermöglicht. Das heißt, dass der Benutzer übergreifend über mehrere Internetseiten hinweg identifiziert werden kann. Besucht ein Benutzer also Website A und danach Website B, so kann es vorkommen, dass Website B Wissen darüber besitzt, wie sich der Nutzer auf Website A verhalten hat. Voraussetzung dafür ist natürlich, dass diese beiden Seiten in irgendeiner Art und Weise miteinander in Verbindung stehen. Eine Verbindung kann beispielsweise dadurch bestehen, dass Website A Code von Website B einbindet oder aber beide Seiten ein externes Fingerprinting Script von demselben Anbieter nutzen.

Ändert sich die System- oder Browserkonfiguration des Nutzers zwischen seinen Besuchen auf einer Seite, dann findet der Tracker möglicherweise keinen passenden Fingerabdruck in seiner Datenbank, wenn der Nutzer die Seite erneut aufruft. In diesem Szenario ist es für den Tracker also schwierig zu erkennen, dass es sich bei dem Nutzer um einen wiederkehrenden und nicht um einen neuen Besucher handelt.

Es ist nicht unwahrscheinlich, dass ein Tracker neben Fingerprinting auch noch weitere Web Tracking Techniken einsetzt, um Nutzer mit einer höheren Erfolgswahrscheinlichkeit zu verfolgen.

2.2.2. Fingerprinting Techniken

Einige Informationen über die Systemkonfiguration des Benutzers werden automatisch beim Verbindungsaufbau an die besuchte Internetseite geschickt, weil das verwendete Kommunikationsprotokoll HTTP bzw. HTTPS diese Informationen immer mitschickt. Die besuchte Internetseite bekommt so Informationen über folgende Merkmale des Nutzers, ohne selbst etwas dafür tun zu müssen:

HTTP Accept Header Der HTTP Accept Header teilt einer Internetseite mit, welche Art von Inhalt vom Browser akzeptiert wird. Ein Beispiel für einen HTTP Accept

```
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
image/apng,*/*;q=0.8, application/signed-exchange;v=b3
```

Abbildung 2.3.: Ein Beispiel für einen HTTP Accept Header.

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/73.0.3683.86 Safari/537.36
```

Abbildung 2.4.: Ein Beispiel für einen User Agent.

Header kann in Abbildung 2.3 gefunden werden. Neben dem Accept Header gibt es auch noch weitere HTTP Header Attribute, die über das Protokoll übertragen werden. Dazu zählen beispielsweise die bevorzugte(n) Sprache(n) des Benutzers, der User Agent sowie die IP-Adresse des Nutzers.

User Agent Der User Agent beinhaltet Informationen über die System- und Browserkonfiguration des Benutzers. Abbildung 2.4 zeigt exemplarisch, wie ein User Agent aufgebaut ist.

Neben dem Namen der Komponenten wird auch eine detaillierte Versionsangabe der Komponenten übermittelt. Durch die sehr detaillierte Angabe der Versionen kommt es eher selten vor, dass mehrere Browser einen identischen User Agent aufweisen. Häufige Updates bekannter Internetbrowser tragen weiterhin dazu bei, dass es eine Vielfalt an Versionsnummern gibt, wodurch die Versionsnummer im User Agent für einen Fingerabdruck an Aussagekraft gewinnt.

IP-Adresse Die IP-Adresse wird dazu verwendet, um das Senden und Empfangen von Daten in einem Netzwerk zu regeln[1]. Dazu wird jedem Gerät eine IP-Adresse zugewiesen. Man kann sich die IP-Adresse eines Gerätes also wie die Postanschrift eines Hauses vorstellen. Damit dies funktioniert, darf es keine IP-Adresse doppelt geben.

Ursprünglich bestanden IP-Adressen aus 32 Bits, wie in Abbildung 2.5 beispielhaft dargestellt. Damit können $2^{32} \approx 4,3$ Milliarden verschiedene Adressen erzeugt werden. 32-Bit Adressen werden als IPv4-Adressen bezeichnet. Angesichts dessen, dass viele Menschen mehr als nur ein Gerät besitzen, welches mit dem Internet verbunden ist, werden die IPv4-Adressen so langsam knapp [Cen19]. Daher wurde eine neue Version eingeführt, bei der die Adressen nun aus 128 Bits bestehen. Diese Adressen werden als IPv6-Adressen betitelt und es ist möglich, $2^{128} \approx 3,4 \cdot 10^{38}$ verschiedene IPv6-Adressen darzustellen. Hier dürfte eine Knappheit an Adressen auch in ferner Zukunft ausgeschlossen sein.

Man könnte an dieser Stelle bereits vermuten, dass die IP-Adresse zur eindeutigen Identifikation eines Gerätes ausreichend ist. Dies ist aber nicht unbedingt immer gegeben. Dafür gibt es im wesentlichen zwei Gründe [Til13]:

[1] Dies gilt für alle Netzwerke, die das Internetprotokoll benutzen. Wir interessieren uns hier aber lediglich für ein spezielles Netzwerk, nämlich für das Internet.

11000000 . 10101000 . 00000000 . 00010111

Abbildung 2.5.: Ein Beispiel für eine IPv4-Adresse in Binärdarstellung. Die Adresse besteht aus 32 Bits.

1. *Dynamische IP-Adressen* erlauben es, dass Geräte ihre IP-Adresse ändern. Damit kann es vorkommen, dass ein Gerät morgen eine andere IP-Adresse als heute besitzt. Folglich kann sich eine Website nicht sicher sein, dass Besucher eindeutig über ihre IP-Adresse identifiziert werden können.

   Es ist dennoch von Vorteil für eine Website, wenn sie die IP-Adresse des Nutzers zur Erstellung eines Fingerabdrucks verwendet. Das liegt daran, dass sich die IP-Adresse im Normalfall nicht vollständig ändert, sondern nur teilweise, weil die IP-Adresse ein Indikator für den Standort des Geräts ist [PDA+16]. Da sich der Internetanschluss bei stationären Rechnern nicht ständig ändert, bleibt ein Teil der IP-Adresse also auch bei dynamischen IP-Adressen beständig. Es kann aber auch Ausnahmen geben, zum Beispiel bei mobilen Geräten. Dort ist eine vollständige Änderung der IP-Adresse nicht ausgeschlossen, weil das Internet an verschiedenen Standorten von unterschiedlichen Netzwerken aus verwendet werden kann.

2. *Network Address Translation (NAT)* erlaubt es, dass mehrere Geräte unter derselben IP-Adresse kommunizieren können [SE01]. Beispielsweise kann ein Privathaushalt mehrere Geräte über denselben Router mit dem Internet verbinden, dann kommunizieren alle Geräte über die IP-Adresse des Routers mit dem Internet. Der Router sorgt dann natürlich auch dafür, dass die empfangenen Daten an das korrekte Gerät geschickt werden.

Aufgrund der Tatsache, dass die oben genannten Merkmale bei jeder Kommunikation zwischen einem Benutzer und der besuchten Internetseite preisgegeben werden, ist es besonders schwierig für den Nutzer, diese Daten geheim zu halten. Zwar ist es möglich, falsche Werte vorzutäuschen, jedoch kann ein Tracker meistens durch aktive Techniken die echten Werte herausfinden und damit eine Inkonsistenz feststellen. Zum Beispiel kann der User Agent im Browser verstellt werden, aber ein Tracker kann mit Hilfe von JavaScript Code den echten User Agent erhalten. Paradoxerweise kann ein solcher Täuschungsversuch des Nutzers sogar dazu führen, dass ein Nutzer leichter zu identifizieren ist, weil die Inkonsistenz der Werte ein weiteres Unterscheidungsmerkmal darstellt [NKJ+13].

Wir wollen uns nun Techniken ansehen, die ein Tracker verwenden kann, um weitere Merkmale zur Erzeugung eines Fingerabdruckes zu sammeln. Jeder Tracker kann selbst entscheiden, welche Merkmale er in seinen Fingerabdruck einfließen lassen möchte und welche nicht. In der Regel versucht ein Tracker aber, möglichst viele Merkmale zu sammeln, um die Genauigkeit eines Fingerabdruckes zu erhöhen. Dazu wird in den allermeisten Fällen JavaScript Code eingesetzt. Der JavaScript Code des Trackers wird dann in dem Browser des Nutzers ausgeführt, wenn dieser der Ausführung von JavaScript zugestimmt hat.

In den folgenden Abschnitten werden wir uns nun die wichtigsten Fingerprinting Techniken ansehen.

JavaScript Standardobjekte

Es gibt bereits einige Standardobjekte, die in JavaScript vorhanden sind und welche Informationen über die Systemkonfiguration bereitstellen. Eine Abfrage mit Hilfe von JavaScript Standardobjekten ist zum einen sehr einfach einzubinden, zum anderen lässt sie sich seitens des Nutzers nicht verhindern, wenn er der Ausführung von JavaScript zugestimmt hat. Deshalb genügen oftmals wenige Zeilen JavaScript Code, um gewisse Konfigurationen des Nutzers abzufragen. Quelltext 2.1 zeigt, wie der User Agent des Nutzers mit einem sehr einfachen Aufruf abgefragt werden kann.

```
<!doctype html>

<script>
    console.log(window.navigator.userAgent);
</script>

</html>
```

Quelltext 2.1: Ermittlung des User Agents mit dem JavaScript Navigator Objekt.

Über das Navigator Objekt können auch noch weitere Merkmale auf analoge Weise ermittelt werden. Dazu gehören beispielsweise das verwendete Betriebssystem; die Einstellung, ob Cookies aktiviert sind; die bevorzugte Sprache des Benutzers sowie eine Liste der vorhandenen Plugins [Moz19]. Ein Tracker würde die gewonnenen Informationen dann nicht auf der Konsole ausgeben lassen, sondern diese an seinen Server schicken. Der Nutzer merkt dies nicht.

Neben dem Navigator Objekt gibt es auch noch weitere Standardobjekte in JavaScript, die einem Tracker behilflich sein können. Zum Beispiel können die Zeitzone des Nutzers über ein Datumsobjekt oder die Anzeigeeinstellungen – wie beispielsweise Höhe und Breite des verwendeten Bildschirms oder die Auflösung des Bildschirms – über das »Screen« Objekt ermittelt werden.

Installierte Schriftarten mit JavaScript ermitteln

Die installierten Schriftarten können nicht so einfach ausgegeben werden, wie die Merkmale, die durch JavaScript Standardobjekte ausgegeben werden konnten. Jedoch wurde in [NKJ+13] ein Trick vorgestellt, durch den ein Tracker herausfinden kann, ob eine bestimmte Schriftart auf dem System des Nutzers installiert ist. Dazu ist es zunächst wichtig, folgende zwei Dinge zu wissen:

1. Falls eine Internetseite eine Schriftart verwenden möchte, die auf dem System des Nutzers nicht installiert ist, muss der Browser trotzdem irgendeine Schriftart wählen. Dabei wählt er als Ausweichmöglichkeit eine Standardschriftart. Eine solche Standardschriftart ist auf jedem System vorinstalliert, weshalb sich der Browser sicher sein kann, dass er diese Schriftart verwenden kann.
2. Die Breite und Höhe eines Textes unterscheidet sich für unterschiedliche Schriftarten – auch wenn der Text identisch ist und dieselbe Schriftgröße hat. Dieser Unterschied ist zwar nur äußerst minimal und für das menschliche Auge nicht wahrzunehmen, aber er ist vorhanden und kann somit gemessen werden.

Mit Hilfe dieses Vorwissens kann ein Tracker nun folgendermaßen vorgehen, um zu prüfen, ob der Nutzer eine bestimmte Schriftart auf seinem System installiert hat: Zuerst wird ein Textelement erstellt, welches einen vordefinierten Text mit einer festen Schriftgröße enthält. Die Schriftart des Textes ist eine Standardschriftart. Nun misst der Tracker die Höhe und Breite des Textes und speichert diese Werte ab. Nach dieser Vorbereitung kann der Tracker eine Liste von Schriftarten durchgehen, für die er testen möchte, ob sie auf dem System des Nutzers installiert sind. Er geht dann für jede dieser Schriftarten so vor, wie eben beschrieben, das heißt, Text und Schriftgröße werden immer identisch gewählt. Im Anschluss vergleicht der Tracker die gemessenen Werte für Höhe und Breite mit den gespeicherten Werten der Standardschriftart.

Sollten die Werte einen Unterschied aufweisen, so kann der Tracker daraus schließen, dass die aktuell betrachtete Schriftart auf dem System des Nutzers installiert sein muss. Wären die Werte identisch, so muss also dieselbe Schriftart gewählt worden sein, das heißt, der Browser musste auf die Standardschriftart ausweichen. Dies bedeutet wiederum, dass die angefragte Schriftart nicht installiert ist.

Eine Implementierung dieses Vorgehens kann beispielsweise in dem öffentlich verfügbaren Quelltext der Seite *AmIUnique* [AmI19] gefunden werden. AmIUnique wurde von einer Gruppe von Wissenschaftlern ins Leben gerufen und zeigt unter anderem auf, wie Fingerprinting in der Praxis umgesetzt werden kann [LRB16].

Canvas Fingerprinting

Eine weitere Form des Fingerprinting ist das *Canvas Fingerprinting*, welches von Mowery und Shacham vorgestellt wurde [MS12]. Ein Canvas ist eine Zeichenoberfläche, auf der beliebige Formen gezeichnet werden können. Bei dieser Technik wird mittels JavaScript ein Text innerhalb eines solchen Canvas gezeichnet. Das Interessante dabei ist, dass unterschiedliche Systeme oftmals kleine Abweichungen bei gleichem Programmcode (gleicher Zeichnung) aufweisen. Eine solche Zeichnung ist aber konsistent, das heißt, sie ändert sich auch bei mehrfacher Ausführung des Codes auf demselben System nicht. Durch die kleinen Abweichungen einzelner Pixel bei unterschiedlichen Betriebssystemen und Internetbrowsern kann durch diese Technik also ein weiteres Unterscheidungsmerkmal gewonnen werden. Der Tracker speichert dazu einfach die Zeichenkette ab, welche das Bild (die Zeichnung) kodiert.

Zusätzlich können auch Inkonsistenzen aufdeckt werden: Wenn zum Beispiel der User Agent durch den Nutzer manipuliert wird, um seine wahre Systemkonfiguration zu verschleiern, so können über Canvas Fingerprinting das tatsächlich verwendete Betriebssystem sowie der eingesetzte Internetbrowser ermittelt werden und diese dann mit der im User Agent angegebenen Konfiguration verglichen werden [LRB16]. Das führt wiederum dazu, dass Nutzer, die ihre Systemkonfiguration verheimlichen wollen, teilweise einfacher identifiziert werden können, weil eben solche Inkonsistenzen vorliegen.

Canvas Fingerprinting kann innerhalb des Bruchteils einer Sekunde durchgeführt werden und der Benutzer bekommt davon in der Regel auch nichts mit.

WebGL

Ähnlich zu Canvas Fingerprinting ist das Fingerprinting mit Hilfe von WebGL. WebGL erlaubt es, dreidimensionale Objekte zu rendern. Durch dieses Vorgehen kann ein Tracker Informationen über die verwendete Hardware des Nutzers gewinnen, wie in einem Experiment der Autoren von [CLW17] gezeigt wurde. Diese Technik erlaubt es beispielsweise, die verwendete Grafikkarte des Systems des Nutzers ziemlich zuverlässig zu ermitteln.

Adobe Flash

Adobe Flash kann ebenfalls verwendet werden, um für die Erzeugung eines Fingerabdruckes relevante Informationen abzufragen. Dazu gehören zum Beispiel die installierten Schriftarten oder das verwendete Betriebssystem. Die installierten Schriftarten können über JavaScript zwar einzeln auf Vorhandensein geprüft werden, mit Adobe Flash kann jedoch eine vollständige Liste aller installierten Schriftarten bestimmt werden [Til13]. Dafür muss der Nutzer selbstverständlich Adobe Flash installiert haben und der Verwendung davon zugestimmt haben.

Adobe Flash verliert jedoch zunehmend an Bedeutung und steht somit auf immer mehr Systemen nicht zur Verfügung. Deshalb nutzen Tracker vorwiegend JavaScript, um Fingerabdrücke zu generieren.

Weitere Techniken

Neben den zuvor beschriebenen Techniken wurden in der Literatur auch noch weitere Techniken vorgestellt, die hier aber nicht genauer behandelt werden sollen. Gängige Open Source Fingerprinting Skripte benutzen vor allem die zuvor vorgestellten Techniken. Weitere Techniken finden eher weniger Einsatz [AmI19][Fin19].

In der Vergangenheit gab es auch noch Techniken, die heute nicht mehr funktionieren. Dies zeigt, dass die Browserhersteller auch aktiv werden, um die Privatsphäre ihrer Nutzer besser zu schützen. Es wurden bereits einige Fortschritte im Bereich des Schutzes gegen

Fingerprinting gemacht, aber auch Tracker entwickeln immer wieder neue Techniken, um die Genauigkeit von Fingerprinting zu erhöhen.

2.2.3. Arten von Fingerprinting

Cross-Browser Fingerprinting

Cross-Browser Fingerprinting beschäftigt sich damit, den Benutzer auch dann identifizieren zu können, wenn er eine Internetseite über unterschiedliche Internetbrowser besucht. Sobald ein Nutzer den Browser wechselt, werden einige Merkmale seiner Konfiguration, die durch den Tracker gesammelt werden, einen anderen Wert annehmen. Dadurch fällt es einem Tracker deutlich schwerer, den Benutzer wiederzuerkennen.

Es gibt bereits einige Verfahren, die versuchen, diese Herausforderung zu meistern. Bereits 2011 wurde in [BMFGI11] eine Idee vorgestellt, wie Cross-Browser Fingerprinting umgesetzt werden könnte. Dort werden die durch den Tracker erhobenen Merkmale auf die Merkmale, die sich nicht von Browser zu Browser unterscheiden, beschränkt. Allgemein wird dabei also lediglich die Systemkonfiguration des Nutzers betrachtet, nicht aber die Browserkonfiguration.

Einige Jahre später haben die Autoren von [CLW17] ein Verfahren beschrieben, womit Cross-Browser Fingerprinting sehr zuverlässig möglich sein soll. Laut Angabe der Autoren konnten Nutzer über mehrere Browser hinweg mit einer Erfolgswahrscheinlichkeit von 83,24% korrekt zugeordnet werden. Der Vorschlag von 2011 soll Nutzer lediglich zu 68,98% korrekt eingeordnet haben.

Es deutet sich bereits an, dass Cross-Browser Fingerprinting durchaus umsetzbar ist. Die Zuverlässigkeit ist dennoch geringer als bei Single-Browser Fingerprinting (Nutzer verwendet immer denselben Browser). Auf die Genauigkeit von Single-Browser Fingerprinting gehen wir gleich noch detailliert ein.

Cross-Device Fingerprinting

Cross-Device Fingerprinting versucht, Nutzer über verschiedene Endgeräte hinweg zuordnen zu können. Diese Form von Fingerprinting stellt den Tracker vor neue Herausforderungen. Wie lässt sich herausfinden, ob ein Fingerabdruck möglicherweise zu einem anderen, bereits gespeicherten Fingerabdruck zugeordnet werden sollte? Diese Fragestellung ist nicht ganz einfach zu beantworten, da nun sowohl Browserkonfiguration als auch Systemkonfiguration nicht mehr aussagekräftig sind.

Da Fingerprinting aber auf genau diesen beiden Eigenschaften basiert, ist Cross-Device Tracking alleine durch Fingerprinting nicht zuverlässig einsetzbar. Um Cross-Device Tracking zu betreiben, werden weitere Tracking Techniken benötigt – beispielsweise ein Login. Mit Hilfe eines Logins lassen sich Benutzer auch über mehrere Geräte hinweg korrekt zuordnen – vorausgesetzt sie loggen sich auf allen Geräten ein.

Merkmal	Wert
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Accept Header	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp, image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Zeichenkodierung	gzip, deflate, br
Bevorzugte Sprachen	de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
Host	amiunique.org
Upgrade-Insecure-Requests Header	1
Referrer	https://amiunique.org/
IP-Adresse	31.17.50.35
Protokoll	https

Abbildung 2.6.: Beispiel für den ersten Teil eines vollständigen Fingerprints. Die Merkmale dieser Tabelle werden über das HTTP Protokoll automatisch mitgeschickt. Weitere Techniken sind hierfür nicht nötig. Dieser Fingerprint wurde mit Hilfe von der Internetseite [AmI19] generiert.

Außerdem besteht die Option, das Surfverhalten der Benutzer dazu verwenden, um eine Verbindung zwischen verschiedenen Geräten herzustellen. Ein Tracker könnte versuchen, Muster im Surfverhalten zu erkennen und diese Muster dann mit den Mustern anderer Geräte zu vergleichen. Fraglich dabei ist nur, wie genau eine solche Zuordnung ist. Die Genauigkeit dieser Vorgehensweise dürfte vermutlich nicht hoch genug sein, um Nutzer zuverlässig zuordnen zu können. Leider gibt es bisher aber keine zuverlässigen Quellen, um herauszufinden, ob eine solche Strategie von Trackern tatsächlich verfolgt wird.

Fingerprinting bei mobilen Geräten

Auch mobile Endgeräte können mit Hilfe von Fingerprinting identifiziert werden. Dazu gab es bereits Experimente, um dies zu überprüfen. Die Autoren von [LRB16] haben festgestellt, dass etwa 81% der untersuchten mobilen Geräte einen eindeutigen Fingerabdruck hinterlassen. Wir wollen jedoch mobile Geräte in dieser Arbeit außer Acht lassen und uns auf Fingerprinting von stationären Endgeräten beschränken.

2.2.4. Genauigkeit von Fingerprinting

Fortan betrachten wir wieder das Szenario, dass ein Nutzer immer denselben Browser und dasselbe Endgerät verwendet. Ein vollständiger Fingerabdruck könnte dann so aussehen, wie in Abbildung 2.6 und Abbildung 2.7 dargestellt. Es ist unwahrscheinlich, dass es mehrere Nutzer gibt, die exakt dieselben Werte für *alle* Merkmale aufweisen. Da es aber vorkommen kann, dass sich die Konfiguration eines Nutzers im Laufe der Zeit ändert, reicht es einem Tracker nicht, dass jeder Nutzer einen einzigartigen Fingerabdruck aufweist. Viel wichtiger ist es, einen Nutzer immer wiedererkennen zu können – auch wenn sich die Konfiguration des Nutzers geändert hat. Nur dann können Tracker herausfinden, welche Aktivitäten Besucher auf ihrer Seite durchführen.

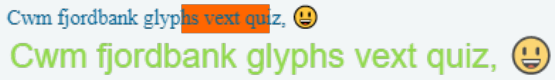
Merkmal	Wert
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.100 Safari/537.36
Betriebssystem	Win32
Cookies aktiviert?	Ja
Zeitzone	-120
Bevorzugte Sprachen	de-DE,de,en-US,en
Installierte Schriftarten	Arial, Arial Black, Arial Narrow, Bahnschrift, Brush Script MT, Calibri, Cambria, Cambria Math, Candara, Comic Sans MS und 56 weitere
Canvas	
Do Not Track Header	nicht angegeben
Local storage aktiv	Ja
Session storage aktiv	Ja
Bildschirmbreite	1920
Bildschirmhöhe	1080
Farbtiefe	24
Erster Pixel oben	153
Erster Pixel links	-1920
Verfügbare Breite	1920
Verfügbare Höhe	1040
Pixeldistanz links	undefiniert
Pixeldistanz oben	undefiniert
Plugins	Plugin 0: Chrome PDF Plugin; Portable Document Format; internal-pdf-viewer. Plugin 1: Chrome PDF Viewer; ; mhjfbmdgcfjbbpaeojofohoefgiehjai. Plugin 2: Native Client; ; internal-nacl-plugin.
BuildID	undefiniert
Audioformate	audio/aac, audio/flac, audio/mpeg, audio/mp4; codecs=“mp4a.40.2“, audio/ogg; codecs=“flac“, audio/ogg; codecs=“vorbis“, audio/ogg; codecs=“opus“, audio/wav; codecs=“1“, audio/webm; codecs=“vorbis“, audio/webm; codecs=“opus“
Videoformate	video/mp4; codecs=“flac“, video/ogg; codecs=“theora“, video/ogg; codecs=“opus“, video/webm; codecs=“vp9, opus“, video/webm; codecs=“vp8, vorbis“
WebGL Vendor	Google Inc.
WebGL Renderer	ANGLE (NVIDIA GeForce GTX 970 Direct3D11 vs_5_0 ps_5_0)
WebGL Daten	
WebGL Parameter	3380 verschiedene Parameter
Installierte Schriftarten (Flash)	Kein Flash vorhanden
Bildschirmauflösung (Flash)	Kein Flash vorhanden
Betriebssystem (Flash)	Kein Flash vorhanden
Adblock aktiviert?	Ja

Abbildung 2.7.: Beispiel für den zweiten Teil eines vollständigen Fingerprints. Die Merkmale dieser Tabelle werden vom Tracker durch weitere Techniken ermittelt. Dieser Fingerprint wurde mit Hilfe von [AmI19] generiert.

Eine eindeutige Identifikation ist nicht immer möglich, aber die Erfolgswahrscheinlichkeit für eine korrekte Zuordnung kann sehr hoch sein. In einer Studie von Peter Eckersley aus dem Jahr 2010 stellte sich heraus, dass 83,6% der rund 470.000 teilnehmenden Webbrowser eindeutig identifiziert werden konnten [Eck10]. Unter allen Browsern, bei denen Adobe Flash oder JavaScript aktiviert waren, lag die Erfolgswahrscheinlichkeit sogar bei 94,2%. Hier ist bereits zu sehen, dass JavaScript für einen Tracker sehr wichtig ist. Selbst wenn sich einige der beobachteten Merkmale verändert haben, konnte bei diesem Experiment mit Hilfe eines relativ einfachen Algorithmus oft ermittelt werden, zu welchem ursprünglichen Fingerabdruck die veränderte Konfiguration gehört. Die Studie hat damals lediglich acht verschiedene Merkmale zur Erstellung eines Fingerprints genutzt und dennoch sehr gute Resultate erzielen können.

Die Autoren von [YXY+12] haben 2012 gezeigt, dass bereits eine Kombination von User Agent und IP-Adresse zu einer Genauigkeit von ungefähr 80% führt, wenn es darum geht, einen Nutzer zu identifizieren. Der User Agent sowie die IP-Adresse werden automatisch über das HTTP Protokoll übertragen und sind somit schwer zu verstecken. Tracker können also auch komplett ohne JavaScript Fingerprints erstellen, die gar nicht mal so ungenau sind.

Eine Studie aus dem Jahr 2017 identifiziert Nutzer nach eigenen Angaben zu 99.24% korrekt [CLW17]. Interessanterweise gab es im darauffolgenden Jahr eine andere Studie, welche komplett andere Erfolgswahrscheinlichkeiten verzeichnete. Bei ungefähr 2 Millionen untersuchten Systemen waren lediglich 33.6% der Fingerabdrücke eindeutig [GBLB18]. Diese Zahl ist deutlich geringer als Werte in älteren Studien. Für stationäre Geräte liegt der Wert in dieser Studie bei 35.7%, während er für mobile Geräte mit 18.5% deutlich geringer ausfällt.

Die Werte unterscheiden sich doch relativ stark, was natürlich die Frage aufwirft, woran das liegen könnte. Hier sind einige Punkte, die möglicherweise eine Rolle spielen könnten:

1. Die älteren Studien haben alle deutlich weniger als 2 Millionen Systeme untersucht. Es ist durchaus denkbar, dass die Anzahl an unterschiedlichen Besuchern einer Seite einen relevanten Einfluss auf das Ergebnis haben kann.

2. Ein Großteil der Teilnehmer der Studie aus dem Jahr 2018 wiesen bereits einige Gemeinsamkeiten auf, beispielsweise haben alle Cookies akzeptiert und kamen aus derselben Region, was dazu führt, dass Lokalität und bevorzugte Sprachen immer gleich gewesen sein dürften.

3. Die Studie aus dem Jahr 2018 wurde auf einer real existierenden Internetseite durchgeführt. Die anderen Studien haben eine Seite verwendet, die zu reinen Forschungszwecken diente. Dadurch kamen die Teilnehmer beispielsweise aus verschiedensten Orten von der ganzen Welt.

Dementsprechend könnte Fingerprinting in sehr großem Ausmaß nicht so effektiv zu sein, wie zunächst angenommen wurde. Sicher sagen können wir das jedoch nicht. Schließlich kann das Ergebnis von Internetseite zu Internetseite stark variieren. Darüber hinaus sind einige Studien auch schon wieder einige Jahre her und die Rahmenbedingungen ändern

sich rasant. Es gibt immer wieder neue Abwehrmaßnahmen sowie neue Fingerprinting Techniken.

Eine eindeutige Identifikation mit Hilfe von Fingerprinting kann also durchaus zuverlässig funktionieren, auch wenn nicht klar zu sein scheint, wie zuverlässig Fingerprinting tatsächlich ist.

2.2.5. Abwehrmaßnahmen gegen Fingerprinting

Wir haben gesehen, dass das Spektrum an möglichen Fingerprinting Techniken sehr breit aufgestellt ist. Um sich erfolgreich gegen Browser Fingerprinting schützen zu können, muss man für jede einzelne dieser Techniken eine Abwehrmaßnahme besitzen. Das ist nicht immer so einfach möglich, wenn man beabsichtigt, die Nutzererfahrung nicht negativ zu beeinflussen.

Natürlich wäre es eine Möglichkeit, das Internet ausgehend von verschiedenen Geräten aus zu besuchen, um zu verhindern, dass man beobachtet wird. Jedoch ist dieses Szenario sehr unrealistisch. Auch mehrere Internetbrowser mit unterschiedlichsten Versionen würden dazu führen, dass Fingerprinting nicht mehr sehr effektiv eingesetzt werden kann. Aber auch diese Überlegung ist nicht wirklich realistisch. Der Nutzer soll keinen Mehraufwand haben, nur weil er seine Privatsphäre schützen möchte.

Die allermeisten existierenden Abwehrmaßnahmen versuchen, eine falsche Konfiguration vorzutäuschen. Meist ist die gefälschte Konfiguration eine Konfiguration, die häufig verwendet wird, damit der daraus resultierende Fingerabdruck möglichst nicht zur erfolgreichen Identifikation ausreicht, da es viele Systeme mit derselben Konfiguration und somit demselben Fingerabdruck gibt. Es gibt aber auch noch weitere Ansätze. Wir werden uns nun einige Abwehrmaßnahmen etwas genauer ansehen.

Deaktivieren von JavaScript, Adobe Flash und Java

Adobe Flash und Java sind zwar in der Überschrift genannt, jedoch verlieren diese zunehmend an Bedeutung – sowohl für Tracker als auch für alle anderen Internetseiten. Das Deaktivieren von JavaScript ist zwar eine Abwehrmaßnahme, die Browser Fingerprinting deutlich erschweren kann, aber dabei gibt es zwei wesentliche Probleme:

1. Für einen vollständigen Schutz gegen Fingerprinting reicht das Deaktivieren von JavaScript leider nicht aus. Für den Tracker ist es zwar deutlich schwieriger, einen Nutzer ohne JavaScript zu verfolgen, aber unmöglich ist dies dennoch nicht.
2. Die Nutzererfahrung leidet darunter, wenn JavaScript deaktiviert wird. Heutzutage nutzen nahezu alle Internetseiten JavaScript, sodass die Nutzung dieser Seiten in vollem Umfang auch nur dann möglich ist, wenn der Besucher die Ausführung von JavaScript erlaubt [W3T19].

Es gibt einige Browser Erweiterungen, die versuchen, einen Kompromiss zwischen einer positiven Nutzererfahrung und Schutz vor Fingerprinting zu finden. Zum Beispiel kann der Nutzer bei der Browser Erweiterung NoScript selbst entscheiden, welche Seiten JavaScript ausführen dürfen [NoS19]. Alle nicht explizit angegebenen Seiten dürfen keinen JavaScript Code in dem Browser des Nutzers ausführen.

Dennoch ist Vorsicht geboten, wenn Tools dieser Art verwendet werden. Paradoxerweise führen einige Tools, die eigentlich vor Fingerprinting schützen sollen, dazu, dass Fingerprinting einfacher umsetzbar wird [Eck10]. Das muss nicht immer der Fall sein, aber zum Beispiel haben die Autoren von [MBYS11] gezeigt, dass die Browser Erweiterung NoScript [NoS19] dafür sorgen kann, dass ein Nutzer leichter identifizierbar ist. Eigentlich ist NoScript eine Browser Erweiterung, die vor Fingerprinting schützen soll, indem der Nutzer explizit angeben muss, auf welchen Seiten JavaScript ausgeführt werden darf. Auf allen weiteren Seiten wird die Ausführung von JavaScript dann verboten. Es ist jedoch weiterhin für alle Seiten möglich, den JavaScript Code einer anderen Seite einzubinden. Lädt ein Tracker nun den JavaScript Code einer anderen Seite, so kann er prüfen, ob diese andere Seite zur Ausführung von JavaScript berechtigt ist oder nicht. Somit lassen sich Informationen darüber gewinnen, welche Websites von dem Nutzer bevorzugt werden. Diese Informationen können zusätzlich in den Fingerabdruck einfließen.

Browser Erweiterungen zum Schutz vor Fingerprinting

Es gibt mittlerweile eine Vielzahl an Browser Erweiterungen, die den Nutzer vor Fingerprinting schützen sollen. In der Vergangenheit wurde von Starov und Nikiforakis gezeigt, dass Nutzer teilweise anhand ihrer installierten Erweiterungen identifiziert werden können [SN17].

Browser Erweiterungen verändern oftmals den Inhalt einer Website, indem sie neue Elemente hinzufügen, Elemente entfernen oder Attribute verändern. Dabei unterscheiden sich die allermeisten Erweiterungen in irgendeiner Art und Weise voneinander, d.h. es gibt mindestens ein Merkmal, welches bei keiner anderen Erweiterung vorkommt.

Durch Überprüfung, ob bestimmte Elemente neu hinzugekommen sind, oder aber ob gewisse Elemente verschwunden sind, kann ein Tracker herausfinden, was sich genau geändert hat. Nun gleicht er diese Information einfach mit dem Verhalten häufig verwendeter Erweiterungen ab und kann so feststellen, welche Erweiterung installiert ist. Dieses Vorgehen funktioniert natürlich nicht immer und nicht für alle beliebigen Browser Erweiterungen. In [SN17] wurde aber gezeigt, dass eine beliebige Internetseite ungefähr 10% der Erweiterungen, die es im Google Chrome Store gibt, eindeutig erkennen kann.

Nutzer müssen also weiterhin gut aufpassen, welche Tools sie verwenden. Sicherlich gibt es auch einige Tools, die wirksam vor Fingerprinting schützen, jedoch sind eben nicht alle Tools so wirksam, wie sie versprechen. Dies ist eine weitere Hürde für viele Nutzer, da nicht klar ist, welche Tools aus einer großen Auswahl nun tatsächlich hilfreich sind.

Tor Netzwerk und Tor Browser

Das Tor Netzwerk ist ein beliebtes Netzwerk zur Wahrung der Anonymität von Nutzern. Der Tor Browser ist explizit darauf ausgerichtet, die Privatsphäre seiner Nutzer zu schützen. Der Tor Browser wurde bereits in einigen Arbeiten als gute Abwehrmaßnahme gegen Fingerprinting (und andere Web Tracking Techniken) dargestellt [May09]. Ob der Tor Browser dafür sorgen kann, Fingerprinting komplett zu unterbinden, ist aber nicht klar. Ein großer Nachteil des Tor Browsers ist vor allem, dass die Nutzererfahrung negativ beeinträchtigt wird. Der Tor Browser ist deutlich langsamer als zum Beispiel Google Chrome und nicht alle Internetseiten werden in vollem Umfang dargestellt, weil beispielsweise einige Inhalte blockiert werden.

2.2.6. Fazit

Fingerprinting ist eine Web Tracking Technik, die durch Nutzer nur schwierig zu verhindern ist, wenn die Nutzererfahrung nicht negativ beeinträchtigt werden soll. Auch wenn es einige Abwehrmaßnahmen gibt, ist es weiterhin eine große Herausforderung, einen vollständigen Schutz vor Fingerprinting zu erreichen. Existierende Abwehrstrategien versuchen häufig, die Systemkonfiguration vor Trackern zu verstecken oder sie so zu verändern, dass der Tracker sie nicht von anderen Systemen unterscheiden kann. Da diese Abwehrmaßnahmen aber möglicherweise nicht immer gewährleisten können, dass Nutzer im Internet unerkannt bleiben, wollen wir uns im nächsten Kapitel Ideen ansehen, wie wir Tracker täuschen können, auch wenn diese es geschafft haben, einen brauchbaren Fingerabdruck des Nutzers zu erstellen.

3. Unsere Strategien

Da es sich eventuell schwierig gestalten kann, seine Systemkonfiguration erfolgreich zu verstecken, wollen wir an dieser Stelle einen völlig anderen Ansatz betrachten, wie Nutzer ihre Privatsphäre im Internet schützen können. Wir versuchen dabei nicht, eine gefälschte Konfiguration vorzutäuschen oder die Konfiguration geheim zu halten. Tracker werden also nicht daran gehindert, einen Fingerabdruck von unserem System zu erstellen. Stattdessen werden wir versuchen, den gesammelten Fingerabdruck wertlos zu machen. Dabei ist es uns wichtig, dass die Nutzererfahrung nicht negativ beeinträchtigt wird. Benutzer sollen keinen zusätzlichen Aufwand tragen müssen, nur weil sie ihre Privatsphäre schützen möchten. Auch Benutzer, die keine Vorstellung davon haben, wie Web Tracking funktioniert, sollen sich effektiv schützen können, ohne sich erst fachliches Wissen aneignen zu müssen.

Grundsätzlich wollen wir Verbindungen zu Seiten vortäuschen, damit ein Tracker davon ausgeht, dass diese Verbindungen durch den Nutzer entstanden sind. Zunächst müssen wir also herausfinden, welche Seiten wir besuchen wollen. Dabei sollen potentielle Tracker mitbekommen, dass eine Verbindung aufgebaut wurde. Dazu überlegen wir zunächst, welche Seiten als potentielle Tracker in Frage kommen. Natürlich kommen alle Seiten, die direkt vom Nutzer besucht worden sind, als potentielle Tracker in Frage (First-party Web Tracking). Da es möglich ist, dass vom Nutzer besuchte Seiten JavaScript Code anderer Seiten einbinden, kommen alle anderen Seiten, deren Script eingebunden wird, ebenfalls als potentielle Tracker in Frage (Third-party Web Tracking). Ein potentieller Tracker muss nicht tatsächlich ein Tracker sein. Anhand eines JavaScript Quelltextes herauszufinden, ob dieser Code zu Tracking Zwecken eingesetzt werden soll, ist ein schwieriges Problem. Deshalb analysieren wir keine Quelltexte, sondern täuschen Verbindungen zu allen Seiten vor, die als potentielle Tracker in Frage kommen.

Das alleinige Vortäuschen von Verbindungen ist jedoch nach wie vor eine sehr schwache Abwehrmaßnahme. Die Besuchsstatistik wird dadurch etwas verfälscht, jedoch werden die Interessen des Nutzers weiterhin unverfälscht preisgegeben. Wir wollen also zusätzlich ein falsches Verhalten auf den besuchten Seiten vorgeben. Dafür versuchen wir, nach Begriffen zu suchen, sofern die besuchten Seiten eine Suche erlauben. Ist kein Suchfeld vorhanden, so führen wir keine weitere Aktion auf der Seite aus. Die nachfolgenden Abschnitte beschreiben diese Idee noch etwas genauer. Im Anschluss daran werden dann die Implementierungsdetails in Kapitel 4 geklärt.

3.1. Bestimmung der zu besuchenden Seiten

3.1.1. Analyse des Browserverlaufes

Zu Beginn wollen wir die zu besuchenden Seiten bestimmen. Dafür nutzen wir den Browserverlauf des Benutzers. Anhand des Verlaufes können wir nun eine Statistik direkt besuchter Seiten erstellen. Mit Hilfe dieser Statistik können wir nun gezielt Verbindungen aufbauen, sodass am Ende alle Seiten exakt gleich oft besucht worden sind. Damit kann ein Tracker nicht mehr erkennen, welche Seiten der Nutzer besonders oft besucht, das heißt, die Seitenaufrufe alleine geben nun keinen Aufschluss über die Interessen des Nutzers. Abbildung 3.1 stellt diese Idee grafisch dar. Die Antwort auf die Frage, welcher Zeitraum für die Ge-

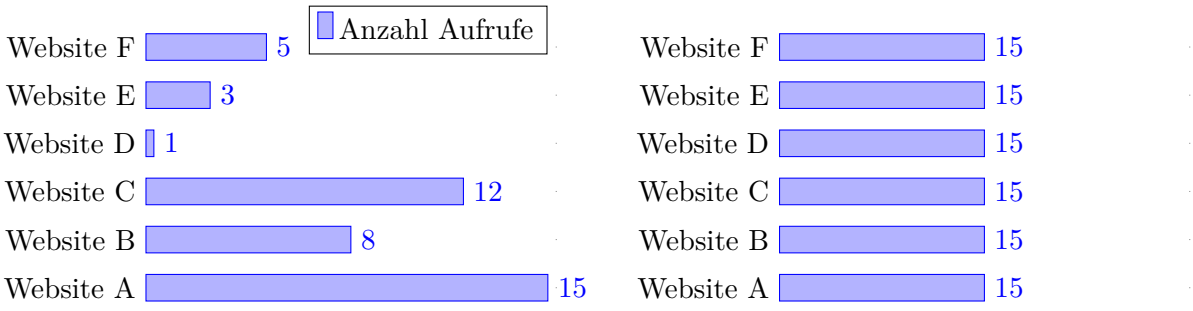


Abbildung 3.1.: Ein Beispiel für die Änderung der Statistik besuchter Seiten. Auf der linken Seite befindet sich eine fiktive Ausgangssituation, rechts die geänderte Statistik nach dem gezielten Aufbau falscher Verbindungen.

nerierung der Ausgangsstatistik sinnvoll ist, hängt vom Nutzerverhalten ab. Für Nutzer, die nur sehr wenige Seiten an einem durchschnittlichen Tag aufrufen, bietet es sich an, den Verlauf der letzten 7 Tage in die Statistik einfließen zu lassen. Bei Nutzern, die bereits an einem durchschnittlichen Tag sehr viele Seiten besuchen, reicht es vielleicht sogar aus, nur die Aufrufe des vorherigen Tages zu betrachten. Aus diesem Grund ist der Zeitraum ein Parameter in der entwickelten Software FPFool, der vom Nutzer selbst gewählt werden kann. Dabei kann der Nutzer einen Wert einschließlich zwischen dem letzten Tag und den letzten 14 Tagen auswählen.

Die Reihenfolge der Besuche ist zu Beginn durch die Anzahl der Besuche festgelegt. Das heißt, dass die Seiten mit den wenigsten Aufrufen als erstes besucht werden. Dazu werden beim Start alle Seiten aus dem Verlauf in den letzten x Tagen nach Aufrufen sortiert und dann in eine Warteschlange eingefügt. Der Wert x ist der vom Nutzer festgelegte Zeitraum. Sobald eine Seite besucht wurde, wird sie erneut an das Ende der Warteschlange der noch zu besuchenden Seiten eingefügt, falls sie immer noch nicht so oft besucht wurde wie die Seite mit den meisten Aufrufen. Das soeben beschriebene Verfahren wird in Quelltext 3.1 in Form von Pseudocode dargestellt.

```
// interval wird durch den Benutzer gesetzt;
// getBrowserHistory liefert den Verlauf in dem durch interval
```

```
// definierten Zeitraum zurueck
var browserHistory = getBrowserHistory(interval);
var queue = [];

for (var url of browserHistory) {
    // Zaehlt die Aufrufe der Seite in dem definierten Zeitraum
    var visits = countVisits(url, interval);
    // Fuegt die url nach Aufrufen sortiert in queue ein
    queue.insertSorted(url, visits);
}
```

Quelltext 3.1: Pseudocode des vorgestellten Algorithmus zur Analyse des Browserverlaufes.

3.1.2. Analyse von Third-party Requests

Wie zu Beginn bereits angedeutet, wollen wir auch Seiten besuchen, von denen JavaScript Code auf tatsächlich besuchten Seiten eingebunden wird. Diese können wir nicht einfach anhand des Browserverlaufes bestimmen. Stattdessen beobachten wir zur Laufzeit des Browsers alle ausgehenden Requests von allen aufgerufenen Seiten. Wir speichern dabei, welche JavaScript Dateien von welcher Seite angefordert wurden, wie in Abbildung 3.2 dargestellt. Diese Tabelle wird stetig aktualisiert, sobald ein neuer Request verschickt wird.

Besuchte Seite	Angefragte JavaScript Datei
Seite A	facebook.de/tracker.js
Seite A	google.de/script.js
Seite B	facebook.de/tracker.js
Seite C	google.de/script.js

Abbildung 3.2.: Eine beispielhafte Tabelle angefragter JavaScript Dateien. Eine Seite kann beliebig viele Dateien anfragen.

Gleichzeitig wollen wir bei jedem neuen Request prüfen, ob die angefragte JavaScript Datei bereits von einer anderen Seite angefragt wurde. Dazu betrachten wir einfach alle Einträge der Tabelle. Alle Seiten, die dieselbe Datei bereits in der Vergangenheit angefragt haben, werden nun ebenfalls in die Warteschlange der noch zu besuchenden Seiten eingefügt. Für die Tabelle in Abbildung 3.2 bedeutet dies konkret, dass wir Seite A und Seite B in die Warteschlange einfügen würden, wenn ein Request beobachtet wird, der die Datei facebook.de/tracker.js anfordert.

Abbildung 3.3 stellt die Tabelle aus Abbildung 3.2 grafisch dar. Das zuvor beschriebene Vorgehen führt nun also dazu, dass wir bei einem Request alle Seiten besuchen, die durch eine Kante mit dem angefragten Script verbunden sind. Sollte das geladene Script nun tatsächlich ein Tracking Script sein, so bekommt der Tracker die vorgetäuschte Verbindung wahrscheinlich mit. Quelltext 3.2 stellt diese Idee in Form von Pseudocode dar.

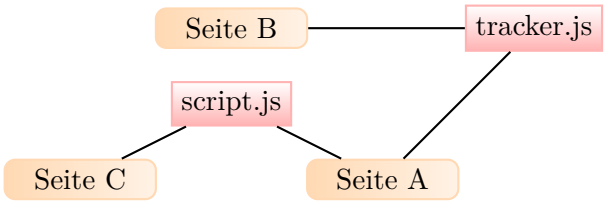


Abbildung 3.3.: Ein beispielhafter Graph von angefragten JavaScript Dateien.

```
function monitorThirdPartyRequest(req) {
    // req.origin ist die anfragende Seite;
    // req.url die angefragte Script Ressource und
    // req.type der Typ der angefragten Ressource
    if (req.type == 'script') {
        // Wenn dieselbe Ressource von einer anderen Seite
        // angefragt wurde, dann fuege die andere Seite in
        // queue ein
        for (var [key, val] of database.entries()) {
            if (val.includes(req.url) && key != req.origin)
                queue.push(key);
        }

        // Fuegt den Request in die Datenbank hinzu, falls
        // er noch nicht vorhanden war
        if (!database.get(req.origin).includes(req.url))
            database.store(req.origin, req.url);
    }
}
```

Quelltext 3.2: Pseudocode des vorgestellten Algorithmus zur Analyse von Third-party Requests.

3.1.3. Zeitpunkt der Verbindungen

Wir wissen nun, welche Seiten besucht werden sollen und wie oft diese Seiten besucht werden sollen. Jedoch ist noch nicht geklärt, zu welchem Zeitpunkt wir diese Seiten besuchen wollen. Statt nur die Anzahl der Besuche pro Seite in dem festgelegten Zeitraum zu zählen, könnten wir die Statistik noch etwas verfeinern. Beispielsweise können wir jeden Tag in Intervalle von jeweils drei Stunden unterteilen und die Anzahl der Besuche abhängig von diesen Intervallen zählen, wie in Abbildung 3.4 dargestellt ist. Wir können das Nutzerverhalten noch realistischer abbilden, wenn wir die falschen Verbindungen zu ähnlichen Zeiten wie zu den tatsächlichen Besuchszeiten des Nutzers aufbauen. Dazu könnte man jedem Zeitfenster eine Wahrscheinlichkeit zuweisen, mit der eine Seite innerhalb dieses

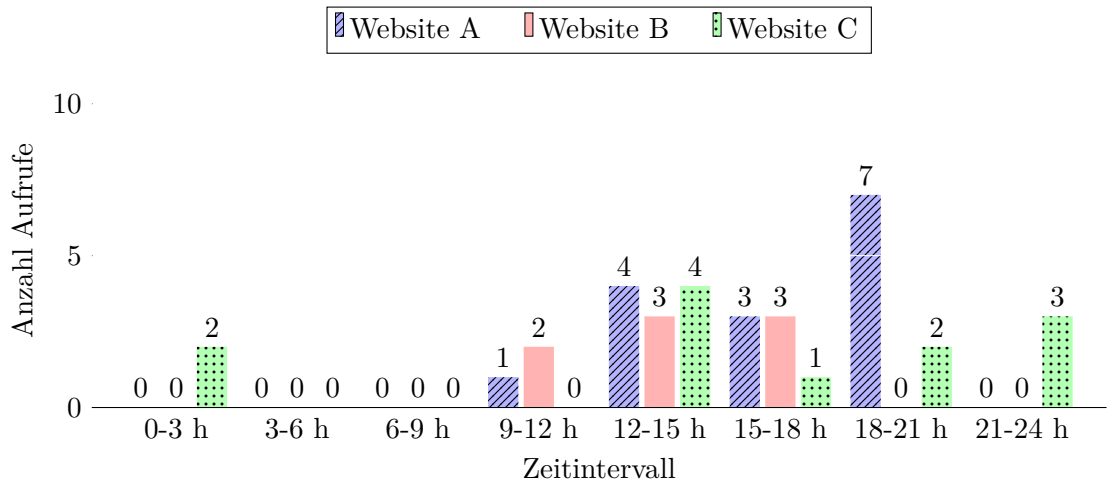


Abbildung 3.4.: Eine beispielhafte Statistik besuchter Seiten mit Zeitfenstern.

Zeitfensters besucht wird. Diese Idee wurde im Rahmen dieser Arbeit aber nicht praktisch implementiert. Die Anzahl der Besuche wird unabhängig von Zeitfenstern gezählt, wie zuvor in Abbildung 3.1 dargestellt. Dies liegt vor allem daran, dass das entwickelte Programm nur solange aktiv ist, wie der Nutzer seinen Internetbrowser geöffnet hat. Wir können Verbindungen also nur dann aufbauen, wenn der Nutzer seinen Browser geöffnet hat. Dadurch ist das Programm automatisch zu Zeiten aktiv, zu denen der Nutzer ebenfalls im Internet unterwegs ist, was ausreichend sein dürfte. Darüber hinaus können wir keine Verbindungen in der Zukunft planen, da wir nicht wissen, ob der Nutzer den Browser zu einem späteren Zeitpunkt weiterhin geöffnet haben wird oder nicht.

3.2. Strategien basierend auf Suchbegriffen

Zuletzt stellt sich die Frage, wie wir die Begriffe bestimmen, nach denen auf den besuchten Seiten gesucht werden soll. Da wir die Interessen des Nutzers verbergen möchten, wollen wir möglichst nicht dieselben Begriffe wählen wie die, nach denen der Nutzer bereits gesucht hat.

Wenn wir stattdessen nach Begriffen suchen würden, die überhaupt nichts mit den tatsächlich gesuchten Begriffen zu tun haben, würden wir vermutlich schon etwas näher an unser Ziel kommen. Diese Begriffe könnten möglicherweise sogar zufällig aus einem Wörterbuch gewählt werden. Dabei wären wir jedoch auf einzelne Suchwörter – also Eingaben, die aus genau einem Wort bestehen – beschränkt. Schließlich können wir nicht zuverlässig sinnvolle Suchanfragen, die aus mehreren Wörtern bestehen, bilden. Zusätzlich könnte es vorkommen, dass jeglicher Zusammenhang zwischen dem zufällig gewähltem Wort und dem ursprünglichen Suchbegriff fehlt. Im schlimmsten Fall kann das zufällig gewählte Wort sogar in einer anderen Sprache sein (zum Beispiel wenn der Benutzer neben deutschen Begriffen auch nach englischen Begriffen sucht und wir immer nur Wörter aus einem

deutschen Wörterbuch wählen). Da wir die Bedeutung einer Suchanfrage nicht durch ein Programm ermitteln können, wissen wir auch nicht, in welcher Sprache die Anfrage ist und können das Wörterbuch somit nicht abhängig von der Anfrage wählen. Das Wörterbuch müsste also statisch bleiben (immer dieselbe Sprache). Da dieser Gedankengang jedoch nicht sehr vielversprechend klingt, wollen wir eine andere Strategie wählen.

Zunächst finden wir heraus, ob der Nutzer bereits auf der entsprechenden Seite nach Suchwörtern gesucht hat. Falls nicht, dann führen wir ebenfalls keine Suche durch. Anderenfalls ermitteln wir den vom Nutzer verwendeten Suchbegriff. Wie genau das umgesetzt wird, folgt in Kürze.

Dann verwenden wir Google, um verwandte Begriffe zu dem ermittelten Suchbegriff herauszufinden: Wenn ein Begriff bei Google eingegeben wird, so gibt Google direkt einige Vorschläge an, wie die Suche vervollständigt werden kann. Diese Vorschläge wollen wir nutzen, um einen Suchbegriff zu ermitteln. Dafür zerlegen wir zunächst den ursprünglichen Suchbegriff des Nutzers in alle möglichen Kombinationen von Wörtern, jedoch ohne die Reihenfolge von einzelnen Wörtern zu verändern. Für den Suchbegriff »Hamburg Wetter« würden wir also die Begriffe »Hamburg«, »Wetter« und »Hamburg Wetter« erhalten. Nun geben wir jeden Begriff, der durch diese Zerlegung entstanden ist, gesondert bei Google ein und erhalten einige Vorschläge zur Vervollständigung der Suchanfrage. Von diesen Vorschlägen wählen wir nun jeweils einen Vorschlag für jeden der Begriffe aus. Dies geschieht zufällig, damit wir nicht immer denselben Vorschlag für identische Begriffe auswählen. In unserem Beispiel könnten wir dadurch für den Begriff »Hamburg« nun den Vorschlag »Hamburg Airport«, für den Begriff »Wetter« den Vorschlag »Wetter Kiel« und für den Begriff »Wetter Hamburg« den Vorschlag »Wetter Hamburg heute« erhalten.

Nun prüfen wir, ob wir einen Suchbegriff gefunden haben, der dieselbe Anzahl an Wörtern wie der ursprüngliche Suchbegriff des Nutzers enthält und kein einziges Wort aus dem ursprünglichen Suchbegriff des Nutzers enthalten ist. Wenn dies der Fall ist, dann nutzen wir diesen Begriff für unsere Suche. Ansonsten wiederholen wir das Verfahren solange, bis wir einen solchen Begriff gefunden haben. Zusätzlich können wir noch ein Limit an Durchläufen festlegen, um Endlosschleifen zu verhindern. Wenn wir das Verfahren wiederholen, dann betrachten wir Kombinationen von Wörtern, die wir bereits betrachtet haben, nicht erneut. In unserem Beispiel würde also im nächsten Durchlauf aus dem Begriff »Hamburg Airport« auch wieder das Wort »Hamburg« entstehen. Da dieses aber bereits vorkam, interessiert es uns jetzt nicht weiter. Ohne diese Einschränkung würden wir immer wieder dieselben Wörter betrachten, was wir natürlich nicht beabsichtigen. Darüber hinaus können wir auch Suchbegriffe außer Acht lassen, die aus mehr Wörtern als der ursprüngliche Suchbegriff des Nutzers bestehen, da die Anzahl identisch bleiben soll und wir durch die Vorschläge zur Vervollständigung nur weitere Wörter hinzufügen. Dieses Vorgehen basiert nach wie vor ein wenig auf dem Zufall, aber die Chancen, einen sinnvollen Begriff zu finden, sind deutlich höher als wenn wir den Suchbegriff vollkommen zufällig wählen würden. Abbildung 3.5 stellt den Ablauf dieses Verfahrens grafisch anhand des soeben vorgestellten Beispieles dar. Zudem befindet sich der Pseudocode dieses Algorithmus in Quelltext 3.3.

```
// original ist der urspruengliche Suchbegriff des Nutzers;
// queue enthaelt noch zu betrachtende Begriffe;
```

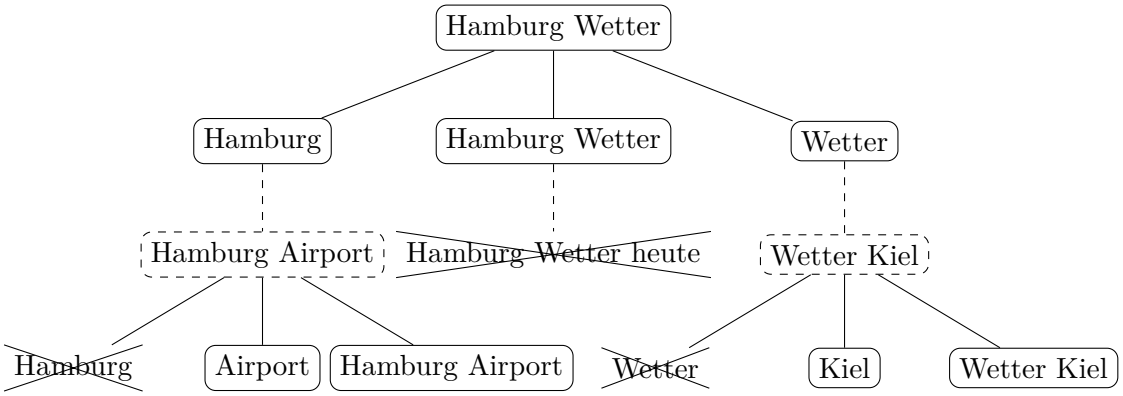


Abbildung 3.5.: Beispiel zur Generierung von neuen Suchbegriffen. Die gestrichelten Knoten stellen die Vorschläge zur Vervollständigung von Google dar. Aus Platzgründen konnte das Verfahren leider nicht bis zum Ende dargestellt werden. Für den Begriff »Airport« könnte im nächsten Schritt zum Beispiel der Vorschlag »Airport Frankfurt« gewählt werden. Dieser enthält genau zwei Wörter und enthält keines der Wörter aus der ursprünglichen Anfrage »Hamburg Wetter« und wäre somit ein möglicher Suchbegriff, nach dem unsere Software FPFool suchen kann.

```
// done ist ein Array bereits betrachteter Begriffe und
// runs die Anzahl der bisherigen Durchlaeufe
function getSearchterm(original, queue, done, runs) {
    if (runs < MAX_RUNS && !queue.isEmpty()) {
        var curr = queue.shift();
        // Liefert ein Array aller moeglichen Kombinationen mit
        // Anzahl Woerter nicht groesser als Anzahl Woerter des
        // urspruenglichen Begriffs
        var splitTerm = getAllCombinations(curr);
        // Alle bereits betrachteten Begriffe entfernen
        splitTerm = removeDone(splitTerm, done);
        done.concat(splitTerm);

        for (var term of splitTerm) {
            // Liefert Vorschlag von Google zurueck und ignoriert
            // dabei bereits betrachtete Begriffe
            var suggestion = requestSuggestion(term, done);
            if (isValid(original, suggestion))
                return suggestion;
            else
                queue.add(suggestion);
        }

        // Kein passender Vorschlag gefunden, also neuer Aufruf
        return getSearchterm(original, queue, done, ++runs);
```

```
    } else {
        return '';
    }
}
```

Quelltext 3.3: Pseudocode des vorgestellten Algorithmus zur Generierung von Suchbegriffen.

Als nächstes wollen wir mit den Implementierungsdetails dieser Strategien fortfahren. In Kapitel 4 wird beschrieben, wie die soeben vorgestellten Strategien praktisch implementiert wurden. Dort werden wir auch sehen, wie wir die vom Nutzer gesuchten Begriffe herausfinden können.

4. Implementierung

Dieses Kapitel beschäftigt sich mit der Implementierung der in Kapitel 3 vorgestellten Strategien. Dazu wurde eine Browser Erweiterung mit dem Namen FPFool für den Browser Google Chrome entwickelt. Da die entwickelte Software nur für einen Browser funktioniert, schützt sie den Nutzer nicht vor Cross-Browser Fingerprinting, falls dieser mehrere Browser verwenden sollte. Die zuvor vorgestellten Strategien selbst sind nicht an einen Browser gebunden, aber jeder Browser erfordert eine eigene Implementierung dieser Strategien.

Wir beginnen zunächst mit dem Aufbau einer Google Chrome Erweiterung und fahren danach mit der Bestimmung der zu besuchenden Seiten fort. Anschließend sehen wir uns an, wie wir Suchbegriffe bestimmen können, Suchfelder auf Internetseiten erkennen können und automatisiert nach Begriffen suchen können.

4.1. Aufbau einer Google Chrome Browser Erweiterung

Eine Google Chrome Browser Erweiterung besteht im Wesentlichen aus zwei wichtigen Komponenten:

Content Scripte werden immer dann ausgeführt, wenn innerhalb des Browsers eine Internetseite aufgerufen wird. Diese Scripte haben direkten Zugriff auf die Seite und können den Inhalt der Seite beliebig manipulieren.

Background Scripte sind dauerhaft im Hintergrund aktiv und haben keinen Zugriff auf im Browser geöffnete Seiten. Dafür stehen uns hier einige Schnittstellen des Browsers zur Verfügung, die über Content Scripte nicht erreichbar sind. Beispielsweise können wir über eine dieser Schnittstellen Tabs erstellen und schließen.

Die beiden Scripte können miteinander kommunizieren, indem sie sich gegenseitig Nachrichten schicken. Ist die Erweiterung im Browser aktiviert, dann wird das Background Script immer sofort beim Start des Browsers ausgeführt und läuft so lange, wie der Browser geöffnet bleibt oder bis alle Instruktionen abgearbeitet wurden. Zudem wird das Content Script immer automatisch ausgeführt, wenn eine beliebige Internetseite innerhalb des Browsers geladen wird.

4.2. Bestimmung der zu besuchenden Seiten

Die Bestimmung der zu besuchenden Seiten erfolgt dank bereits vorhandener Schnittstellen auf sehr einfache Art und Weise. Das Background Script hat direkten Zugriff auf eine

Schnittstelle des Browsers, die es erlaubt, sämtliche Informationen über den Browserverlauf zu erhalten. Somit können wir leicht herausfinden, welche Seiten in dem eingestellten Zeitraum besucht worden sind und wie häufig sie besucht worden sind. Außerdem hat das Background Script ebenfalls die Möglichkeit, alle ausgehenden Requests zu beobachten, indem die dafür vorgesehene Schnittstelle verwendet wird. Wenn wir einen Request finden, der eine JavaScript Datei anfragt, dann speichern wir diesen Request einfach ab. Somit können wir auch Third-party Requests jederzeit beobachten.

4.3. Vortäuschen von Verbindungen

Aufgrund dessen, dass wir Verbindungen zu Internetseiten vortäuschen wollen, müssen wir diese auf irgendeine Art und Weise aufrufen. Eine erste Überlegung wäre, einen einfachen HTTP Request zu versenden. Diese Vorgehensweise erzielt aber nicht den gewünschten Effekt. Ein einfacher HTTP Request führt nämlich nicht dazu, dass von der angefragten Internetseite eingebetteter Code ausgeführt wird. Bei einem HTTP Request erhalten wir lediglich ein HTML Dokument als Antwort, welches nicht im Browser geladen wird. Weil wir Tracker täuschen möchten, ist es für uns sehr wichtig, dass eingebetteter Code ausgeführt wird. Um dies sicherzustellen, öffnen wir stattdessen einen neuen Tab, der die Internetseite tatsächlich aufruft. Damit der Nutzer von diesem Tab möglichst wenig gestört wird, wird der Tab in einem neuen Browserfenster geöffnet, welches minimiert ist. Das Öffnen sowie Schließen von Tabs und Fenstern wird uns durch eine bereits vorhandene Schnittstelle sehr einfach gemacht.

4.3.1. Häufigkeit von Verbindungen

Die von uns besuchten Seiten sollen selbstverständlich nicht merken, dass die Aufrufe nicht durch einen Menschen erfolgt sind. Deshalb ist es wichtig, dass wir nicht zu viele Verbindungen in zu kurzer Zeit aufbauen. Viele Seiten verwenden auch einen Botschutz, wenn zu viele Anfragen auf einmal ankommen. Dies kann beispielsweise ein Captcha sein, welches zunächst gelöst werden muss, bevor man auf die gewünschte Seite gelangt. Ein Beispiel für den Botschutz der Seite Google kann in Abbildung 4.1 gefunden werden. Wir wollen in jedem Fall verhindern, dass ein Botschutz auftaucht, da unsere Software FPFool einen solchen nicht umgehen kann. Deshalb werden die Verbindungen nicht alle innerhalb weniger Sekunden aufgebaut. Dies würde einen Botschutz sofort auslösen, wenn ein solcher vorhanden ist.

Die Verweildauer auf einer Seite sollte bestenfalls so gewählt werden, dass sie menschlichem Verhalten ähnlich ist. Besuchte Seiten sollen keineswegs vermuten, dass es sich bei dem Besuch um keinen echten Menschen handelt. Die Autoren von [LWD10] haben gezeigt, dass es immer unwahrscheinlicher wird, dass ein Besucher eine Internetseite verlässt, je länger er diese bereits geöffnet hat. Andersherum ist es zu Beginn am wahrscheinlichsten, dass ein Besucher eine Internetseite direkt wieder verlässt. Genauer ausgedrückt bedeutet dies, dass die meisten Besucher eine Internetseite innerhalb der ersten 10-20 Sekunden

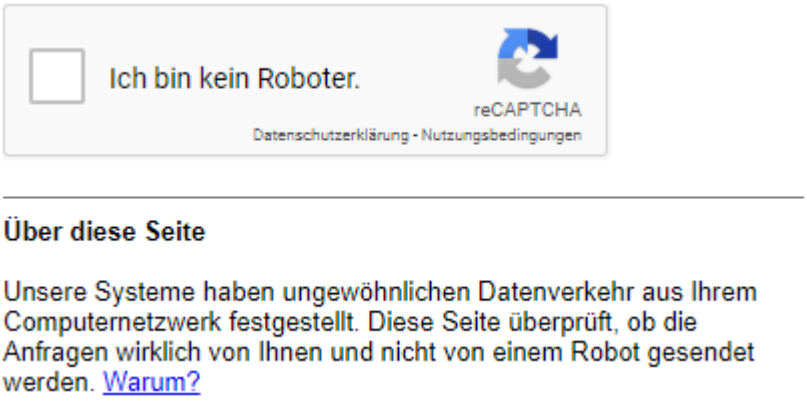


Abbildung 4.1.: Ein Beispiel für einen Botschutz der Seite Google.

direkt wieder verlassen [Nie11]. Diese Information machen wir uns zunutze, indem wir eine besuchte Seite innerhalb von 10-20 Sekunden wieder verlassen. Der exakte Wert wird zufällig bestimmt.

Aufgrund dessen, dass wir ein Limit an gleichzeitig geöffneten Tabs festgelegt haben, macht es auch keinen Sinn, schneller neue Verbindungen aufzubauen als Tabs geschlossen werden. Ein solches Limit an Tabs ist nötig, da sonst zu viele Tabs gleichzeitig geöffnet werden könnten, was sowohl CPU als auch RAM zu stark beanspruchen könnte. Deshalb sollten Verbindungen in etwa so häufig neu aufgebaut werden, wie alte Verbindungen beendet werden, sodass sich die Anzahl an Tabs möglichst konstant hält. Die Häufigkeit von Verbindungen hängt also damit zusammen, wie lange ein Tab geöffnet bleibt, das heißt, wie lange wir auf einer Seite verweilen. Da wir 10-20 Sekunden auf einer Seite verweilen, bedeutet dies also auch, dass wir ungefähr alle 10-20 Sekunden eine neue Verbindung aufbauen.

Zuletzt ist die Anzahl an Verbindungen für einen Tag limitiert. Dieses Limit hängt von dem Nutzerverhalten ab. Dazu ermitteln wir die Anzahl n an Seiten, die der Nutzer an einem durchschnittlichen Tag (während des eingestellten Zeitraumes) aufruft. FPFool soll nun maximal $x \cdot n$ zusätzliche Aufrufe an einem Tag durchführen, wobei $x \in \{1, 2, 3, 4, 5\}$ vom Nutzer ausgewählt werden kann.

4.4. Bestimmung der Suchbegriffe

Bevor wir unser Verfahren zur Generierung von Suchbegriffen anwenden können, müssen wir herausfinden, nach welchen Begriffen der Nutzer gesucht hat. Dazu analysieren wir die URLs aus dem Browserverlauf.

4.4.1. Analyse von URLs, um Suchbegriffe zu erhalten

Uns liegt der Browserverlauf des Benutzers vor, also die vom Nutzer besuchten URLs. Nun wollen wir anhand der gespeicherten URLs herausfinden, welche Suchbegriffe vom Nutzer gesucht worden sind.

Die URLs sind alle so ähnlich aufgebaut, wie in Abbildung 4.2 dargestellt. Sie beginnen mit dem verwendeten Protokoll (meistens http oder https), worauf dann der Hostname – also der Name der besuchten Seite – folgt. Dahinter kann dann noch ein Pfad angegeben sein, welcher zu der angeforderten Ressource führt. Außerdem besteht die Möglichkeit, Parameter anzufügen. Diese werden durch ein Fragezeichen hinter dem Pfad eingeleitet und sind jeweils ein Paar bestehend aus Schlüssel und Wert. Werden mehrere Parameter verwendet, so werden diese durch ein kaufmännisches Und getrennt. Wenn nun auf einer Seite nach

$$\underbrace{\text{https://}}_{\text{Protokoll}}\underbrace{\text{www.beispiel.de}}_{\text{Hostname}}\underbrace{\text{/pfad}}_{\text{Pfad}}\underbrace{\text{?s=beispiel\&p=test}}_{\text{Parameter}}$$

Abbildung 4.2.: Ein Beispiel für eine URL mit Parametern.

einem Suchbegriff gesucht wird, dann wird der Begriff – nachdem auf »Suchen« geklickt wurde und eine Weiterleitung stattgefunden hat – nahezu immer irgendwo als Parameter in der URL stehen. In seltenen Fällen können Internetseiten auch anders programmiert sein, aber diese Sonderfälle vernachlässigen wir. Wir können also annehmen, dass nach der Weiterleitung zu den Suchergebnissen in der URL irgendwo hinter dem Fragezeichen etwas wie x=*Suchbegriff* stehen muss, wobei x ein beliebiger Bezeichner sein kann. Da dieser Bezeichner von jeder Seite beliebig gewählt werden kann, können wir nicht einfach nach einem bestimmten Bezeichner suchen.

Stattdessen nutzen wir die folgende Vorgehensweise, um an die Suchbegriffe zu kommen: Zunächst prüfen wir, ob der Hostname der URL bereits in unserer Datenbank, welche die zugehörigen Suchparameter speichert, vorhanden ist. Diese Datenbank ist so aufgebaut, wie in Abbildung 4.3 dargestellt. Für jede Internetseite wird dort genau ein Suchparameter gespeichert, nämlich der Parameter, welcher für Textsuche verwendet wird. Falls der

Hostname	Suchparameter
www.beispiel.de	s
...	...

Abbildung 4.3.: Exemplarischer Aufbau der Datenbank für die Suchparameter von Webseiten.

gesuchte Hostname bereits vorhanden ist, lesen wir den zugehörigen Parameter einfach aus und durchsuchen die URL aus dem Verlauf nach diesem Parameter. Ist diese Suche ohne Erfolg, dann kann die Seite höchstwahrscheinlich nicht durchsucht werden und wir sind fertig. Bei erfolgreicher Suche können wir einfach den zugehörigen Wert zum Suchparameter auslesen, welcher sich hinter dem Gleichheitszeichen befindet.

Ist der Hostname noch nicht in der Datenbank vorhanden, dann rufen wir die Seite aus dem Verlauf auf, jedoch ohne Parameter. Wenn also die URL https://www.beispiel.de/

pfad?s=beispiel&p=test in dem Verlauf gefunden wird, dann rufen wir die Seite https://www.beispiel.de/pfad auf. Nun kann es der Fall sein, dass es nicht möglich ist, auf der Seite nach Begriffen zu suchen, das heißt, die Seite stellt kein Suchfeld zur Verfügung. In diesem Fall kann der Nutzer zuvor auch nichts auf der Seite gesucht haben. Wir müssen also keine Suchwörter ermitteln. Trotzdem merken wir uns, dass auf der Seite nicht gesucht werden kann, um in Zukunft Zeit zu sparen, falls der Nutzer dieselbe URL noch einmal aufrufen sollte. Dafür speichern wir für den Hostnamen eine leere Zeichenkette als Suchparameter ab.

Sollte es möglich sein, auf der Seite zu suchen, dann wählen wir eine zufällige Zeichenkette, geben sie in das Suchfeld ein und suchen danach. Wie genau die Suche automatisiert erfolgen kann, folgt in Kürze. Nun werden wir auf die Seite mit den Suchergebnissen weitergeleitet und unser Suchbegriff befindet sich irgendwo in den Parametern der URL, aber wir wissen noch nicht wie der zugehörige Schlüssel aussieht. Also durchsuchen wir die URL nach dem zuvor zufällig gewählten Suchbegriff. Wenn wir diesen gefunden haben, dann wissen wir, dass davor ein Gleichheitszeichen steht und vor diesem Gleichheitszeichen befindet sich der zugehörige Schlüssel. Wir speichern uns den Schlüssel jetzt in unserer Datenbank ab. Damit brauchen wir dieses Verfahren für den aktuell betrachteten Hostnamen zukünftig nicht zu wiederholen. Quelltext 4.1 stellt dieses Verfahren in Form von Pseudocode dar.

```
function getSearchTermFromUrl(url) {
    // Nur URLs mit Parametern sind interessant
    if (url.indexOf('?') < 0)
        return;

    // database speichert den Suchparameter fuer jede url
    if (database.contains(url)) {
        // Gibt den Wert fuer einen Suchparameter in der URL
        // zurueck
        return getSearchTermByParam(url, database.get(url));
    } else {
        // Erstellt neuen Tab mit angegebener URL
        var tab = createTab(url.substring(0, url.indexOf('?'));
        // Sucht auf der aktuellen Seite nach 'dummy term',
        // wodurch eine Weiterleitung zu den Suchergebnissen
        // stattfindet (die URL des Tabs aendert sich daher)
        searchTab(tab, 'dummy term');

        // Suche den zugehoerigen Parameter zu dem gesuchten Wert
        // key ist der Parameter, val der zugehoerige Wert
        var params = new URLSearchParams(tab.url.split('?')[1]);
        for (const [key, val] of params.entries()) {
            if (val == 'dummy term')
                database.store(url, key);
        }
```

```
    }
}
```

Quelltext 4.1: Pseudocode zur Analyse von URLs, um Suchbegriffe zu erhalten.

Nun gibt es auch Seiten, wo nicht nur nach einem Begriff gesucht werden kann, sondern auch noch weitere Suchangaben getätigt werden können. Beispielsweise kann der Nutzer auf einigen Seiten den Zeitraum der Suche einschränken oder die Suche auf spezielle Kategorien beschränken. Ein Beispiel dafür ist die Seite www.booking.com, auf der nach Unterkünften gesucht werden kann. Dort kann neben dem gewünschten Reiseziel (Suchwort) auch noch die Zeitspanne des Aufenthaltes angegeben werden. Wenn wir daran interessiert sind, alle Suchangaben zu erhalten, dann müssen wir die Parameter nicht nur nach einem Suchbegriff, sondern auch noch nach weiteren Angaben durchsuchen. Im folgenden Abschnitt werden einige Ideen beschrieben, wie dies umgesetzt werden könnte.

4.5. Nach Suchbegriffen suchen

Da wir über das Content Script Elemente auf Seiten ändern können, ist es möglich, Text in Suchfelder einzufügen und dann auch dafür zu sorgen, dass die Suche ausgeführt wird, indem wir einen Klick auf den zugehörigen Knopf simulieren. Die Suchfelder zu finden ist jedoch nicht so einfach, hierfür müssen wir etwas mehr Aufwand betreiben.

4.5.1. Suchfelder auf Seiten finden

Suchfelder sind in HTML Code einfache `<input>` Elemente. Genauer genommen sind Suchfelder in der Regel `<input type="search">` oder `<input type="text">` Elemente. Da es nun noch einen Knopf (oder ähnliches) geben muss, um die Suche zu starten, befinden sich das Suchfeld und der Knopf in der Regel in einem HTML Formular. Ein HTML Formular sendet die eingegebenen Daten an eine angegebene URL, sobald das Formular abgeschickt wird. Quelltext 4.2 zeigt den Aufbau eines HTML Formulars.

```
<form action="/search">
    <input type="search">
    <button type="submit">Suchen</button>
</form>
```

Quelltext 4.2: Aufbau eines HTML Formulars.

Über sogenannte CSS Selektoren können wir leicht an alle Elemente herankommen, die eine bestimmte Eigenschaft aufweisen. Ebenfalls können wir für jedes `<input>` Element das zugehörige HTML Formular erhalten. Wenn es ein `<input type="search">` Element gibt, dann nutzen wir dieses Feld für unsere Suche. Das heißt, wir fügen unseren Suchbegriff in das Feld ein und schicken das Formular ab. Wie dies umgesetzt werden kann, zeigt Quelltext 4.3.

```
// Suchfeld finden
var inputField = $(':input[type=search]').first();

if (inputField != null) {
    // Zuvor ermittelten Suchbegriff in das Suchfeld eingeben
    $(inputField).val(searchTerm);

    // Suche starten
    $(inputField).closest('form').submit();
}
```

Quelltext 4.3: Durchführung einer einfachen Suche mit Hilfe von jQuery, einer JavaScript Bibliothek, die uns die Manipulation der Elemente einfacher macht.

Nun gibt es aber relativ viele Internetseiten, die ihr Suchfeld nicht als Suchfeld kennzeichnen, sondern die ein einfaches `<input type="text">` Element verwenden. Jetzt könnte man das obige Verfahren einfach adaptieren und das erste `<input type="text">` Element wählen, wenn es kein als Suchfeld gekennzeichnetes Eingabefeld gibt. Dies führt aber häufig zu Problemen, da solche Eingabefelder oft für Registrierungen oder Logins verwendet werden und wir solche Felder nicht ausfüllen wollen. Deshalb stellen wir sicher, dass das HTML Formular nur genau ein Eingabefeld für Text enthält. Damit ist ausgeschlossen, dass wir Anmeldungsformulare oder Loginformulare ausfüllen. Die Wahrscheinlichkeit, dass wir auch tatsächlich ein Suchfeld erwischen, ist damit recht hoch, aber hundertprozentige Sicherheit haben wir dennoch nicht. Zudem wählen wir immer das erste gefundene Eingabefeld aus. Sollte eine Seite mehrere Eingabefelder haben und das Suchfeld ist unglücklicherweise nicht das erste dieser Felder, dann haben wir ebenfalls keinen Erfolg. Trotz vieler Experimente konnten wir keine weiteren Verbesserungen für dieses Verfahren herausfinden.

Suchformulare mit mehreren Suchangaben

Nun wurde im vorherigen Abschnitt bereits erwähnt, dass einige Seiten weitere Suchangaben erwarten können. Zum Beispiel können auf der Seite www.booking.com neben dem Reiseziel auch das Anreisedatum sowie das Abreisedatum angegeben werden. Es ist eine schwierige Aufgabe, alle solche Suchangaben herauszufinden. Ein möglicher Ansatz wäre es, alle Eingabefelder innerhalb des HTML Formulars zu ermitteln, wie in Quelltext 4.4 vorgestellt. Dazu gehen wir einfach alle Elemente innerhalb des HTML Formulars durch und fügen alle Eingabefelder zur Ausgabeliste hinzu. Wenn ein Element selbst noch weitere Elemente enthält (also Kinder besitzt), dann betrachten wir diese Kinder ebenfalls rekursiv.

```
var inputField = $(':input[type=search]').first();
var inputs = getAllInputsInForm($(inputField).closest('form'));
```

```
function getAllInputsInForm(form) {
    var inputs = [];

    $(form).children().each((index, elem) => {
        if ($(elem).is(':input')) {
            inputs.push(elem);
        } else if ($(elem).children().length > 0) {
            inputs.concat(getAllInputsInForm(elem));
        }
    });

    return inputs;
}
```

Quelltext 4.4: Ermittlung aller Eingabefelder eines HTML Formulars.

Mit diesem Verfahren finden wir alle Eingabefelder eines HTML Formulars heraus. Nun gibt es nach wie vor einige Probleme. Erstens kann es wieder vorkommen, dass eine Seite ihr Suchfeld nicht als Suchfeld markiert. In diesem Fall wählen wir wieder das erste Eingabefeld für Text aus, also `var inputField = $(':input[type=text]').first();`. Das hat zur Folge, dass wir nicht sicher sein können, ob das zu diesem Eingabefeld zugehörige Formular wirklich das Suchformular ist. Zweitens kann es nun wieder passieren, dass wir ein Loginformular erwischen. Dieses Problem könnte wahrscheinlich umgangen werden, indem wir sicherstellen, dass in dem Formular kein `<input type="email">` Element sowie kein `<input type="password">` Element vorkommen darf. Drittens kann es vorkommen, dass eine Seite weitere Angaben über ein Element entgegen nimmt, welches erst angeklickt werden muss, bevor Eingaben getätigt werden können. Dies ist beispielsweise bei www.booking.com der Fall. Dort gibt es eine Schaltfläche »Anreise – Abreise«, welche erst angeklickt werden muss, bevor ein Zeitraum ausgewählt werden kann. Solche Elemente zu erkennen ist äußerst schwierig, da die Seite JavaScript Code verwendet, um sie anzuzeigen. An dieser Stelle kommen wir mit unseren bisherigen Verfahren nicht weiter.

Wenn eine Seite ihren HTML Code dynamisch mit Hilfe von JavaScript verändert, dann ist eine Analyse des HTML Codes eine sehr schwierige Herausforderung. Deshalb setzt die in dieser Arbeit entwickelte Software FPFool kein Verfahren zur Erkennung von mehreren Sucheingaben ein. Uns reicht es bereits aus, nach Suchbegriffen zu suchen, ohne weitere Angaben zu übermitteln, wie es in Quelltext 4.3 umgesetzt wurde.

Selbst wenn wir alle Eingabeelemente finden würden, dann müssten wir noch die zugehörigen Suchparameter ausfindig machen, um Sucheingaben des Nutzers anhand der im Browserverlauf gespeicherten URLs herausfinden zu können. Die Suchparameter zu finden ist ebenfalls eine schwierige Aufgabe, da wir nun nicht einfach Begriffe eingeben können und die Seite nach diesen Begriffen durchsuchen können, wie wir es zuvor gemacht haben. Schließlich können andere Eingaben zum Beispiel in Form von Checkboxen entgegen genommen werden. Diese Eingaben in der URL wiederzufinden ist deutlich schwieriger.

5. Evaluation

Dieses Kapitel beschäftigt sich mit der Evaluation der entwickelten Browser Erweiterung FPFool. Wir wollen mit Hilfe von Experimenten herausfinden, ob FPFool in der Lage ist, die persönlichen Interessen des Nutzers zu verschleiern. Dafür wird zunächst der Aufbau der Experimente beschrieben, bevor es dann mit der Auswertung der Experimente weiter geht.

5.1. Aufbau der Experimente

Um möglichst präzise Annahmen darüber treffen zu können, welche Informationen über uns von Trackern abgespeichert werden, wollen wir personalisierte Werbeanzeigen manuell auswerten. Eine automatische Auswertung von Werbeanzeigen wäre äußerst schwierig umzusetzen, weshalb die Auswertung manuell erfolgt.

5.1.1. Konfiguration der Testumgebung

Für die Durchführung der Experimente verwenden wir zwei Systeme. Dabei nutzt eines dieser beiden Systeme die entwickelte Software FPFool und das andere nicht. Natürlich wäre es optimal, wenn beide Systeme dieselbe Hardware sowie dasselbe Betriebssystem verwenden würden. Dies kann leider nicht sichergestellt werden, sollte aber dennoch kein Problem darstellen. Das System, welches FPFool installiert hat, nutzt das Betriebssystem Windows 7 (Version 6.1.7601). Das andere System verwendet hingegen MacOS Mojave (Version 10.14.5).

Konfiguration des Google Chrome Browsers

Beide Systeme verwenden die neueste Version des Google Chrome Browsers (Version 76.0.3809.87). Es werden die Standardeinstellungen des Browsers verwendet, welche bei der Installation mitgeliefert werden. Der Browser wurde auf beiden Systemen frisch installiert und wurde zuvor noch nicht benutzt.

Konfiguration der Browser Erweiterung FPFool

Die Einstellungen innerhalb von FPFool selbst wurden für die Experimente folgendermaßen gewählt:

- Das Intervall zur Auswahl der zu betrachtenden Einträge des Browserverlaufes wurde auf 14 Tage (maximal möglicher Wert) festgelegt. Wir wollen den gesamten Verlauf, der während der Experimente entstanden ist, einbeziehen.
- Die maximale Anzahl an gleichzeitig geöffneten Tabs ist auf zehn (maximal möglicher Wert) beschränkt. Somit können möglichst viele Seiten gleichzeitig besucht werden.
- Für die maximale Anzahl an täglichen Verbindungen durch FPFool möchten wir verschiedene Werte ausprobieren. Dadurch versuchen wir herauszufinden, ob eine aggressivere Vorgehensweise (viele Verbindungen aufbauen) besser wirken kann als eine Vorgehensweise, bei der nur wenige Verbindungen aufbaut werden. Dafür soll der Faktor x die Werte 1 und 3 annehmen, sodass die maximale Anzahl an Verbindungen auf $x \cdot n$ begrenzt ist, wobei n die durchschnittliche Anzahl an tatsächlichen Verbindungen pro Tag durch den Benutzer in dem festgelegten Intervall entspricht.

5.1.2. Ablauf der Experimente

Auf beiden Systemen werden dieselben Aktionen manuell ausgeführt. Die Aktionen sollen realistisches Nutzerverhalten abbilden, das heißt, es werden Seiten geöffnet, Begriffe gesucht sowie Links angeklickt. Dies soll über einige Tage hinweg fortgesetzt werden. Darüber hinaus wollen wir beobachten, welche Werbeanzeigen wir auf den besuchten Seiten erhalten.

Da wir uns für Werbeanzeigen interessieren, wollen wir Seiten untersuchen, die möglichst viele Werbeanzeigen enthalten. Abbildung 5.1 listet einige Seiten auf, dessen Werbeanzeigen wir beobachten und miteinander vergleichen möchten. Diese Seiten wurden haupt-

Seite	Art
amazon.de	Online-Shop
booking.com	Buchung von Unterkünften
focus.de	Nachrichtendienst
google.com	Suchmaschine
tz.de	Nachrichtendienst
web.de	E-Mail Anbieter
wetter.com	Wetterdienst
youtube.com	Videoplattform

Abbildung 5.1.: Liste zu besuchender Seiten im Rahmen der Experimente, auf denen wir die Werbeanzeigen beobachten möchten.

sächlich anhand ihrer Beliebtheit in Deutschland ausgewählt [Ale19]. Zudem wurde bei der Auswahl darauf geachtet, dass ausreichend viele Werbeanzeigen vorhanden sind, weshalb

es auch mehrere Nachrichtendienste in der Liste gibt. Soziale Netzwerke wurden nicht mit in die Auswahl aufgenommen, da diese ein Login verlangen. Um soziale Netzwerke verwenden zu können, müssten wir also jeweils eine gefälschte Identität anlegen, was jedoch schnell auffallen kann und eine Sperrung des Accounts zur Folge haben kann. Damit uns solche Schwierigkeiten erspart bleiben, verzichten wir auf Seiten, die ein Login erfordern.

Während der Experimente wollen wir nun hauptsächlich die Seiten aus Abbildung 5.1 besuchen und dort gezielt nach einem bestimmten Thema suchen. Wir führen dabei mehrere Durchläufe mit verschiedenen Themen durch. Bei den Themen handelt es sich um

1. eine Reise nach Berlin und
2. einen Schuhkauf für Herren.

Während wir nach Begriffen suchen, besuchen wir auch weitere Seiten, die uns durch die Suchmaschine vorgeschlagen werden und die möglicherweise nicht in Abbildung 5.1 aufgelistet sind. Jedoch betrachten wir nicht die Werbeanzeigen auf diesen Seiten. Im kommenden Abschnitt wird detailliert beschrieben, wie genau vorgegangen wurde und welche Ergebnisse dabei zustande gekommen sind.

5.2. Auswertung der Experimente

Suche nach einer Reise nach Berlin

Wir beginnen mit der Suche nach einer Reise nach Berlin. Zunächst wurden einige Aktionen durch den Nutzer ausgeführt, welche detailliert in Abbildung 5.2 aufgelistet sind. Dabei war der Faktor x zur Limitierung der künstlich erzeugten Verbindungen auf 1 festgelegt. Das bedeutet, dass täglich genau so viele Verbindungen künstlich erzeugt werden dürfen, wie der Nutzer an einem durchschnittlichen Tag tatsächlich aufbaut.

Besuchte Seite	Ausgeführte Aktion
google.com	Suche nach »Hotel Berlin«
trivago.de	Suche nach Einzelzimmer in Berlin
hotel.check24.de	Suche nach Einzelzimmer in Berlin
booking.com	Suche nach Einzelzimmer in Berlin
booking.com	Einen Eintrag ansehen
wetter.com	Suche nach »Berlin«
google.com	Suche nach »YouTube«
youtube.com	Suche nach »Berlin«
youtube.com	Video auswählen und komplett ansehen (Werbung scheinbar noch nicht personalisiert)
youtube.com	Suche nach »Berlin Sehenswürdigkeiten«
youtube.com	Video auswählen und komplett ansehen (Werbung enthält nun booking.com, muss aber nicht personalisiert sein)
youtube.com	Weiteren Videovorschlag komplett ansehen

Browser neu starten (FPFool hat nun einen Verlauf zur Verfügung)	
google.com	Suche nach »Berlin Sehenswürdigkeiten«
google.com	Brandenburger Tor Vorschlag anklicken
berlin.de	Artikel zu Sehenswürdigkeiten in Berlin lesen
wikipedia.de	Artikel zu Sehenswürdigkeiten in Berlin lesen
focus.de	Seite durchstöbern, ohne etwas anzuklicken
tz.de	Seite durchstöbern, ohne etwas anzuklicken
hotel.check24.de	Seite durchstöbern, ohne etwas anzuklicken
youtube.com	Suche nach »Reiseziele Berlin«
youtube.com	Video auswählen und komplett ansehen (booking.com Werbung)
wetter.com	Werbung scheinbar nicht personalisiert (Autoversicherung auf dem einen System, Schuhe auf dem anderen)

Abbildung 5.2.: Besuchte Seiten durch den Nutzer bei der Durchführung der Experimente.

Während der Durchführung dieser Aktionen hat die Software FPFool ebenfalls einige Aktionen ausgeführt. Diese können in Abbildung 5.3 gefunden werden.

Seite	Gesuchte Begriffe durch FPFool
google.com	bauhaus, speedtest
wetter.com	bahn
youtube.com	regenradar, rossmann, wiking, bundesliga

Abbildung 5.3.: Gesuchte Begriffe durch die Software FPFool.

Es zeichnet sich bereits ab, dass einige Suchanfragen keinen Sinn ergeben, wie zum Beispiel die Suche nach dem Begriff »bahn« auf der Seite www.wetter.com, da dort nur nach Städtenamen gesucht werden kann. Zudem war das tägliche Limit an Verbindungen schnell ausgeschöpft und es konnten noch keine Unterschiede zwischen Werbeanzeigen festgestellt werden, die darauf hindeuten, dass FPFool eine Auswirkung auf die Werbeanzeigen gehabt hat.

Wir wollen ab jetzt nach Artikeln suchen, die in Online-Shops angeboten werden. Es ist nämlich schwierig, herauszufinden, ob die Werbung von booking.com wirklich durch unsere Aktivitäten angezeigt wurde, weil diese Werbung teilweise bereits erschienen ist, als noch gar keine Aktivitäten in diese Richtung stattgefunden hatten. Daher eignet sich booking.com doch nicht für unsere Experimente. Wenn wir aber Werbung für einen zuvor angesehenen Artikel in einem Online-Shop erhalten, können wir uns relativ sicher sein, dass diese Werbeanzeige durch unsere Aktivität zustande gekommen ist.

Suche nach Schuhen für Herren

Damit FPFool mehr Verbindungen aufbauen kann, soll das Limit nun erhöht werden. Dazu könnten wir einfach deutlich mehr manuelle Aktionen durchführen, weil das Limit davon abhängig ist. Eine Schwierigkeit dabei wäre jedoch, dass die Aktionen nicht mehr vernünftig dokumentiert werden könnten, da sie zu viel Platz wegnehmen würden. Deshalb wird der limitierende Faktor für künstlich erzeugte Verbindungen direkt auf 3 erhöht, auch wenn wir noch nicht viele Ergebnisse mit dem Faktor 1 sammeln konnten.

Da wir nun ein anderes Thema nutzen wollen, löschen wir den Verlauf, damit FPFool nicht weiterhin Seiten zum Thema Reisen besucht. Unsere Aktionen sollen nun vorwiegend auf google.com und amazon.de stattfinden. Wenn wir lediglich zwei Seiten betrachten, erleichtert dies die Dokumentation der Ergebnisse deutlich. Werbeanzeigen wollen wir nun ausschließlich auf den Seiten focus.de, spiegel.de, tz.de und web.de vergleichen. Diese Seiten dienen nur zum Vergleich von Anzeigen und werden nach einem Besuch direkt wieder aus dem Verlauf gelöscht, damit unsere Software FPFool sie nicht erneut besucht. Dies würde uns nicht weiterhelfen, da wir die Werbeanzeigen nicht sehen können, wenn FPFool die Seiten automatisiert besucht.

Unsere Aktionen finden nun hauptsächlich auf google.com und amazon.de statt, da wir nun nach Schuhen für Herren suchen. Dazu starten wir zunächst eine Suchanfrage bei Google nach Schuhen für Herren und besuchen dann nacheinander einige der Suchergebnisse. Wir besuchen die Seiten zalando.de, amazon.de, footlocker.de und nike.de, falls sie in den Suchergebnissen auftreten. Auf jeder dieser Seiten schauen wir uns zwei Produkte an. Zusätzlich ändern wir die Suchanfrage noch zwei Mal leicht ab, sodass wir am Ende nach den Begriffen »Herren Schuhe«, »Herren Schuhe Sommer« sowie »Nike Schuhe« bei Google gesucht haben. Dieses Vorgehen wiederholen wir nun mehrere Tage lang. Die Resultate des ersten Tages befinden sich in Abbildung 5.4. Die obere Tabelle zeigt, welche Begriffe durch FPFool auf welcher Seite gesucht wurden. Wenn Begriffe mehrfach aufgelistet sind, bedeutet dies, dass FPFool mehrfach danach gesucht hat. Die untere Tabelle listet die Werbeanzeigen auf, die wir jeweils erhalten haben.

Seite	Gesuchte Begriffe durch FPFool
amazon	hotmail, h&m, speedtest, hagebaumarkt, hermes, skyscanner, sane, hvv, samsung, hvv, spotify, drangstedt, wochenendticket, denizli, speedtest, skyscanner, hagebaumarkt, bankdrücken, h&m, speedtest, saturn
google	hermes, hagebaumarkt, speedtest, haspa, skyscanner, h&m, hsv, spotify, saturn, hagebaumarkt, parkett, hotmail, pinterest

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	TV-Streaming	Fotobuch und Schuhe auf amazon.de
tz.de	Outdoorbekleidung	Dell Laptop
focus.de	ADAC (Pannenhilfe)	ADAC (Pannenhilfe)
web.de	web.de Mobilfunk	Lotto (Glücksspiel)

Abbildung 5.4.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am ersten Tag.

Auf der Seite wetter.com haben wir nun ohne FPFool Werbung für Schuhe bekommen, mit FPFool jedoch nicht. Da ein einziger Versuch aber natürlich nicht aussagekräftig ist, wollen wir dasselbe Vorgehen nun über eine Woche hinweg täglich wiederholen. Die Ergebnisse des zweiten Tages sind in Abbildung 5.5 zu finden. Wir haben an diesem Tag grundsätzlich dieselben Aktionen wie am ersten Tag ausgeführt, jedoch haben wir nicht unbedingt exakt dieselben Produkte angeklickt. Zusätzlich wurden die Werbeanzeigen nun drei Mal am gleichen Tag miteinander verglichen, um mehr Ergebnisse zu erhalten (morgens, mittags und abends).

Seite	Gesuchte Begriffe durch FPFool
amazon	speedtest, skyscanner, spotify, spotify, spiegel, domino, spiegel, hotmail, moldaustausee, sheego
google	spotify, spotify, hermes, hsv, skyscanner, penny, hermes, spiegel, parkett, hvv

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	Gelenkschmerzen, Krankenversicherung und Nike Schuhe	Gumbies (Schuhe), Tagesgeldkonto und H&M Bekleidung
tz.de	Nike Jacken, Nike Schuhe auf footlocker.de und Nike Schuhe	Regalsysteme, Schuhe auf amazon.de und Dell Laptop
focus.de	keine Werbung, keine Werbung und Gehaltsvergleich	keine Werbung, 1&1 und ADAC sowie FRITZ!Box und ADAC
web.de	web.de Energie, web.de Bonusprogramm und web.de Mobilfunk	web.de Energie, web.de Energie und web.de Mobilfunk

Abbildung 5.5.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am zweiten Tag. Dieses Mal wurden die Werbeanzeigen drei Mal am selben Tag miteinander verglichen.

Es treten nun bereits häufiger Werbeanzeigen auf, die mit zuvor besuchten Seiten in Verbindung stehen. Es sieht ganz danach aus, als wären einige der Anzeigen tatsächlich personalisiert. Offenbar hatte FPFool einen Einfluss auf die Ergebnisse, denn das System mit FPFool erhält deutlich mehr dieser personalisierten Werbeanzeigen als das System ohne FPFool. Das könnte durchaus daran liegen, dass FPFool die manuell besuchten Seiten erneut besucht hat, sodass diese Seiten nun öfter besucht wurden als auf dem System ohne FPFool.

Da wir nun bereits zwei Tage in Folge intensiv nach ein und demselben Thema gesucht haben, führen wir nun einen Tag lang keine manuellen Aktionen durch und beobachten lediglich die Werbeanzeigen, die wir dabei erhalten. FPFool arbeitet weiter im Hintergrund, auch wenn wir keine neuen Aktivitäten ausüben. Die Ergebnisse sind in Abbildung 5.6 zu finden.

Seite	Gesuchte Begriffe durch FPFool
amazon	news, neumünster, smileys, speedtest, nospa, netflix, shein, spotify, netflix, sternzeichen, sport1, spiegel, nachrichten, spotify
google	hornbach, skyscanner, undertale, nachrichten, news, nike, hagebaumarkt, spotify

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	Autoversicherung, Autoversicherung und Mazda (Auto) sowie Mazda	Mazda, Mazda und Gumbies (Schuhe) sowie Mazda
tz.de	zalando, Abnehmen und Auto verkaufen	Mastercard, Halbleiter und Bundeswehr IT
focus.de	Saturn, Saturn und Saturn	Saturn, Saturn und Saturn
web.de	Aldi, Aldi sowie Parship (Partnervermittlung) und Aldi	Tschibo und Aldi, Lidl und Aldi sowie Tschibo und Aldi

Abbildung 5.6.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am dritten Tag. Auch dieses Mal wurden die Werbeanzeigen drei Mal am selben Tag miteinander verglichen.

Die Werbeanzeigen unterscheiden sich teilweise auch hier wieder zwischen dem System mit FPFool und dem System ohne FPFool. Die Unterschiede sind jedoch nur minimal. Außerdem fällt auf, dass die Anzeigen zum Thema Schuhe im Vergleich zum vorherigen Tag deutlich abgenommen haben. Dies könnte durchaus daran liegen, dass wir keine zusätzlichen Aktionen durchgeführt haben. Deshalb führen wir unsere Aktionen fortan täglich immer wieder aus. Die Aktionen bleiben dabei immer gleich und wir wiederholen das Vorgehen für weitere vier Tage, sodass wir am Ende insgesamt eine Woche lang täglich aktiv waren. Die Ergebnisse sind in Abbildung 5.7, Abbildung 5.8, Abbildung 5.9 und Abbildung 5.10 zu finden. Wir haben die Werbeanzeigen dabei wieder einmal täglich miteinander verglichen.

Seite	Gesuchte Begriffe durch FPFool
amazon	hlg, haspa, hotmail, versicherungsnummer, hagebaumarkt, haspa, haspa
google	ndr2, hsv

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	zalando.de Schuhe	Kaminausstellung
tz.de	zalando.de Schuhe	Snipes Schuhe
focus.de	Kochpfanne	Lotto
web.de	Aldi	Aldi

Abbildung 5.7.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am vierten Tag.

Am vierten Tag haben wir nun wieder mehr Werbeanzeigen für Schuhe erhalten. Zwischen den beiden Systemen gibt es jetzt stärkere Unterschiede in den beobachteten Anzeigen.

Auch wenn nur vier Werbeanzeigen miteinander verglichen wurden, sind doch 75% dieser Anzeigen zwischen den beiden Systemen verschieden. Auch hier sieht es wieder danach aus, als hätte FPFool einen Einfluss auf die Werbeanzeigen gehabt, weil das System mit FPFool erneut mehr Anzeigen für Schuhe erhalten hat. Die durch FPFool gesuchten Begriffe scheinen nach wie vor noch nichts mit den Inhalten der Werbeanzeigen zu tun zu haben.

Seite	Gesuchte Begriffe durch FPFool
amazon	hsv, haspa, dompfaff, hvv, haspa, hotmail, hvv, haspa, hagebaumarkt, haspa, liverpool, hagebaumarkt, baubeschlagshop, hochzeitsgeschenke, hagebaumarkt, parkplatz
google	nike, h&m, hsv, ndr, nospa

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	Kaminausstellung	Kaminausstellung
tz.de	Snipes Schuhe	Mercedes-Benz
focus.de	Treppenlift	Lotto
web.de	Aldi	Aldi

Abbildung 5.8.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am fünften Tag.

Am fünften Tag ist die Anzahl an Werbeanzeigen für Schuhe wieder von drei auf eins zurückgegangen. Einige der Werbeanzeigen unterscheiden sich nach wie vor zwischen den beiden Systemen, jedoch sind dieses Mal nur die Hälfte der vier Anzeigen verschieden. Die durch FPFool gesuchten Begriffe scheinen auch dieses Mal nicht die Werbung beeinflusst zu haben.

Seite	Gesuchte Begriffe durch FPFool
amazon	hltv, vsco, hlae, wetter, programmiersprache, livestream, liveticker, wetter, testamentsvollstrecker, hvv, hagebaumarkt, hagebaumarkt, klumpen, hagebaumarkt, haspa, wochenendtrip
google	sport1, ndr, hagebaumarkt, ndr

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	Snipes Schuhe	Kaminausstellung und Snipes Schuhe
tz.de	Snipes Schuhe	Mastercard
focus.de	Treppenlift	Treppenlift
web.de	Aldi	Aldi

Abbildung 5.9.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am sechsten Tag.

Ein ähnliches Resultat haben wir am sechsten Tag bekommen. Erneut unterscheiden sich die Hälfte der vier Werbeanzeigen zwischen den beiden Systemen. Wir haben nun wieder

drei Werbeanzeigen für Schuhe beobachten können. Da sich die manuell ausgeführten Aktionen aber nicht geändert haben, scheint die Anzahl der Anzeigen zum Thema Schuhe nicht unbedingt davon abzuhängen, wie oft der Nutzer nach Schuhen sucht.

Seite	Gesuchte Begriffe durch FPFool
amazon	kiellauf, lidl, haspa, indeed, kielius, kindergeburtstag, payback, derbe, h&m, hsv, postbank, medpex, hotmail, wahlomat, liveticker, ingolstadt, hansaland, wikipedia, lieferheld
google	hvv, netflix, ndr2

Seite	Werbung mit FPFool	Werbung ohne FPFool
wetter.com	Mazda und Eurowings	Kaminausstellung und Immobilien
tz.de	Eurowings	Snipes Schue
focus.de	Treppenlift	Treppenlift
web.de	Aldi	Aldi

Abbildung 5.10.: Begriffe, die durch die Software FPFool gesucht wurden (oben) sowie die Werbeanzeigen, die wir erhalten haben (unten) am siebten Tag.

Für den letzten Tag stellen wir fest, dass das System mit FPFool zwei Mal Werbung von Eurowings (Fluggesellschaft) erhalten hat. Zuvor hat FPFool bereits einige Male nach Begriffen gesucht, die mit Flügen in Verbindung stehen: Es wurde an vorherigen Tagen mehrfach nach »skyscanner« gesucht, was ein Portal zur Buchung von Flügen ist. Auch wurde am siebten Tag nach dem Begriff »kielius« gesucht, wobei es sich um einen Bus zum Hamburger Flughafen handelt. Es deutet sich also an, dass die von FPFool gesuchten Begriffe nun etwas mit den Inhalten der Werbeanzeigen zu tun haben könnten.

5.2.1. Fazit

Auch wenn die Experimente innerhalb gewisser Grenzen durchgeführt wurden, deuten die Ergebnisse doch darauf hin, dass FPFool einen Einfluss auf die angezeigten Werbeanzeigen gehabt haben könnte. Selbstverständlich sind deutlich mehr Ergebnisse über mehrere Wochen hinweg notwendig, um diese Vermutung endgültig bestätigen zu können. Wir konnten aber sehen, dass die Werbeanzeigen auf dem System mit FPFool oftmals andere waren als die Werbeanzeigen auf dem System ohne FPFool. Zum Schluss haben wir sogar beobachten können, dass das System mit FPFool Anzeigen erhalten hat, die inhaltlich mit zuvor gesuchten Begriffen zusammenhängen. Gleichzeitig hat das System ohne FPFool diese Anzeigen nicht erhalten, was ebenfalls dafür spricht, dass FPFool die Werbeanzeigen beeinflussen konnte.

Im nächsten Kapitel fassen wir die wesentlichen Erkenntnisse dieser Arbeit noch einmal zusammen und geben einen Ausblick darüber, wie die Software FPFool in Zukunft noch weiterentwickelt werden könnte, um Tracker noch gezielter zu täuschen.

6. Zusammenfassung und Ausblick

Zu Beginn haben wir gezeigt, wie Suchbegriffe aus URLs extrahiert werden können. Dadurch können wir für beliebige Internetseiten aus dem Browserverlauf herausfinden, nach welchen Begriffen der Nutzer auf diesen Seiten gesucht hat.

Darüber hinaus ist es uns gelungen, Vorschläge von Google zur Vervollständigung von Suchbegriffen zu verwenden, um daraus neue Suchbegriffe zu generieren. Ebenfalls wurde beschrieben, wie Suchfelder auf Internetseiten automatisiert gefunden werden können und wie automatisiert nach Begriffen gesucht werden kann. Dadurch ist es möglich, automatisiert neue Suchbegriffe zu generieren und nach diesen zu suchen, um Tracker zu täuschen.

Im nächsten Schritt könnte die Software FPFool noch weiterentwickelt werden, indem nicht nur nach Suchbegriffen gesucht wird, sondern auch auf Ergebnisse von Suchanfragen geklickt wird. Wir haben bei den Experimenten festgestellt, dass Werbeanzeigen oftmals von zuvor angeklickten Suchergebnissen abhängen. Die Schwierigkeit dabei liegt darin, den HTML Code der Internetseite so analysieren zu können, dass wir erkennen, wo sich die Suchergebnisse befinden. Wir könnten zwar theoretisch alle verfügbaren Links anhand des HTML Codes erhalten, jedoch ist es gefährlich, einen Link auszuwählen, wenn man sich nicht sicher ist, welche Aktion dadurch ausgelöst wird. Wenn der Nutzer auf einer Seite eingeloggt ist, so können Aktionen mit den Rechten des Nutzers ausgeführt werden. Beispielsweise könnten dadurch Daten gelöscht werden, die eigentlich nicht gelöscht werden sollen. Daneben können natürlich weitere unerwünschte Folgen jeglicher Art eintreten, wenn FPFool willkürliche Aktionen ausführen würde. Um dieses Problem zu eliminieren, müsste man also in der Lage sein, den HTML Code so zu interpretieren, dass man weiß, welche Folgen ein Klick auf einen Link hat. Dazu wären aber neue Techniken nötig, was nicht im Bereich dieser Arbeit liegt.

Möglicherweise könnte FPFool in der Zukunft auch noch erweitert werden, indem Suchbegriffe semantisch analysiert werden. Somit kann vermieden werden, dass automatisch generierte Suchbegriffe unpassend sind. Zum Beispiel könnte dadurch sichergestellt werden, dass nur nach Städtenamen gesucht wird, wenn ein Suchfeld für die Suche einer Stadt vorgesehen ist. Eine semantische Analyse von Suchbegriffen ist jedoch ein schwieriges Problem und liegt daher ebenfalls außerhalb des Bereichs dieser Arbeit.

Durch diese beiden Erweiterungen wäre FPFool dann in der Lage, noch gezielter nach Suchbegriffen zu suchen und sogar Suchergebnisse auszuwählen. Es wäre also möglich, einzelne Suchergebnisse aufzurufen, so wie ein echter Nutzer es tun würde. Dadurch könnte menschliches Nutzerverhalten sehr realitätsnah modelliert werden und ein Tracker noch effektiver getäuscht werden.

A. Verzeichnisse

Abbildungsverzeichnis

2.1. Beispielszenario für First-party Web Tracking. 4
2.2. Beispielszenario für Third-party Web Tracking. 5
2.3. Ein Beispiel für einen HTTP Accept Header. 7
2.4. Ein Beispiel für einen User Agent. 7
2.5. Ein Beispiel für eine IPv4-Adresse in Binärdarstellung. 8
2.6. Beispiel für den ersten Teil eines vollständigen Fingerprints. 13
2.7. Beispiel für den zweiten Teil eines vollständigen Fingerprints. 14

3.1. Ein Beispiel für die Änderung der Statistik besuchter Seiten. 20
3.2. Eine beispielhafte Tabelle angefragter JavaScript Dateien. 21
3.3. Ein beispielhafter Graph von angefragten JavaScript Dateien. 22
3.4. Eine beispielhafte Statistik besuchter Seiten mit Zeitfenstern. 23
3.5. Beispiel zur Generierung von neuen Suchbegriffen. 25

4.1. Ein Beispiel für einen Botschutz. 29
4.2. Ein Beispiel für eine URL mit Parametern. 30
4.3. Exemplarischer Aufbau der Datenbank für die Suchparameter von Webseiten. 30

5.1. Liste zu besuchender Seiten im Rahmen der Experimente. 36
5.2. Besuchte Seiten durch den Nutzer bei der Durchführung der Experimente. . 38
5.3. Gesuchte Begriffe durch die Software FPFool. 38
5.4. Ergebnisse der Experimente. 39
5.5. Ergebnisse der Experimente. 40
5.6. Ergebnisse der Experimente. 41
5.7. Ergebnisse der Experimente. 41
5.8. Ergebnisse der Experimente. 42
5.9. Ergebnisse der Experimente. 42
5.10. Ergebnisse der Experimente. 43

Quelltextverzeichnis

2.1. Ermittlung des User Agents mit dem JavaScript Navigator Objekt. 9

3.1. Pseudocode des vorgestellten Algorithmus zur Analyse des Browserverlaufes. 20
3.2. Pseudocode des vorgestellten Algorithmus zur Analyse von Third-party Requests. 22
3.3. Pseudocode des vorgestellten Algorithmus zur Generierung von Suchbegriffen. 24

4.1. Pseudocode zur Analyse von URLs, um Suchbegriffe zu erhalten. 31
4.2. Aufbau eines HTML Formulars. 32
4.3. Durchführung einer einfachen Suche. 33
4.4. Ermittlung aller Eingabefelder eines HTML Formulars. 33

Literaturverzeichnis

[Ale19] *Alexa - Top Sites in Germany - Alexa.* https://www.alexa.com/topsites/countries/DE, 2019. – [Online; Zugriff am 09.08.2019]

[AmI19] *Am I unique?* https://amiunique.org, 2019. – [Online; Zugriff am 18.03.2019]

[BCELBR17] Bujlow, Tomasz ; Carela-Espanol, Valentin ; Lee, Beom-Ryeol ; Barlet-Ros, Pere: A Survey on Web Tracking: Mechanisms, Implications, and Defenses. In: *Proceedings of the IEEE* 105 (2017), aug, Nr. 8, 1476–1510. http://dx.doi.org/10.1109/jproc.2016.2637878. – DOI 10.1109/jproc.2016.2637878

[BMFGI11] Boda, Károly ; Máté Földes, Ádám ; Gulyás, Gábor ; Imre, Sandor: User Tracking on the Web via Cross-Browser Fingerprinting, 2011

[Cen19] Centre, Ripe Network C.: *RIPE NCC IPv4 Available Pool - RIPE Network Coordination Centre.* https://www.ripe.net/publications/ipv6-info-centre/about-ipv6/ipv4-exhaustion/ipv4-available-pool, 2019. – [Online; Zugriff am 23.03.2019]

[CLW17] Cao, Yinzhi ; Li, Song ; Wijmans, Erik: (Cross-)Browser Fingerprinting via OS and Hardware Level Features, 2017

[Eck10] Eckersley, Peter: How Unique is Your Web Browser? In: *Proceedings of the 10th International Conference on Privacy Enhancing Technologies.* Berlin, Heidelberg : Springer-Verlag, 2010 (PETS'10). – ISBN 3–642–14526–4, 978–3–642–14526–1, 1–18

[Fin19] *Fingerprint.js - The most advanced open-source fraud detection JS library.* https://fingerprintjs.com/, 2019. – [Online; Zugriff am 04.04.2019]

[FPF19] *Malte311/FPFool: A Google Chrome browser extension to protect against browser fingerprinting.* https://github.com/Malte311/FPFool, 2019. – [Online; Zugriff am 05.08.2019]

[GBLB18] Gómez-Boix, Alejandro ; Laperdrix, Pierre ; Baudry, Benoit: Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In: *Proceedings of the 2018 World Wide Web Conference.* Republic and Canton of Geneva, Switzerland : International World Wide Web Conferences Steering Committee, 2018 (WWW '18). – ISBN 978–1–4503–5639–8, 309–318

[HSL+14] Hannak, Aniko ; Soeller, Gary ; Lazer, David ; Mislove, Alan ; Wilson, Christo: Measuring Price Discrimination and Steering on E-commerce Web Sites. In: *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC* (2014), 11, S. 305–318. http://dx.doi.org/10.1145/2663716.2663744. – DOI 10.1145/2663716.2663744

[LRB16] Laperdrix, P. ; Rudametkin, W. ; Baudry, B.: Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. In: *2016 IEEE Symposium on Security and Privacy (SP)*, 2016. – ISSN 2375–1207, S. 878–894

[LWD10] Liu, Chao ; White, Ryen W. ; Dumais, Susan: Understanding Web Browsing Behaviors Through Weibull Analysis of Dwell Time. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA : ACM, 2010 (SIGIR '10). – ISBN 978–1–4503–0153–4, 379–386

[May09] Mayer, Jonathan R.: *"Any person... a pamphleteer" Internet Anonymity in the Age of Web 2.0*, Woodrow Wilson School of Publicand International Affairs, Bachelorarbeit, 2009

[MBYS11] Mowery, Keaton ; Bogenreif, Dillon ; Yilek, Scott ; Shacham, Hovav: Fingerprinting Information in JavaScript Implementations. In: Wang, Helen (Hrsg.) ; IEEE Computer Society (Veranst.): *Proceedings of W2SP 2011* IEEE Computer Society, 2011

[MGEL12] Mikians, Jakub ; Gyarmati, László ; Erramilli, Vijay ; Laoutaris, Nikolaos: Detecting price and search discrimination on the Internet, 2012

[Moz19] *Navigator - Web API Referenz | MDN*. https://developer.mozilla.org/de/docs/Web/API/Navigator, 2019. – [Online; Zugriff am 01.07.2019]

[MS12] Mowery, Keaton ; Shacham, Hovav: Pixel Perfect : Fingerprinting Canvas in HTML 5, 2012

[Nie11] Nielsen, Jakob: *How Long Do Users Stay on Web Pages?* https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/, 2011. – [Online; Zugriff am 05.08.2019]

[NKJ+13] Nikiforakis, Nick ; Kapravelos, Alexandros ; Joosen, Wouter ; Krügel, Christopher ; Piessens, Frank ; Vigna, Giovanni: Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In: *2013 IEEE Symposium on Security and Privacy* (2013), S. 541–555

[NoS19] *NoScript - JavaScript/Java/Flash blocker for a safer Firefox experience!* https://noscript.net/, 2019. – [Online; Zugriff am 18.03.2019]

[PDA+16] Padmanabhan, R. ; Dhamdhere, A. ; Aben, E. ; claffy, k. ; Spring, N.: Reasons Dynamic Addresses Change. In: *Internet Measurement Conference (IMC)*, 2016

[SE01] Srisuresh, P. ; Egevang, K.: *Traditional IP Network Address Translator (Traditional NAT)*. RFC 3022. http://dx.doi.org/10.17487/RFC3022. Version: 2001 (Request for Comments)

[SES14] Schneider, Markus ; Enzmann, Matthias ; Stopczynski, Martin: *Web-tracking-report 2014*. Fraunhofer Verlag, 2014

[SN17] Starov, O. ; Nikiforakis, N.: XHOUND: Quantifying the Fingerprintability of Browser Extensions. In: *2017 IEEE Symposium on Security and Privacy (SP)*, 2017. – ISSN 2375–1207, S. 941–956

[Sta19] *Meistgenutzte Browser weltweit - Mai 2019 | Statista*. https://de.statista.com/statistik/daten/studie/158095/umfrage/meistgenutzte-browser-im-internet-weltweit/, 2019. – [Online; Zugriff am 05.08.2019]

[Til13] Tillmann, Henning: *Browser Fingerprinting: Tracking ohne Spuren zu hinterlassen*, Humboldt-Universität zu Berlin, Diplomarbeit, 2013

[W3T19] W3Techs: *JavaScript Market Report, March 2019*. https://w3techs.com/technologies/details/cp-javascript/all/all, 2019. – [Online; Zugriff am 21.03.2019]

[Woo12] Woodruff, Mandi: *Using LinkedIn? You're Twice As Likely To Report ID Theft*. https://www.businessinsider.com/linkedin-users-are-over-twice-as-likely-to-report-identity-theft-2012-3?IR=T, 2012. – [Online; Zugriff am 15.03.2019]

[YXY+12] Yen, Ting-Fang ; Xie, Yinglian ; Yu, Fang ; Yu, Roger P. ; Abadi, Martín: Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. In: *The 19th Annual Network and Distributed System Security Symposium (NDSS) 2012*, Internet Society, February 2012