Diplomarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

Entwicklung eines Browser Plug-ins für transparentes Surfen

Tobias Fielitz
Matrikelnummer: 3816539
t.fielitz@fu-berlin.de

Erstgutachterin: Prof. Dr. Ina Schieferdecker
Zweitgutachter: Dr.-Ing. Edzard Höfig

Berlin, den 1. Oktober 2012

Zusammenfassung

Beim Surfen im World Wide Web kommunizieren wir häufig nicht nur mit der aufgerufenen Webseite, sondern auch mit Drittanbietern. Diese Drittanbieter haben die Möglichkeit Benutzerprofile über uns zu erstellen. Ziel dieser Arbeit ist die Entwicklung eines Prototyps eines Browser Plug-ins zur Visualisierung von Informationen über die Webseite und beteiligte Drittanbieter. Dazu werden zunächst grundlegende Techniken des Internets erläutert, existierende Browsererweiterungen aus den Bereichen Transparenz, Sicherheit und Privatsphäre vorgestellt und die technischen Möglichkeiten und sammelbaren Informationen der verschiedenen Browser (Chrome, Internet Explorer, Firefox, Opera, Safari) verglichen. Anschließend wird die Entwicklung des Prototyps für Chrome beschrieben.

Eigenständigkeitserklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 1. Oktober 2012 Tobias Fielitz

Danksagung

Ich bedanke mich bei Prof. Ina Schieferdecker für Ihre Betreuung dieser Arbeit. Ebenso bedanke ich mich bei Dr. Edzard Höfig für seine konstruktive Beratung und erfreuliche Betreuung.
Ich bedanke mich bei meiner Familie, bei meinen Eltern für ihre lange Unterstützung während meines Studiums und den fortwährenden Glauben an meine Fähigkeiten. Ich bedanke mich auch bei meinen Geschwistern für ihr offenes Ohr und ihre Motivation. Meiner Tante, Edda Blaesing, danke ich für ihre Geduld beim Korrekturlesen dieser Arbeit.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **4**
2.1 Vertrauen 4
2.1.1 Im Internet 4
2.2 Kommunikation im World Wide Web 5
2.2.1 Verschlüsselung 6
2.3 Webtechnologien 8
2.3.1 HTML 8
2.3.2 CSS 9
2.3.3 JavaScript 11
2.3.4 Cookies, Flash-Cookies, localStorage und Web Datenbanken 13
2.4 Tracking 14
2.5 Black- und Whitelists 14
2.6 Add-ons, Extensions und Plug-ins 16

3 Relevante Erweiterungen: Stand der Technik **17**
3.1 Informationsvisualisierung 17
3.2 Blockieren 19
3.3 Säubern 20
3.4 Tracking 21
3.5 Web of Trust 23
3.6 Gegenüberstellung dieser Arbeit zu relevanten Erweiterungen 24

4 Sammelbare Informationen und Browserunterstützung **27**
4.1 IP-Adresse 27
4.2 URL 28
4.3 DNS-Auflösung 28
4.4 Antwort-Header 29
4.5 Antwortzeiten 29
4.6 Externe Abfragen 29
4.7 Externe Befehle 31
4.8 SSL-Zertifikate 32
4.9 Tabellarische Übersicht 32

5 Systementwicklung **35**
5.1 Ziele 35
5.2 Vergleich und Wahl des Systems 35
5.2.1 Entscheidung für ein System 35
5.2.2 System-spezifische Eigenschaften 36

5.3 Softwarearchitektur 36
5.4 Entwurf 39
5.5 Visualisierung der sammelbaren Informationen 42
5.5.1 IP-Adresse 44
5.5.2 URL, Schema und Domain 44
5.5.3 Antwort-Header 45
5.5.4 Cookies 45
5.5.5 Serveranzahl 45
5.5.6 Länderinformationen 46
5.5.7 Tracking von Benutzerverhalten 46
5.6 Implementierung 46
5.6.1 Erstellung einer neuen Session - Zusammenspiel der webRequest-Events 47
5.6.2 Konfigurationsmöglichkeiten 50
5.6.3 Blockieren von Elementen 51
5.6.4 Cookies 54
5.6.5 Übertragung der Daten an die Freie Universität Berlin 54
5.7 Wiederverwendbarkeit der Module 55

6 Fazit und Ausblick 56
6.1 Umsetzung der Ziele 56
6.2 Kritik am gewählten System 56
6.3 Erfahrungen 57
6.4 Weiterführende Arbeit 58
6.5 Fazit 59

7 Appendix 60
7.1 Listings 60
7.2 Quellen verwendeter Abbildungen 62
7.3 Weitere Abbildungen 63

1 Einleitung

In Deutschland nutzen laut einer Onlinestudie von ARD und ZDF (2012) [11] ca. 76% der Menschen das Internet. In den Altersgruppen von 14-19, 20-29 und 30-39 Jahren liegt der Anteil sogar fast bei 100%. Dabei sind die meisten mit einem PC (73%) und/oder Laptop (58%) online [12].
Doch wissen viele nicht, dass beim Surfen Daten anfallen und über sie gesammelt werden. Noch weniger haben Kenntnis darüber, zu welchen Zwecken diese benutzt werden und was mit ihnen geschieht. Häufig bemerken Benutzer nur Dinge wie „Der Online-Shop schlägt Artikel vor, die mir gefallen.“ oder „Mein Online-Radio spielt immer die Musik, die ich mag, auch wenn ich nicht explizit angegeben habe, welche Bands ich gerne höre“. Weniger offensichtlich ist, dass sich die eingeblendete Werbung dem aktuellen Interessengebiet anpasst. Dies kann von Vorteil sein, wenn sich der passende Artikel, Kontakt oder die geeignete Leistung unter den Anzeigen befindet; das Erstellen eines Profils bei diesen Beispielen kann also durchaus einen Gewinn für den Nutzer bringen.
Es gibt aber auch Instanzen, die Daten erheben, ohne dass daraus ein Vorteil für den Benutzer entsteht. Im Gegenteil, es kann sogar von Nachteil sein, wenn diese Informationen in die falschen Hände geraten (z. B. durch ein Versehen oder einen Einbruch in die Server). Diese Daten werden im Hintergrund und für den Nutzer (ohne weitere Werkzeuge) unsichtbar übertragen, ausgewertet und verarbeitet. Dieses Sammeln von Daten dient hauptsächlich den beteiligten Firmen, die so Profile, ohne dass der Nutzer davon Kenntnis erhält, erstellen können. Ist Ihnen die Person näher bekannt, z. B. weil sie einen Account bei der Firma besitzt, kann sie sogar genauer identifiziert werden.
Der Benutzer wird transparenter und kategorisierbar. Firmen wissen mehr ihn, als er vielleicht von sich bekannt geben möchte.
An dieser Stelle setzt diese Arbeit an. Sie soll verdeutlichen, welche Firmen möglicherweise den Benutzer überwachen und Daten von ihm sammeln können. Dabei geht es nicht nur darum, ein größeres Bewusstsein über Privatsphäre zu vermitteln, sondern auch die Möglichkeit zu bieten, die Überwachung einzuschränken, wenn nicht sogar komplett zu unterbinden.

Für einen geläufigen Webbrowser entsteht eine Erweiterung, die Informationen visualisiert, die für den Benutzer sonst nicht sichtbar gewesen wären. Dazu zählt, in welche Länder, an welche Firmen Anfragen (Requests) gestellt werden, die von einer aufgerufenen Webseite ausgehen, und welche Firmen dabei potentiell die Möglichkeit haben, den Benutzer zu überwachen. Webseiten werden „durchsichtiger“, transparenter.

Transparenz schafft Vertrauen! Der Slogan ist nicht nur in der Politik, der Wirtschaft oder in zwischenmenschlichen Beziehungen gültig; auch beim Surfen im Web können mehr Informationen über die Kommunikation mit Firmen oder Informationen über die Firmen selbst (mehr) Vertrauen schaffen.
Vertrauen kann sich durch die Visualisierung zusätzlicher Informationen aber auch verringern. Gibt beispielsweise eine Firma auf ihrer Webseite einer anderen Firma (einem Drittanbieter) die Möglichkeit, Benutzer zu überwachen (zu *tracken*), kann dies das Vertrauen in die erstgenannte Firma negativ beeinflussen.

In dieser Arbeit werden zunächst die grundlegenden Techniken des Internets erläutert. Die Anfrage einer Webseite wird, von dem initialen Aufruf bis zur Darstellung der Inhalte aus allen zugehörigen Quellen, aus denen Daten bezogen werden, beschrieben. Erläuterte Techniken sind dabei das IP-Protokoll, das HTTP-Protokoll inklusive der Sicherheitsschicht SSL und die Namensauflösung per DNS. Weiterhin wird erklärt, wie Webseiten aufgebaut sind und wie die Technologien HTML, CSS und JavaScript funktionieren und welche Sicherheitsrisiken sie beim Surfen im Web bergen. Dabei wird genauer auf Cross-Site-Scripting und verankerte Sicherheitsmaßnahmen wie die Same-Origin-Policy eingegangen.
Im dritten Kapitel werden vorhandene Erweiterungen aus dem Bereich Transparenz, Privatsphäre und Sicherheit vor- und gegenübergestellt. Es gibt Erweiterungen, die zusätzliche Informationen bereitstellen/visualisieren; dabei gibt es große Unterschiede bezüglich des Umfangs und der Auswahl der Daten. Leichtgewichtigere Erweiterungen zeigen beispielsweise nur das Land, in dem der Server der aufgerufenen Webseite steht, andere leiten auf die eigene Webseite weiter und zeigen dort viele Informationen bezüglich der IP-Adresse und der Domain wie z. B. den Internet Service Provider, *whois*-Informationen[1] und Aktivität in Sozialen Netzwerken.
Erweiterungen können auch (zusätzlich) die Kommunikation einschränken. Dies geschieht in der Regel durch Listen von blockierten Elementen.
Erweiterungen, die bekannte Tracking-Firmen visualisieren (und blockieren), werden in einem eigenen Abschnitt gesondert betrachtet. Sie zeigen Webbugs (auch Trackingpixel genannt) und liefern zusätzliche Informationen über die initiierenden Firmen.
Weiterhin ist es möglich, mit einigen Erweiterungen Surfspuren, die auf dem Rechner des Benutzers anfallen, zu löschen. Dazu gehören z. B. Cookies oder Einträge in der Browserhistorie. Abschließend wird das Web of Trust vorgestellt, eine Plattform, inklusive einer eigenen Erweiterung, zur Bewertung

[1] *whois* ist ein Protokoll über das Informationen zu Domains und IP-Adressen abgefragt werden können. Unter den Informationen können sich Name der Firma, Ansprechpartner, Telefonnummer, Adresse und weitere Daten befinden.

von Webseiten in verschiedenen Kategorien.
Das folgende Kapitel zeigt die unterschiedlichen technischen Funktionalitäten der fünf meistgenutzten Webbrowser und stellt diese gegenüber. Dabei wird gezeigt, welche technischen Möglichkeiten, z. B. Anfragen überwachen, Cookies auslesen oder Zugriff auf gespeicherte SSL-Zertifikate, die unterschiedlichen Browser bieten. Die daraus darstellbaren Informationen werden herausgearbeitet, und es wird auf weitere Eigenschaften, wie Betriebssystemverfügbarkeit und Anzahl der herunterladbaren Erweiterungen der verschiedenen Systeme eingegangen.
Kapitel fünf befasst sich mit der Systementwicklung. Zunächst werden die Ziele und Mindestvoraussetzungen erläutert und anschließend eine Entscheidung für ein System getroffen. Es folgen Implementierungsdetails, Architektureigenschaften, Designentscheidungen und Feinheiten des gewählten Systems. Dazu gehören die Erstellung einer neuen Session (Bündelung von zusammengehörigen Anfragen) und das Blockieren von ausgewählter Kommunikation.
Rückblickend wird die Arbeit im letzten Kapitel hinsichtlich des gewählten Systems und der Entwicklung kritisch beurteilt. Den Abschluss bildet ein Ausblick auf mögliche weiterführende Arbeiten.
Im Anhang finden sich Screenshots vieler der vorgestellten Erweiterungen, Abläufe von getätigten Anfragen sowie Quellenangaben der verwendeten Grafiken.

Das Ziel dieser Arbeit ist, nach Erläuterung der Grundlagen, Auswertung vorhandener Erweiterungen, Gegenüberstellung der Funktionalitäten der unterschiedlichen Webbrowser und dem anschließenden Entwurf für eine eigene Erweiterung, einen funktionierenden Prototyp für ein eigenes Browser Plug-in zu entwickeln.

2 Grundlagen

In dieser Arbeit werden aufgrund der Popularität [105] [106] [119] lediglich folgende Webbrowser/Systeme und deren Erweiterungen betrachtet: Google Chrome, Microsoft Internet Explorer, Mozilla Firefox, Opera und Safari.

2.1 Vertrauen

Im Duden [97] wird Vertrauen beschrieben durch „nur Gutes von jmdm. erwarten“ und „sich auf ihn, sein richtiges Handeln verlassen“. Beide Formulierungen beschreiben zwischenmenschliches Verhalten.
Vertrauen kann aber auch in Mensch-Maschine oder allgemeiner: Lebewesen-System Beziehungen eine Rolle spielen. Beispiele sind das Vertrauen in die Flugfähigkeit von Flugzeugen oder in ein politisches System wie die Demokratie. Dabei ist immer mindestens ein Lebewesen beteiligt, das vertraut.

Sowohl *Gutes* [114] als auch *Richtiges* kann von unterschiedlichen Menschen anders bewertet werden. Eine Handlung kann als *richtig*, von einem anderen Menschen aber als *falsch* empfunden werden und *Gutes* kann global betrachtet auch gut sein, dafür lokal aber schlecht und andersherum.
Die Beweggründe, wann und warum ein Mensch vertraut, sind sehr unterschiedlich. Menschen wachsen unterschiedlich auf, machen verschiedene Erfahrungen, sind unterschiedlich geprägt und haben gelernt, sich auf andere zu verlassen, oder wurden enttäuscht.
Es ist deshalb nicht möglich, allgemeingültig zu sagen, ob ein Mensch, eine Maschine oder ein System vertrauenswürdig ist oder nicht. Diese Entscheidung wird von jedem Menschen individuell getroffen [39].

2.1.1 Im Internet

Bei Vertrauen im Web handelt es sich um eine Beziehung zwischen Mensch und einem technischen System. Um einer Webseite vertrauen zu können, sind mehrere Voraussetzungen nötig:

- Der Benutzer versteht den Inhalt der Seite, sprachlich und inhaltlich.
- Der Benutzer ist sicher, dass er mit der gewünschten Gegenseite kommuniziert (Authentizität).
- Der Benutzer ist sicher, dass Daten, die er erhält, nicht während der Übertragung verändert wurden (Integrität).
- Der Benutzer erhält Kenntnis über alle Informationen (z. B. Anfragen, Speicherung von Cookies), die im Hintergrund ablaufen.
- Der Benutzer weiß, welche Daten, zu welchen Zwecken auf Servern gespeichert und in welchem Umfang sie verarbeitet werden (Privatsphäre).

2.2 Kommunikation im World Wide Web

Das Internet besteht aus vielen zusammengeschlossenen Rechnernetzwerken. Die Kommunikation zwischen den einzelnen Rechnern erfolgt über das Internetprotokoll (IP) [62]. Wesentlicher Bestandteil ist das *routing*, das Weiterleiten von IP-Paketen. Dieses findet anhand von IP-Adressen statt. Theoretisch erhält jeder beteiligte Rechner eine Adresse, über die er angesprochen werden kann. Aufgrund der Adressknappheit und der Architektur des Webs verbergen sich aber häufig mehrere Rechner hinter einer IP-Adresse.
In Version 4 (IPv4) des Internetprotokolls handelt es sich dabei um 32 Bit-Adressen aufgeteilt in 4 Blöcke zu je einem Byte. Geschrieben werden die Blöcke in Dezimaldarstellung, Beispiel: 160.45.117.200.
Seit längerem zeichnet sich ab, dass der IPv4 Adressraum nicht ausreichend ist. Auch deshalb wurde mit IPv6 [103] ein neuer Standard entwickelt, dessen Adressen 128 Bit groß sind. Geschrieben werden sie in 8 Blöcken, die aus jeweils 16 Bit bestehen; die Darstellung ist hexadezimal. Dabei werden die einzelnen Blöcke durch einen Doppelpunkt voneinander getrennt, Beispiel: 2001:0db8:0000:1337:0000:0000:0000:0017.
Mit IPv6 „[...] ist es theoretisch möglich, 655 570 793 348 866 943 898 599 Adressen pro Quadratmeter Erde zu vergeben“ [38].

Die wenigsten Nutzer geben allerdings IP-Adressen in ihren Browser ein. Wird keine Suchmaschine genutzt, geschieht die Navigation über URLs (**U**niform **R**esource **L**ocator) [111]. Eine URL kann wie folgt aussehen:

```
Schema://Domain:Port/Pfad?Query_String
```

Das Schema bestimmt das Protokoll (z. B. http, ftp) über das kommuniziert wird. Die Domain besteht aus mehreren Teilen (Beispiel: www.fu-berlin.de, third-level-domain . second-level-domain . top-level-domain). Der Port (dient der Zuordnung von Verbindungen zu Diensten) wird fast immer weggelassen, weil Webserver per Default (siehe RFC 2616 [102]) auf Port 80, bei SSL-Verbindungen auf Port 443, auf Anfragen warten. Nach einer Pfadangabe folgt der Query_String bestehend aus Name-Wert-Paaren.
Nachdem in den Browser eine URL eingegeben wurde, muss zu der Domain die IP-Adresse ermittelt werden. Ist diese nicht bereits lokal gespeichert, wird das DNS (**D**omain **N**ame **S**ystem) [94] [95] genutzt. Dabei wird der nächstliegende DNS-Server (z. B. der des Internetproviders) kontaktiert. Hat dieser die gewünschten Informationen nicht, fragt er auf der nächst höheren Ebene nach. Dieser Vorgang setzt sich fort, bis die angefragte IP-Adresse ermittelt wurde oder sicher ist, dass die Domain nicht mit einer IP-Adresse verknüpft ist.
Damit ein „Man-in-the-middle“-Angriff (durch jemanden, der die Daten abfängt und falsche Informationen weiterleitet) bei DNS-Abfragen keinen Erfolg hat, wurde das ursprüngliche DNS um DNSSEC [101] (**D**omain **N**ame

Security Extensions) erweitert, sodass durch den Einsatz von asymmetrischer Verschlüsselung die Authentizität der Daten verifiziert werden kann.
Wenn die IP-Adresse bekannt ist, öffnet der Browser einen TCP (**T**ransmission **C**ontrol **P**rotocol) Socket und öffnet via IP eine Verbindung zu dem Webserver. Diese Anfrage dient allerdings lediglich dem Aufbau der eigentlichen Verbindung zum Austausch der Daten. Die neue Verbindung, ebenfalls über TCP, wird auf einem anderen Port geöffnet, da nur eine Verbindung pro Port gleichzeitig aktiv sein kann und mehrere Anfragen seriell statt parallel abgearbeitet werden müssten, was nicht effizient wäre.

Der Browser schickt seine Anfrage gemäß HTTP (**H**yper**t**ext **T**ransfer **P**rotocol) [102]. Zu Beginn einer Anfrage steht eine der in HTTP spezifizierten Methoden (z. B. HEAD, GET, PUT, DELETE). Mit diesen ist es möglich, Inhalte vom Server abzufragen, hinzuzufügen oder zu löschen. GET und POST werden im Internet am häufigsten verwendet. Das folgende Beispiel fragt die Ressource *login* auf der Domain *www.exampledomain.net* an und sendet dabei die Parameter *username* und *password* mit.

```
GET   /login?username=foo&password=bar HTTP/1.1
HOST  www.exampledomain.net
```

Mit der Anfrage werden weitere Informationen (Header-Felder), wie Cookies, gewünschte Sprache, akzeptierte Medientypen und eine Beschreibung des Browsers mitgesandt.
Der Server antwortet mit Protokollversion, Statuscode und -meldung, eigenen Header-Feldern, wie: Servername und -version, Größe und Medientyp des mitgelieferten Inhalts und dem Inhalt selbst.
Dieser kann ein HTML Dokument, Stylesheet, JavaScript, Objekt (z. B. Flash-Objekt, Java-Applet) sein, welcher vom Webbrowser dargestellt wird.

2.2.1 Verschlüsselung

Eine vertrauenswürdige Verbindung zeichnet sich durch die Sicherstellung aus, dass der Benutzer auch wirklich mit der gewünschten Gegenseite kommuniziert und die ausgetauschten Daten nicht während der Übertragung geändert werden können.
Um dies zu erreichen, werden zwei Verschlüsselungsverfahren eingesetzt. Bei dem symmetrischen Verfahren haben beide Kommunikationspartner denselben Schlüssel, der zum Ver- und Entschlüsseln benutzt werden kann. Bei asymmetrischer Verschlüsselung hat eine Seite zwei Schlüssel: einen öffentlichen und einen privaten. Dabei können Informationen, die mit einem der beiden Schlüssel kodiert wurden, nur mit dem jeweils anderen dekodiert werden. Welche Rolle ein Schlüssel einnimmt, hängt von dem gewünschten Nutzen ab.
Sollen Daten unlesbar übertragen werden, verschlüsselt der Sender die Ori-

ginaldaten mit dem öffentlichen Schlüssel der Gegenseite. Diese können nur mit dem zugehörigen privaten Schlüssel, den nur der Kommunikationspartner besitzt, entschlüsselt werden.
Mit dem Verfahren ist es auch möglich, den Absender zu verifizieren. Dabei verschlüsselt der Absender seine angegebenen Informationen (Organisation, Domain) mit seinem privaten Schlüssel und sendet dieses Paket an den Empfänger. Diesem ist der öffentliche Schlüssel des Absenders bekannt und nur, wenn damit die Informationen wieder lesbar gemacht werden können, kann sich der Empfänger sicher sein, dass er mit dem gewünschten Absender kommuniziert.
Bei SSL wird eine Mischung aus symmetrischer und asymmetrischer Verschlüsselung sowie Absender-Validierung eingesetzt. Dabei wird nur zu Beginn der Kommunikation eine asymmetrische, danach aufgrund des geringeren Rechenaufwands eine symmetrische Verschlüsselung eingesetzt [16].
Damit im Web über eine sichere Verbindung kommuniziert werden kann, muss das System zunächst vorbereitet werden.
Es gibt Zertifizierungsstellen (*Certificate Authorities*, CAs), die meist kommerziell agieren, deren Root-Zertifikate im Browser enthalten sind.
Dafür generiert die CA zunächst ein Schlüsselpaar: Der öffentliche Schlüssel wird mit dem privaten Schlüssel signiert (digital unterschrieben, häufig wird nur ein Hashwert über Informationen der Organisation verschlüsselt) und an die Browserhersteller ausgegeben. Diese integrieren die CA-Zertifikate in ihre Webbrowser.
Ein Webserverbetreiber, der für seine Besucher eine verschlüsselte Verbindung bereitstellen möchte, generiert seinerseits einen öffentlichen und einen privaten Schlüssel. Den öffentlichen Schlüssel (sein Zertifikat) sendet er an die Zertifizierungsstelle und lässt ihn signieren. Die CA unterschreibt diesen Schlüssel mit dem eigenen privaten Schlüssel.
Damit ist die Vorbereitung abgeschlossen: Der öffentliche Schlüssel der CA befindet sich im Browser und damit auf dem Rechner des Clients und der öffentliche Schlüssel des Webservers ist mit dem privaten Schlüssel der CA signiert und liegt auf dem Webserver zur Auslieferung bereit.
Stellt nun ein Client eine Anfrage an den Webserver, z. B. um eine sichere Verbindung zu seiner Bank zu öffnen, bekommt er das Zertifikat (den öffentlichen Schlüssel des Webservers) ausgehändigt. Mit Hilfe des im Browser integrierten öffentlichen Schlüssels der CA kann er sicherstellen, dass das Zertifikat echt ist.
Anschließend generiert der Browser per Zufall einen Sitzungsschlüssel für die folgende symmetrische Kommunikation. Diesen kodiert er mit dem öffentlichen Schlüssel des Webservers und versendet das Ergebnis, das nur mit dem passenden privaten Schlüssel dekodiert werden kann.
Damit haben beide Kommunikationspartner denselben symmetrischen Sitzungsschlüssel, mit dessen Hilfe die folgende Kommunikation verschlüsselt abläuft.

In einer URL wird der Einsatz von SSL durch ein angehängtes „s“ im Schema (http**s**) angezeigt. Viele Browser zeigen zudem ein Schloss oder ein ähnliches Symbol in der Adresszeile und färben z. B. das Schema der URL grün, um deutlich anzuzeigen, dass die Verbindung sicher ist.

2.3 Webtechnologien

2.3.1 HTML

Hyper**t**ext **M**arkup **L**anguage [56] ist eine Auszeichnungssprache, die fast ausschließlich im Web eingesetzt wird. Sie dient zur strukturierten Darstellung von Texten, Bildern, Videos, Audioinhalten und Objekten (z. B. Flash [3], Java Applets [93]).
HTML besteht aus Elementen (z. B. <h1>,), auch *Tags* genannt, die mit Attributen versehen werden können. Elemente können sichtbare Inhalte, wie Bilder darstellen, Text beinhalten oder der Strukturierung anderer Elemente dienen. Ebenso wie XML (E**x**tensible **M**arkup **L**anguage) [68] hat sich HTML aus SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) [20] entwickelt. Es ist XML-ähnlich, unterscheidet sich aber in einigen Punkten. So ist z. B. in HTML für <br> oder <p> Elemente kein schließendes Tag nötig. Die Menge an Elementen ist in HTML fest vorgegeben, in XML können hingegen eigene Elemente definiert werden.
Bei XHTML (HTML auf Basis von XML) müssen entsprechend schließende Tags gesetzt werden, weil das Dokument sonst nicht wohlgeformt ist. Es wurde lange an XHTML2 [71], der Weiterentwicklung von XHTML1.1 gearbeitet. Die Arbeitsgruppe wurde aber 2009 zugunsten von HTML5 geschlossen [108].
HTML5, die Weiterentwicklung von HTML4.01 wurde von der WHATWG (**W**eb **H**ypertext **A**pplication **T**echnology **W**orking **G**roup) [60] begonnen. Diese wurde 2004 von Mitarbeitern von Apple [8], der Mozilla Foundation [61] und Opera Software [90] nach einem W3C (**W**orld **W**ide **W**eb **C**onsortium) [118] Workshop gegründet. Die Mitarbeiter waren mit dem Kurs des W3C bezüglich der Weiterentwicklung von HTML nicht zufrieden: sie sei zu wenig autorenbezogen [121].
Das W3C hat sich dem Druck der Browserhersteller gebeugt und HTML5 ist inzwischen eine W3C Recommendation (Empfehlung) mit dem Status „Last Call“ [98] (letzter Aufruf für Änderungsvorschläge). Das Ziel ist es, HTML5 2014 als fertige Recommendation zu veröffentlichen.
Große Teile von HTML5 funktionieren bereits in den meisten Webbrowsern.

Das Grundgerüst von HTML besteht normalerweise aus einem umschließenden <html> Element, in dem sich die <head> und <body> Elemente befinden. Im <head> Element werden externe Dateien wie Scripte oder Stylesheets referenziert, Metainformationen, wie Zeichenkodierungen festgelegt

und der Titel des Dokumentes gesetzt. Scripte oder Stylesheets können in entsprechenden Elementen auch direkt in das HTML-Dokument geschrieben werden. Der eigentliche Inhalt des Dokuments wird im <body> Element notiert.
Mit den in HTML5 eingeführten Elementen wie <header>, <footer> und <nav> wird die Struktur von HTML-Dokumenten stärker semantisch ausgerichtet und für Maschinen verständlicher.

Ein in HTML5 nicht mehr erlaubtes Element ist das <frames> Tag. Mit Hilfe dessen kann eine Webseite aus verschiedenen Teilen, deren Inhalte sogar von unterschiedliche Domains stammen können, zusammengesetzt werden. Dabei werden die Rahmen nicht überlappend platziert. Möglich ist dies mit dem weiterhin erlaubten <iframe> Element. Auch bei diesem können Inhalte von einer anderen Domain geladen und in die Webseite eingebunden werden.
Die iframes-Technologie schafft die Grundlage für *Clickjacking*. Dabei wird ein unsichtbares iframe über die Webseite, oder Teile davon, gelegt. Wenn der Benutzer denkt, er würde auf z. B. einen Link klicken, klickt er stattdessen in das iframe und startet eine ganz andere als die gewünschte Aktion, oder er gibt seine Bankdaten statt auf der ursprünglichen Webseite in ein unsichtbares Formular ein und sendet diese anschließend unabsichtlich an Dritte.
Neben der visuellen Repräsentation einer Webseite gibt es im Browser eine interne Darstellung, das **D**ocument **O**bject **M**odel (DOM). Mit Hilfe dessen können Manipulationen an der Webseite mit JavaScript vorgenommen werden und auch CSS-Eigenschaften wie Schriftart und -größe werden mittels des DOM vererbt. Abb. 1 zeigt exemplarisch ein DOM.

2.3.2 CSS

Cascading **S**tyle**s**heets (CSS) beschreibt eine deklarative Sprache zur Stilisierung von Dokumenten. CSS-Regeln haben folgendes Schema:

Listing 1: Schema von CSS

```
Selektor {
    Eigenschaft1: Wert1;
    Eigenschaft2: Wert2;
}
```

Styles können an mehreren Stellen im Dokument notiert werden:

- als externe Datei per <link> Element,
- im HTML <head> in einem <style> Element und
- direkt am Element in Form des *style*-Attributs.

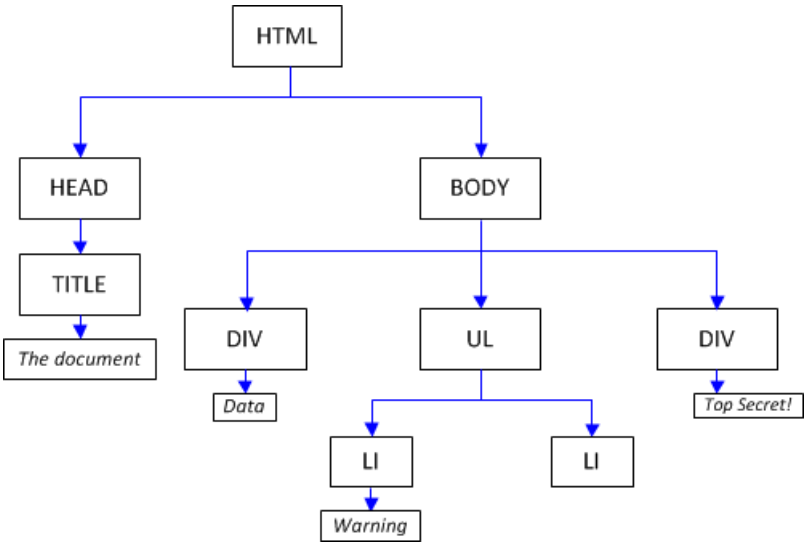

Abbildung 1: Exemplarische Abbildung eines DOM (Quelle: siehe Anhang, S. 62)

Selektoren sind z. B.
#id, ein Element mit einer eindeutigen id,
.class, alle Elemente die eine bestimmte Klasse besitzen,
*****, alle Elemente und
h2, alle Elemente vom Typ h2.
Eigenschaften sind beispielsweise: Höhe, Breite, Schriftart und -größe, Farbgebungen und Hintergrundbilder.
Wird dieselbe Eigenschaft mehrmals unterschiedlich definiert, überschreibt die Definition, die dem Element am nächsten ist, die jeweils anderen.
Ein Beispiel: Eine Regel wird für ein Elternelement festgelegt, die sich auf Kindelemente vererbt. Am Kindelement ist aber bereits eine Regel für dieselbe Eigenschaft als Attribut definiert; es gilt die letztere.
Durch CSS ist es sehr einfach möglich, ein Dokument für verschiedene Ausgabearten unterschiedlich zu stilisieren. So werden beim Drucken häufig Bilder weggelassen und Schriften stärker hervorgehoben.
CSS Level 3 ist seit dem Jahr 2000 in der Entwicklung, aber noch kein offizieller W3C Standard [30]. Dennoch lassen sich, ebenso wie bei HTML5, bereits einige Neuerungen wie das Transformieren (Bewegen, Drehen, Skalieren) von Elementen, für das bisher JavaScript eingesetzt werden musste, in vielen Browsern nutzen.

2.3.3 JavaScript

JavaScript ist eine imperative Programmiersprache, die überwiegend im Web eingesetzt wird. Der Programmcode läuft in der Regel beim Client im Webbrowser. Als Entwicklungsumgebung dient ebenfalls ein Browser.
Variablen in JavaScript sind nicht getypt und die Sprache ist turing-vollständig.
Abgesehen von den 2009 erstmals vom W3C aufgeführten *web workers* [59] gibt es keine parallele Ausführung von Programmcode in JavaScript. *Events*, hervorgerufen durch z. B. *clicks*, *timer*, *web worker messages* oder Ajax-Anfragen, werden in den Programmablauf „eingeschoben". Synchronisationsmechanismen wie z. B. *join* in *Java* [92] gibt es in nicht. Die eingesetzte Methode für Fälle, in denen z. B. auf den Rückgabewert „gewartet" werden soll, sind *callbacks*: eine Funktion, die als Parameter übergeben und, bevor die ursprüngliche Funktion zurückkehrt, mit diesem Wert aufgerufen wird. Sollen mehrere asynchrone Funktionen nacheinander ausgeführt werden, müssen diese geschachtelt werden.
JavaScript wird im Browser aus Sicherheitsgründen in einer Sandbox [89] ausgeführt, sodass nicht die Möglichkeit besteht, auf Komponenten, wie z. B. das Dateisystem, außerhalb der Sandbox zuzugreifen.
Im World Wide Web war es anfangs möglich, JavaScript im Browser zu deaktivieren, ohne auf einen großen Teil der Funktionalität zu verzichten. Bei den meisten Webseiten änderte sich nichts an der Funktionalität, weil wenig JavaScript eingesetzt wurde und der Großteil der Funktionalität in HTML integriert war.
Mit dem „Web 2.0" [112] und der Verfügbarkeit von JavaScript-Bibliotheken wie *Prototype* [100], *jQuery* [110] und dem *Dojo Toolkit* [109] hat der Einsatz von JavaScript stark zugenommen.
Viele Webseiten sind nur noch eingeschränkt nutzbar, wenn JavaScript deaktiviert ist.
Bei vielen Browsern sind aufgrund von Benutzerfreundlichkeit einige Möglichkeiten von JavaScript standardmäßig deaktiviert. Dazu gehören: die Fenstergröße zu ändern, neue Fenster (Pop-ups) zu öffnen und die Möglichkeit, das Kontextmenü abzuschalten.
Im Trend liegen JavaScript-Compiler und -Packer, die ein JavaScript umwandeln, sodass es effizienter und/oder kleiner (es benötigt weniger Speicherplatz, und die Übertragung ist schneller) wird. Diese Transformation macht es für Menschen fast unmöglich, solche Scripte zu lesen.
Ein Sicherheitsproblem, das häufig im Zusammenhang mit JavaScript genannt wird, ist Cross-Site-Scripting (XSS). Dabei wird z. B. durch ein Formular in einer Foren-Software JavaScript-Code persistent auf dem Server gespeichert und so immer wieder ausgeliefert. Dabei hat dieser Programmcode dieselben Ausführungsrechte und -möglichkeiten wie ein regulär ausgeliefertes JavaScript.

Dieses eingebettete Schadscript könnte Benutzerdaten (z. B. Loginname, Passwort) abgreifen oder stellvertretend für den Benutzer handeln.
Gegen diese Art von Angriff hilft keine verschlüsselte Verbindung, sondern nur eine saubere Prüfung von Benutzerinhalten von Seiten des Servers.
Scripten von anderen Domains ist es aufgrund der „Same Origin Policy“ nicht gestattet, auf Webinhalte der Wurzeldomain zuzugreifen. Dies gilt für Frames und iFrames. Wird der Versuch unternommen, erscheint eine Fehlermeldung. Diese ist im Folgenden dargestellt.

```
Unsafe JavaScript attempt to access frame with URL
http://[..].dyndns.org/frames/transparency.html
from frame with URL
http://page.mi.fu-berlin.de/[..]/da/injected.html.
Domains, protocols and ports must match.
```

Wenn das Script hingegen im selben Dokument eingebettet ist, hat es dieselben Rechte wie ein Script, das von der ursprünglichen Domain stammt.
Die Sicherheit ist nicht mehr gewährleistet, wenn eine verschlüsselte Seite ein Script von einer unverschlüsselten Seite nachlädt. Google Chrome (Vers. 18.0.1025.168, Ubuntu 11.10) informiert den Benutzer darüber und empfiehlt, das Script nicht zu laden. Mozilla Firefox (Vers. 10.0, Ubuntu) und Opera (Vers. 11.61, Linux i686) laden und führen es ohne Nachfrage aus.

Ein leichtgewichtiges Datenformat, aus dem JavaScript-Sprachraum, ist die **J**ava**S**cript **O**bject **N**otation (JSON) [64]. JSON setzt sich aus zwei Strukturen zusammen: Name-Wert-Paaren, auch bekannt unter dem Namen Assoziative Arrays und einfachen Listen. Das Format eignet sich zur Datenübertragung, auch zwischen unterschiedlichen Systemen und Programmiersprachen, und zur begrenzten Serialisierung von Objekten. Objekt-Methoden werden dabei ignoriert. Um ein JavaScript-Objekt in einen JSON-String umzuwandeln gibt es die Methode *JSON.stringify()*. Um die Operation umgekehrt stattfinden zu lassen, muss *JSON.parse()* aufgerufen werden
Da JSON teil von JavaScript ist, können einfach neue JSON-Objekte erstellt werden. Ein Beispiel zeigt Listing 2.

Listing 2: Beispiel JSON-Datenstruktur

```
var jsonObject = {"key": ["item1", "item2", "item3"],
   "hello": function(){console.log("Hello World")}};

var jsonString = JSON.stringify(jsonObject);
-> '{"key": ["item1", "item2", "item3"]}'
```

JSON wird in dieser Arbeit zur Übertragung gespeicherter Sessions (siehe Kapitel 5.4 und 5.6.5) genutzt. Auch die Abfrage des WOT-Rankings (siehe Kapitel 3.5) liefert eine Antwort im JSON-Format. JSON-Bibliotheken gibt es für diverse Programmiersprachen, z. B. für C, Java, PL/SQL, Python.

2.3.4 Cookies, Flash-Cookies, localStorage und Web Datenbanken

Ein Server bekommt bei einer Anfrage von einem Client mehrere Informationen über HTTP mitgeliefert: IP-Adresse, User-Agent, Accept-Language und weitere, die Auskunft über den Client geben. Alle Parameter zusammen ergeben eine gewisse Sicherheit, den Client eindeutig zu identifizieren. Diese ist aber nicht zuverlässig. Beispielsweise kann sich aufgrund eines *DISCONNECTS* beim Client die IP-Adresse ändern, oder mehrere Rechner benutzen die gleiche IP-Adresse, weil sie über denselben Router kommunizieren (NAT – **N**etwork **A**ddress **T**ranslation) [96].
Eine eindeutige Identifizierung wird dadurch erreicht, dass der Server Informationen, meist in Form einer Session-ID, die wenige Bytes groß ist, beim Client ablegt. Diese wird dann bei jeder Anfrage an den Server übertragen. Dort können beliebig viele Daten mit der Session-ID verknüpft sein.
Das häufigste Verfahren, das für die clientseitige Speicherung genutzt wird, sind Cookies. Dabei weist der Server dem Antwortfeld „Set-Cookie“ einen String zu. Diese Informationen werden dann bei jeder folgenden Anfrage des Clients per „Cookie“-Header an den Server übermittelt.
Informationen können beim Client auch über Flash-Cookies, den *localStorage* [58] und Web-Datenbanken gespeichert werden. Um diese Informationen abzufragen muss ein entsprechendes Flash-Script bzw. JavaScript eingebunden werden.
Flash-Cookies können nicht wie normale Cookies über die Browsereinstellungen verwaltet werden. Sie werden zentral vom Flash-Plug-in [3] verwaltet und sind somit auch über verschiedene Browser hinweg gültig.
Im *localStorage*-Objekt per JavaScript als Schlüssel-Wert-Paare gespeicherte Informationen sind ebenfalls nach Beendigung des Browsers permanent gespeichert. Gegenüber Cookies bietet der *localStorage* mehr Speicherplatz und läuft nicht nach einer bestimmten Zeit ab.
Web-Datenbanken können ebenfalls clientseitig abgefragt werden allerdings nicht per einfachem Schlüssel-Wert-Prinzip, sondern per SQL (**S**tructured **Q**uery **L**anguage). Der verwendete Standard ist aber nicht bei allen Browsern gleich [57].
Verfahren der Wiedererkennung werden ständig im Internet eingesetzt. Sie dienen neben dem Aufbau einer Login-Session und dem Speichern von Artikeln in einem virtuellen Warenkorb, dem Überwachen und Protokollieren von Nutzerverhalten.

2.4 Tracking

Um Benutzerprofile, z. B. für Werbenetzwerke, zu erstellen, ist es für die Betreiber (Drittanbieter) wichtig zu wissen, zu welchen Zeiten, für welchen Zeitraum und auf welchen Seiten sich ein Benutzer aufgehalten hat.
Dazu wird auf einer Partner-Webseite eine Ressource vom einem Drittanbieter eingebunden. Diese ist häufig ein Script, ein iFrame oder ein 1x1 Pixel großes Bild (Zählpixel), das für den Anwender unsichtbar ist. Der Browser sendet während der Anfrage einen ggf. zuvor gespeicherten Cookie an den Server (der zeitliche Ablauf ist in Abb. 2 dargestellt). In der Anfrage ist enthalten (z. B. in die URL kodiert), auf welcher Seite die Ressource eingebunden ist. Das folgende Beispiel, Listing 3 zeigt exemplarisch ein eingebettetes Bild-Element.

Listing 3: Beispiel für einen eingebetteten Zählpixel

```
<img src="www.trackingcompany.com/transparent_pixel.
    jpg?source=www.partnerwebseite.de&sport&fussball">
```

Damit kennt die überwachende Instanz die abgefragte Ressource, den Zeitpunkt und kann den Benutzer, per zuvor gesetztem Cookie, wiedererkennen. Wenn die Ressource kontinuierlich abgefragt wird, kann sogar die ungefähre Verweildauer festgestellt werden. Je häufiger kommuniziert wird, desto genauer ist der Wert.
Wie im vorherigen Abschnitt erwähnt, kann ein Benutzer auch über andere Verfahren als Cookies wiedererkannt werden.
Zwei große Firmen, die Tracking betreiben, sind Google und Facebook. Beide setzen Tracking ein, um Benutzern personalisierte Werbung zu zeigen [35] [41]. Dabei werden Benutzer mit Hilfe von z. B. Cookies über mehrere Webseiten (auch über mehrere Webserver) hinweg wiedererkannt. Anhand der Inhalte der aufgerufenen Seiten, kann ein Benutzerprofil erstellt werden. Daraus ergibt sich die Möglichkeit, personalisierte statt nicht personalisierter Werbung zu zeigen. Die Folge ist eine Erhöhung der Klickrate auf die eingeblendete Werbung. Je mehr Klicks getätigt werden, desto höher ist der Gewinn der Werbetreibenden.

2.5 Black- und Whitelists

Blacklists werden eingesetzt, um bestimmte Personen, Dinge, Elemente in einem bestimmten Kontext zu benachteiligen oder sogar auszuschließen. Viele der in dieser Arbeit vorgestellten Browser-Erweiterungen (siehe Kapitel 3) setzen Blacklists ein, um die Kommunikation zu bestimmten Domains oder Ressourcen zu unterbinden. Dabei können die Einträge auch z. B. Reguläre Ausdrücke für URLs sein.
Elemente, die hingegen auf einer Whitelist stehen, werden in einem bestimmten Kontext bevorzugt behandelt.

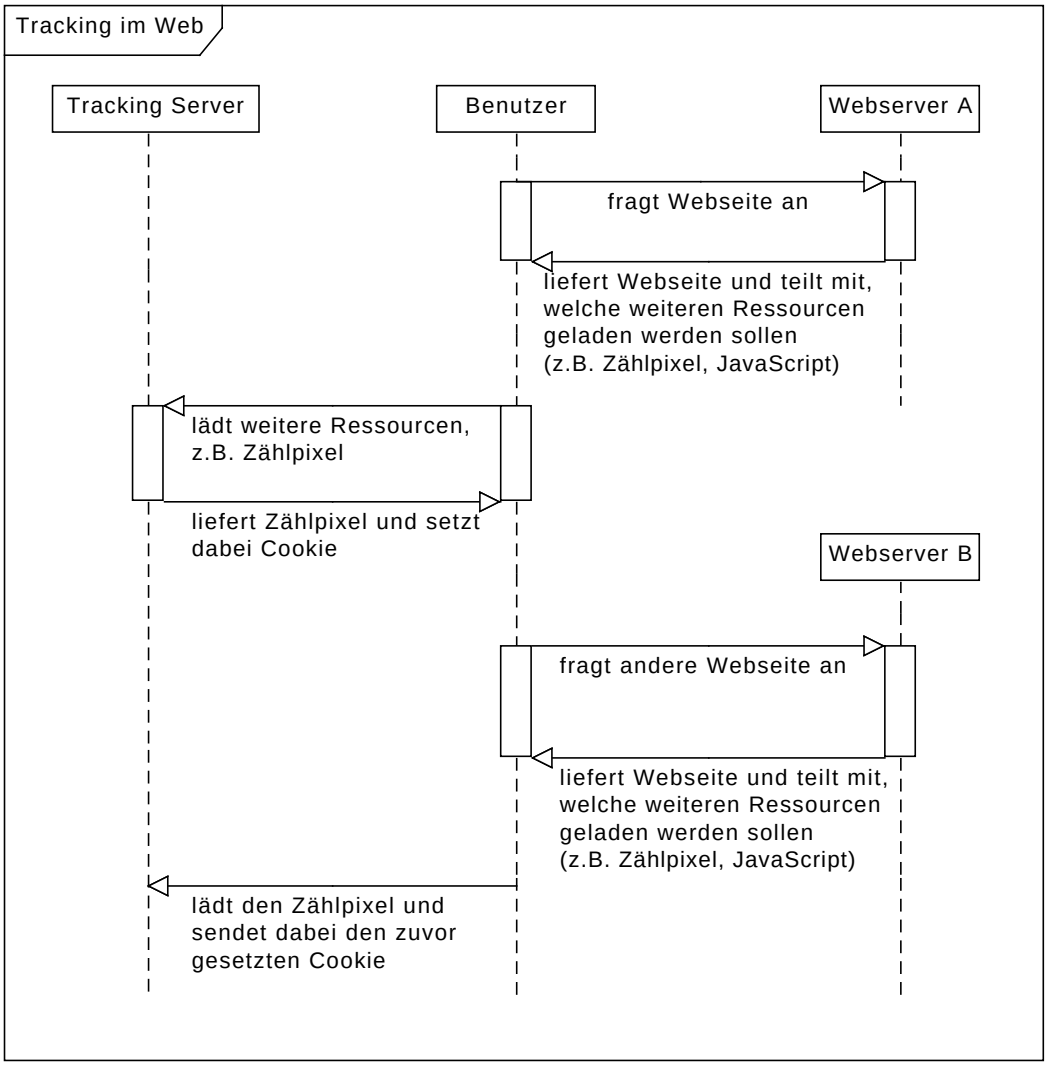

Abbildung 2: Tracking im Web

2.6 Add-ons, Extensions und Plug-ins

Im Softwarebereich handelt es sich dabei um Zusatzprogramme, die die eigentliche Software erweitern. Die Software ist - in der Regel - aber auch ohne Add-on, Extension und Plug-in lauffähig.
Apple, Google, Mozilla und Opera benutzen dieselben Bezeichnungen, um Zusatzprogramme für ihre Browser zu kategorisieren [9] [47] [78] [91].

- Extensions können mit dem Webinhalt interagieren, Webseiten verändern und ganze Seiten oder Teile davon blockieren, Daten von anderen Servern laden und zusätzlichen Inhalt anzeigen. Extensions bestehen häufig aus HTML, CSS und JavaScript.
  Die sichtbaren Elemente einer Extension bilden ein Icon, ein Pop-up und eine Konfigurationsseite.

- Plug-ins sind Zusatzprogramme, die einen oder mehrere Medientypen darstellen können. Beispiele sind: das Flash Player Plug-in oder das Adobe Reader Plug-in zur Darstellung von PDFs im Browser.

- Add-on wird als Oberbegriff für Zusatzprogramme gebraucht und bezeichnet im Allgemeinen Extensions, Plug-ins, Sprachpakete und Themes (Stilisierungen des Browserfensters) [78].

3 Relevante Erweiterungen: Stand der Technik

Unter den vielen existierenden Browser-Erweiterungen gibt es bereits einige, die mehr Transparenz schaffen. Aufgrund der Nähe zu den Bereichen Sicherheit und Privatsphäre werden an dieser Stelle auch andere Erweiterungen betrachtet. Diese lassen sich in folgende Kategorien einteilen (einige Erweiterungen fallen in mehrere Kategorien): **Informationsvisualisierung**, **Blockieren**, **Säubern** und **Tracking**. Zusätzlich gibt es Erweiterungen, wie Web of Trust (siehe Abschnitt 3.5), die die Möglichkeit bieten, Webseiten aufgrund verschiedener Kriterien zu bewerten, und Erweiterungen, die auf Sicherheitsrisiken der aktuellen Seite hinweisen (z. B. die Möglichkeit, dass Passwörter unverschlüsselt übertragen werden könnten).

3.1 Informationsvisualisierung

Kriterien wie: Inhaber der Domain, IP-Adresse, Positionsanzeige, (Darstellung auf einer Karte, ggf. mit exakter Adresse) werden sichtbar gemacht. Die folgenden Erweiterungen visualisieren Informationen, die direkt mit der aufgerufenen Webseite verknüpft sind.

WorldIP und FlagFox [5] [22] zeigen für die aktuelle Seite eine Landesflagge in der Adresszeile des Browsers. Im Pop-up werden Domain, IP-Adresse und Land angezeigt. WorldIP zeigt zusätzlich den Ländercode und den **I**nternet **S**ervice **P**rovider (ISP) und stellt die genannten Informationen auch für die eigene IP-Adresse bereit. Beide Erweiterungen wurden für Mozilla Firefox entwickelt.

Chrome Flags [66] ändert das Icon ebenfalls in eine Landesflagge, basierend auf der IP-Adresse der Webseite. Die Zuordnung geschieht aufgrund einer mitgelieferten Tabelle. Das Pop-up zeigt neben dem Land auch die Stadt und die Region an, in der sich der Webserver befindet. Weitere angezeigte Informationen sind der Google PageRank (Bewertungsindikator für die Relevanz einer Webseite), der Alexa Rank [4] und die WOT Bewertung (siehe Abschnitt 3.5). Alle angegebenen Daten sind mit den entsprechenden Webseiten verlinkt, auf denen zusätzliche Informationen eingeholt werden können.

IP Address and Domain Information [117] bietet die umfangreichste Darstellung von Informationen bezüglich der Domain und der damit verknüpften IP-Adresse. Hinter der Erweiterung steht die Webseite `tcpiputils.com`, auf der die angezeigten Informationen gesammelt und vorbereitet werden.
Angezeigt werden auf Basis der IP-Adresse, der ISP, ob die Adresse auf einer SPAM- oder Blocklist steht, die kompletten *whois*-Informationen und

die Position der Server auf einer Karte. Für die Domain werden diverse Informationen bereitgestellt, unter anderem: Google PageRank, Web of Trust Ranking, Aktivität in Sozialen Netzwerken wie Google+ [48] und Facebook [34], die Anzahl der Tweets [113] und Charts über die Backlink-Entwicklung und die Entwicklung des täglichen Traffics. Für die Charts werden externe Quellen herangezogen [70] [17].
Die Erweiterung funktioniert nicht nur für IPv4-Adressen, sondern auch für IPv6.
Neben den Informationen über die aufgerufene IP-Adresse kann man sich auch Informationen über die eigene IP-Adresse anzeigen lassen. Die Positionsbestimmung der Adresse der FU (Freien Universität, 87.77.78.245) war allerdings falsch, ihr wurde eine Position in Bossier City, Louisiana, United States (US) zugeordnet.
Schon auf der Startseite von `tcpiputils.com` sind folgende Dienste aufgelistet: *whois*, *traceroute*, IP Geo Lookup (liefert die Geo-Koordinaten), DNS Lookup, E-Mail Test, DNS Blackhole List, *ping*, Domain Neighbors (andere Domains, die unter der IP-Adresse erreichbar sind), W3C Validator, Geo Targeting SEO (zeigt die Suchergebnisse basierend auf einer ausgewählten Region) und Reverse Tinyurl (das Entschlüsseln einer verkürzten URL).
Diese sind teilweise direkt aus der Erweiterung verlinkt, sodass die Ergebnisse auf der Webseite und nicht in der Erweiterung dargestellt werden.
Eine verantwortliche Firma findet sich auf der Webseite nicht. Die Einträge der *whois*-Abfrage der Domain `tcpiputils.com` selbst machen keinen authentischen Eindruck:

```
owner-lname: Hukkel
owner-street: Hukstreet 1
owner-city: Hukcity
```

IPvFox und IPvFoo [19] [116] zeigen im Gegensatz zu den bereits vorgestellten Erweiterungen alle angefragten Domains und zugehörige IP-Adressen im Pop-up. IPvFoo für Chrome zeigt zusätzlich, ob die Verbindungen verschlüsselt sind.

Perspectives [26] visualisiert, überwacht und testet zur Domain zugehörige SSL-Zertifikate. Der Grundgedanke ist eine dezentrale Zertifizierung. Dafür existieren laut dem Perspectives Project [21] mehrere über die Welt verteilte Notar-Server, die Historien von gesehenen SSL-Zertifikaten verwalten.
Wird eine verschlüsselte Webseite besucht, gleicht die Erweiterung das Zertifikat bei den eingetragenen Notar-Servern ab. Ist das Zertifikat nicht geläufig oder selten gesehen, ändert sich das Icon und ein Tooltip weist darauf hin.
Vertraut man den eingetragenen Notar-Servern nicht, kann man einen eigenen Notar-Server auf Ubuntu-Basis in der Amazon-Cloud [6] installieren.
Die Abfragen an die Notar-Server sind durch ein Public-Key-Verfahren ge-

schützt. Abb. 16, S. 65 zeigt die in einem Zeitraum von 30 Tagen gesehenen Zertifikate von `https://www.google.de`.
Perspectives wurde als Forschungsprojekt vom Computer Science Department der Carnegie Mellon University [104] gestartet und ist eine Erweiterung für Mozilla Firefox.

3.2 Blockieren

Bestimmte Elemente (z. B. Bilder, *frames*) ausgewählter Domains werden nicht mehr geladen. Inhalte werden aufgrund ausgewählter Stichwörter (Kinderschutz) nicht mehr angezeigt. Blockierte unerwünschte Inhalte schaffen mehr Sicherheit, weil das Risiko, Schadcode zu empfangen, geringer ist. Ein erwünschter Nebeneffekt ist, dass sich die Geschwindigkeit des Seitenaufbaus steigern kann, weil Daten nicht geladen und Scripte nicht ausgeführt werden müssen.

Personal Blocklist [54] blockiert das Laden von Ressourcen auf Basis von Domain-Listen. Die Erweiterung ist sowohl von der Funktionalität, als auch von der Oberfläche sehr übersichtlich gehalten. Bei Google-Suchen werden Links zum Blockieren der jeweiligen Domain direkt in die Suchergebnisse eingebettet. Domains, die gerade besucht werden, können mit zwei Klicks über das Pop-up blockiert werden. Ebenfalls ist es möglich, eine Liste von Domains gleichzeitig zu sperren.
Die Erweiterung ist von Google für Chrome entwickelt worden; blockierte Einträge werden an Google übertragen. Ähnliche Erweiterungen, z. B. BlockSite [31] gibt es auch für andere Browser.

JavaScript Blacklist [65] ist eine sehr einfache und übersichtliche Erweiterung, die ausschließlich zum Blockieren von JavaScript dient.

NoScript [40] (siehe Abb. 17, S. 66) kann das Laden von JavaScript ebenfalls verhindern. Darüber hinaus ist es aber auch möglich, nur die Ausführung von JavaScript zu blockieren. Per Standardeinstellung werden zunächst alle Scripte, mit Ausnahme einer mitgelieferten Whitelist, nicht ausgeführt. Für die so blockierten Scripte gibt es die Möglichkeiten, sie als nicht, temporär oder langfristig als vertrauensvoll einzustufen. Entsprechend werden sie gar nicht mehr, einmalig oder immer ausgeführt.
Die Erweiterung bietet kein absolutes, aber ein zusätzlichen Maß an Sicherheit vor Cross-Site-Scripting-Angriffen, erhöht dafür aber auch den nötigen Aufwand, weil die Ausführung jedes unbekannten Scripts manuell geprüft werden muss.
NoScript kann neben JavaScript auch Java, Flash und andere Plug-ins blockieren und ist eine Erweiterung für Mozilla Firefox.

Adblock [73] ist die, nach eigenen Angaben, populärste Erweiterung für Google Chrome mit über sechs Millionen Benutzern. Sie dient dazu, Werbung zu blockieren. Die Erweiterung lässt sich mit verschiedenen Blacklists (z. B. EasyList [27]) konfigurieren. Die Einträge sind Reguläre Ausdrücke, auf Basis derer URLs blockiert werden. Die Ausdrücke reichen von GET-Parametern bis zu kompletten URLs.
Ab Version 2.0 wird die Werbung gar nicht mehr geladen, statt nur nicht angezeigt. Adblock gibt es neben Chrome für Safari [13]. Auch für Firefox, Internet Explorer und Opera gibt es diverse Erweiterungen, um Werbung zu blockieren.

Facebook Disconnect und Facebook Blocker [23] [120] blockieren die komplette Kommunikation mit Facebook (mit den Domains: facebook.com, facebook.net, fbcdn.net). „Like-Buttons“ (siehe Abb. 3) werden so nicht mehr geladen. Wird allerdings die Facebook [34] Seite selbst aufgerufen, wird die Kommunikation nicht blockiert. Sowohl Facebook Disconnect als auch Facebook Blocker bieten keine Möglichkeit zur Konfiguration.

Abbildung 3: Facebook like-Button (Quelle: siehe Anhang, S. 62)

IBA Opt-out und Keep My Opt-Outs [50] [51] blockieren ausgewählte Cookies. Beides sind Erweiterungen von Google, die es erlauben, personalisierte Werbung an- oder abzuschalten. Technisch wird der Cookie, der für die Wiedererkennung des Nutzers nötig ist, nicht mehr gesetzt. Die Konfiguration geschieht online und nicht wie üblich in den zugehörigen Optionen. Keep My Opt-Outs funktioniert im Gegensatz zu IBA Opt-out auch für andere Werbenetzwerke als für Google.

3.3 Säubern

In diesem Abschnitt befinden sich Erweiterungen, die das Löschen von Cookies, *localStorage*-Einträgen und der Browser-Historie ermöglichen. Das Löschen von Einträgen, die dem Überwachen des Surfverhaltens dienen, schafft mehr Privatsphäre, da der Nutzer Drittanbietern gegenüber weniger transparent ist. Das Löschen der Browser-Historie schützt die Privatsphäre vor Dritten, die lokal Zugriff auf den Rechner haben. Sie ist nicht über JavaScript zugänglich und kann demnach auch nicht ausgelesen werden. Das *history*-Objekt unterstützt lediglich die Navigation (vor und zurück, auch mehrmals) ohne die Möglichkeit, URLs auszulesen.

Vanilla Cookie Manager [18] (siehe Abb. 18, S. 66) erlaubt das Verwalten und Löschen von Cookies unter Chrome. Dafür liefert die Erweiterung eine entsprechende Liste von Domains, deren Cookies potentiell „unerwünscht“ sind (z. B. weil sie dazu dienen, das Nutzerverhalten zu überwachen). In den Optionen ist es möglich, Domains auf eine Whitelist zu setzen. Das Pop-up bietet die Möglichkeit, die komplette aktuelle Domain oder alle Domains unterhalb der aktuellen Second Level Domain (*.sld.tld) auf die Whitelist zu setzen. Mit einem Klick können alle „unerwünschten“ Cookies gelöscht werden. Die Erweiterung kann so konfiguriert werden, dass dies beim Starten des Browsers automatisch geschieht.
Google Chrome (Vers. 19.0.1041.0 dev) selbst bietet allerdings die Möglichkeit, in den Optionen Cookies von Drittanbietern zu blockieren. Diese Einstellung ist standardmäßig deaktiviert. Um die Einstellung zu ändern, müssen die folgenden Schritte vollzogen werden.

```
Chrome -> Settings -> Advanced Settings -> Content
   Settings -> Block third-party cookies and site data
```

An dieser Stelle kann ebenfalls eingestellt werden, Cookies von Drittanbietern bei jedem Beenden von Chrome zu entfernen. Flash-Cookies werden durch den Vanilla Cookie Manager nicht entfernt.
Um Flash-Cookies bei Mozilla Firefox zu löschen, kann die **BetterPrivacy**-Erweiterung [87] installiert werden. Sie listet alle gespeicherten Flash-Cookies auf und bietet die Möglichkeit, diese automatisch beim Start, beim Beenden oder in einem ausgewählten Zeitintervall zu löschen. Bei Google Chrome ist das nicht-selektive Löschen von Flash-Cookies über den Punkt „Delete Cookies and other site and plug-in data“ im „Clear browsing Data“-Dialog möglich.

Forget Me [37] bietet die Möglichkeit, den *localStorage* der aktuellen Webseite zu löschen. Forget Me ist eine Erweiterung für Chrome, die mit einem Klick auf das Erweiterungs-Icon alle Cookies, den localStorage und den letzten Eintrag in der Browserhistorie der aktuellen Domain löscht. Optional ist es auch möglich, die Domain komplett aus der Browserhistorie zu löschen.

3.4 Tracking

Dabei handelt es sich um das Visualisieren von Drittanbietern, die das Surfverhalten des Nutzers überwachen können. Die Visualisierung allein schafft Transparenz, zusätzliches Blockieren schützt die Privatsphäre.

Collusion [115] visualisiert Verbindungen zu Drittanbietern in einem Graph. Alle Kommunikationspartner werden als Knoten dargestellt. Bis der Browser geschlossen oder der Graph manuell zurücksetzt wird, erweitert Collusion den bestehenden Graph. Bekannte Tracking-Seiten werden mit einem

roten Rand markiert und blockiert. Die Visualisierung sieht optisch ansprechend aus, wird mit vielen Aufrufen zu unterschiedlichen Seiten aber zunehmend unübersichtlicher, siehe Abb. 19, S. 67. Collusion bietet keine Einstellungsmöglichkeiten.

Do Not Track Plus [1] ist eine Erweiterung der Firma Abine, Inc. [2], deren Produkte (DoNotTrackPlus, DeleteMe, PrivacyWatch, GoogleSharing, Privacy Suite) ausschließlich im Bereich Privatsphäre angesiedelt sind. Auf der Webseite werden diesbezüglich viele Informationen angeboten. So wird z. B. anschaulich erklärt, wie Tracking funktioniert.
Die Erweiterung (siehe Abb. 20, S. 68) ist sehr übersichtlich und verständlich gehalten. Blockierte Tracking-Elemente/Firmen werden übersichtlich kategorisiert nach Sozialen Netzwerken, Werbenetzwerken und Firmen dargestellt. Als Grundlage dient eine Liste mit Tracking-Firmen und -Netzen. Auf der Webseite kann man sich über die Firmen gezielter informieren. Eine Konfigurationsmöglichkeit gibt es nicht.

Ghostery [33] (siehe Abb. 21, S. 69), eine Erweiterung der Firma Evidon, Inc. [32], durchsucht die Seite und informiert den Benutzer über gefundene Scripte, Zählpixel und weitere Elemente von Drittanbietern und bietet die Möglichkeit, ausgewählte Firmen zu blockieren. Auf ihrer Webseite bietet Evidon, Inc. kurze Beschreibungen über die ermittelten Firmen und die Kontaktdaten an und zeigt, welche Daten (z. B. Browserinformationen, Anzahl der Seitenaufrufe) gespeichert werden. Zudem kann man sich informieren, welche weiteren Tracking-Netze von den Firmen betrieben werden und auf welchen Seiten diese zu finden sind.
Ghostery gibt es für die fünf am weitesten verbreiteten Browser, funktioniert für den Internet Explorer allerdings nur mit Einschränkungen [33].

SaferChrome [14] (siehe Abb. 22, S. 70) ist eine Erweiterung von Celine [15] & Elie Bursztein [29] für Chrome. Die Erweiterung zeigt potentiell überwachende Drittanbieter, den aktuellen Status der Verschlüsselung und Dateistatistiken für die aktuelle Seite. Es ist möglich, sich über gewünschte Sicherheitskategorien (Alarme, Warnungen, Privatsphäre, Informationen) in Form eines Pop-ups, das in die Seite eingebettet wird, informieren zu lassen. Die Erweiterung warnt, wenn Passwörter oder Kreditkartendaten unverschlüsselt übertragen werden.

ScriptNo [7] (siehe Abb. 23, S. 70) ist eine Erweiterung für Chrome, die ebenfalls überwachende Drittanbieter anzeigt. Diese können dann auf Black- und Whitelists verteilt werden. In den Optionen können zusätzlich ausgewählte Elemente wie *script*, *object*, *iframe* und *frame* sowie Bilder blockiert werden. ScriptNo bietet zudem die Möglichkeit, Cookies von bekannten Werbe- und Malware-Domains nicht zu setzen, Schaltflächen von sozialen

Netzwerken oder unsichtbaren Inhalt von Drittanbietern zu entfernen. Des Weiteren kann der Benutzer die Browserkennung (Hersteller und Versionsnummer), die an den Server gesendet wird, ändern.
Das Icon zeigt die Anzahl, das Pop-up Details über die gefundenen Auffälligkeiten (geblockte Domains, Zählpixel). Weiterhin gelangt man über den „Rating“-Link auf die Web of Trust Webseite.

3.5 Web of Trust

Das **W**eb **o**f **T**rust (WOT) [125] ist eine Bewertungsplattform für Webseiten. Das Hauptprodukt der verantwortlichen Firma, WOT Services Ltd., ist eine Browser-Erweiterung, die die Möglichkeit bietet, Webseiten zu bewerten und berechnete „Durchschnittswerte“ auf Basis der Bewertungen farblich anzuzeigen.
Die Erweiterung existiert für alle großen Browser (Mozilla Firefox, Google Chrome, Internet Explorer, Opera und Safari) und ist kostenlos.
Bewertungen können in vier Kategorien abgegeben werden:

- Vertrauenswürdigkeit
- Zuverlässigkeit von Händlern
- Datenschutz
- Kindertauglichkeit

Die Werte werden auf Basis von Bewertungen und weiteren Quellen, wie Blacklists für schadhafte Seiten, berechnet [125] und als Erweiterungs-Icon und neben den Suchergebnissen vieler Suchmaschinen angezeigt.
Dem Problem des Missbrauchs (z. B. Verbessern des eigenen Rankings oder negatives Bewerten der Konkurrenz) treten die Autoren mit Überwachung von Benutzerverhalten und Algorithmen zur Aufspürung von manipulierten Bewertungen entgegen.

Um den Dienst in eigenen Anwendungen nutzen zu können, stellen die Entwickler eine Programmierschnittstelle (API) bereit [125]. Diese liefert per einfacher GET-Anfrage zu einer angegebenen Domain oder IP-Adresse die gespeicherten WOT Bewertungen. In den Bedingungen muss zugesichert werden, innerhalb eines Zeitfensters eine bestimmte Anzahl von Anfragen nicht zu überschreiten.
Zusätzlich wird empfohlen, WOT in der eigenen Anwendung zu erwähnen, das User-Agent-Feld bei der Anfrage eindeutig identifizierbar zu belegen und dieselbe Information nicht mehrmals innerhalb von 30 Minuten abzufragen. Sollte dies nötig sein, ist ein lokaler Cache zu nutzen. Des Weiteren wird darum gebeten, eine kleine Beschreibung über die eigene Anwendung im WOT Wiki [125] zu hinterlassen.

3.6 Gegenüberstellung dieser Arbeit zu relevanten Erweiterungen

Die, im Zuge dieser Arbeit, entstehende Erweiterung soll, ebenso wie Collusion, die Kommunikation zu Drittanbietern visualisieren. Dabei sollen aber alle Kommunikationspartner angezeigt werden, auch die, die nicht als Tracking-Firmen bekannt sind. So soll z. B. visualisiert werden, dass bei einem Besuch auf `facebook.com` auch Daten von `fbcdn.net` geladen werden.
Einige der vorgestellten Erweiterungen (z. B. Ghostery) zeigen URLs in Webseiten auf, die allein zum Überwachen eingebettet wurden. Dabei werden auch Fundstellen gezeigt, die nicht kommunizieren (weil sie z. B. durch eine andere Erweiterung wie Adblock daran gehindert wurden). Die im Zuge dieser Arbeit entwickelte Erweiterung zeigt nur die tatsächliche Kommunikation, sodass für den Benutzer genau sichtbar ist, welchen Instanzen Daten übermittelt werden.
Das Blockieren von Ressourcen geschieht hier nicht wie bei vielen anderen Erweiterungen aufgrund von mitgelieferten Listen von Domains, sondern aufgrund von Listen von Firmen, die der Benutzer selbst zusammenstellt. Dies ist für den Benutzer übersichtlicher. So kann beispielsweise die Kommunikation mit der *Firma Facebook* unterbunden werden. Bei einer Domain-Sperre müsste der Benutzer wissen, dass die Domain `fbcdn.net` ebenfalls zu Facebook gehört.
Für alle Drittanbieter soll visuell dargestellt werden, ob sie Daten auf dem Client-Rechner speichern (und damit potentiell Nutzerverhalten überwachen können), in welches Land die Kommunikation erfolgt, ob die Verbindung gesichert ist und wie viele Anfragen gemacht werden. Für jede Anfrage soll die Möglichkeit bestehen, den Inhalt anzuzeigen. Zusätzlich soll für jeden Kommunikationspartner das Web of Trust-Ranking angezeigt werden.

Zusätzlich soll es folgende Konfigurationsmöglichkeiten geben: Benachrichtigung bei Kommunikation mit ausgewählten Firmen/in ausgewählte Länder und Benachrichtigung bei Datenspeicherung von Drittanbietern.
Keine der bisher vorgestellten Erweiterungen bietet eine Benachrichtigung bei Kommunikation mit ausgewählten Ländern.
Tabelle 1, S. 26 zeigt eine Gegenüberstellung der hier vorgestellten Erweiterungen.
Eine weitere Funktionalität dieser Arbeit ist das Sammeln von Surfdaten (Angefragte URL, Länderinformationen auf Basis der IP-Adresse und weitere) zur Speicherung und Auswertung auf einem zentralen Server der Freien Universität Berlin. Dabei werden keine persönlichen Daten erhoben (Formularinhalte und Cookies werden nicht übertragen). Alle anfallenden Daten werden so anonym wie möglich gespeichert, sodass daraus kein Nutzerprofil erstellt werden kann.
Auf Basis dieser Daten kann eine weitere Arbeit entstehen, mithilfe derer

Abfragen auf der gesammelten Datenmenge möglich sind.
Das Einschalten der Datenübertragung ist optional und im Auslieferungszustand der Software deaktiviert. Der Benutzer wird über die Art und den Umfang der übertragenen Daten gründlich informiert. Es ist jederzeit möglich, sich ein Beispiel der Daten anzuschauen, die übertragen werden würden.

Funktionalitaet / Erweiterung	WorldIP	Flagfox	Chrome Flags	IP Address and Domain Information	IpvFox und IpvFoo	Perspectives	Personal Blocklist	JavaScript Blacklist	NoScript	Adblock	Facebook Disconnect	Facebook Blocker	IBA / Keep My Opt-Outs	Vanilla Cookie Manager	Forget Me	Collusion	Do Not Track Plus	Ghostery	Safer Chrome	ScriptNo	Web of Trust	Diese Arbeit: Transparency
Visualisierung von SSL-Zertifikaten						✓																
Laenderinformationen fuer ROOT (die angefragte Autoritaet) und/oder Drittanbieter	✓	✓	✓	✓																		✓
Visualisierung ob Cookies gesetzt wurden														✓								✓
Anzeigen aller Kommunikation mit Drittanbietern (ggf. mit Verschluesselung)					✓																	✓
Visualisierung und/oder Blockierung von Tracking durch Drittanbieter											✓	✓	✓			✓	✓	✓	✓	✓		
Moeglichkeit zur Blockierung von Webinhalten							✓	✓	✓	✓							✓	✓		✓	✓	✓
Saeubern von angefallenen Surfspuren														✓	✓							
Anzeigen des Web of Trust Ranking			✓	✓																✓	✓	✓
Hinweis auf die Uebertragung von unverschluesselten Passwoertern																			✓			✓
Benachrichtigung bei ausgewaehlter Kommunikation (Laender, Firmen)																						✓
	Informationsvisualisierung						**Blockieren**							**Saeubern**			**Tracking**			**Benutzerbewertung**		

Tabelle 1: Uebersicht der relevanten Erweiterungen

4 Sammelbare Informationen und Browserunterstützung

In diesem Kapitel wird beschrieben, welche Informationen gesammelt werden können und welche technische Unterstützung der Browser diesbezüglich vorhanden ist.
Auf den ersten Blick unterscheidet sich vor allem der Internet Explorer von den anderen Browsern dadurch, dass Erweiterungen auf Basis von .NET und nicht den nativen Browsertechniken HTML, CSS und JavaScript entwickelt werden. Der Aufbau von Erweiterungen ist, mit Ausnahme des Internet Explorers, sehr ähnlich. Es gibt ein Script, das im Hintergrund abläuft und mit einem Pop-up oder mit in die Webseite injizierten Scripten (Content Scripts) über Nachrichten kommunizieren kann.
Grundvoraussetzung für diese Arbeit ist das Ermitteln der IP-Adresse und der URL für jede Anfrage. Dabei müssen die Requests den unterschiedlichen Tabs zuzuordnen sein. Aus der IP-Adresse können dann weitere Informationen abgeleitet werden.

4.1 IP-Adresse

Bei Opera und Safari findet sich in der API keine Möglichkeit, die IP-Adresse auszulesen. Über ein in die Seite injiziertes JavaScript kann sie ebenfalls nicht ermittelt werden. Für den Internet Explorer wäre der Aufwand für eine Testversion sehr hoch. Laut einem Blogeintrag auf `http://www.blackfishsoftware.com` [25] gibt es aber die Möglichkeit, einen eigenen ***A****synchronous* ***P****luggable* ***P****rotocol (APP) Handler* zu implementieren, den Default-Handler, der für alle HTTP-und HTTPS-Anfragen zuständig ist, zu ersetzen und so alle Anfragen mitzulesen. Das Ermitteln der IP-Adresse geschieht bei Firefox über *Observer Notifications* [81]; es muss auf das „http-on-examine-response“-Event gewartet werden. Man erhält ein *httpChannel*-Objekt, aus dem der *Host-Header* ausgelesen werden kann. Die Übersetzung in eine IP-Adresse geschieht dann mittels DNS. Bei Chrome liefert die Methode *chrome.webRequest.onCompleted.addListener()* das *onCompleted*-Event. Per *callback* erhält man ein Objekt, in dessen Attribut *ip* die IP-Adresse der Anfrage gespeichert ist, siehe Listing 4.

Listing 4: Registrieren des *onCompleted*-Listeners

```
chrome.webRequest.onCompleted.addListener(function(
   details){
    console.log(details.ip);
});
```

Die IP-Adresse kann dazu verwendet werden, das Land, in dem sich der Server befindet, zu ermitteln.

4.2 URL

Sowohl in der API von Opera als auch in der von Safari findet sich keine Methode, die URLs aller Anfragen auszulesen. Bei Opera kann die URL der aufgerufenen Seite ermittelt werden, weitere Anfragen können über die API nicht überwacht werden. Bei Safari sieht es ähnlich aus: findet eine neue Navigation statt, kann über das *SafariBeforeNavigateEvent* die angesurfte URL ausgelesen und die Navigation sogar verhindert werden [10]. Weitere Anfragen sind auch hier über die API nicht zu erfassen. Eine andere Methode, die URLs aller Anfragen in Safari und Opera auszulesen, wäre die Registrierung eines „beforeload"-Event-Listeners in einem Content Script. Bei beiden Browsern ist das Injizieren von Scripten in die Webseite möglich. Entscheidend ist, dass die Scripte vor dem Laden und der Ausführung aller anderen Scripte ausgeführt werden müssen. Diese Methode wurde allerdings nur für Opera getestet, nicht für Safari. Bei Chrome ist die URL ebenso wie die IP-Adresse über das *onCompleted*-Event zugänglich. Die URL ist bei Firefox über ein *nsIURI*-Objekt erreichbar. Dieses ist als Attribut *originalURI* des *httpChannel*-Objekts (s.o.) gespeichert. Das *nsIURI*-Objekt enthält die Attribute *scheme*, *host*, *path* und weitere, aus der die komplette URL zusammengesetzt werden kann. Da bei dem Internet Explorer der Default-Handler, der für alle Anfragen zuständig ist, ersetzt wird, ist es auch hier möglich, die URLs mitzulesen.
Aus der URL können das Schema, die Domain und der Pfad extrahiert werden. Das Schema zeigt an, ob eine Verbindung verschlüsselt ist oder nicht. Die Domain kann anschließend für die Zuordnung einer Firma zu der Anfrage verwendet werden.

4.3 DNS-Auflösung

Eine DNS-Auflösung muss bei Chrome nicht explizit vorgenommen werden. In dem zurückgelieferten Objekt ist die URL enthalten, aus der die Domain extrahiert werden kann. Bei Firefox muss zunächst ein *dnsService*-Objekt erstellt werden, siehe Listing 5.

Listing 5: Bereitstellung der DNS-Funktionalität unter Firefox

```
var dnsService = Components.classes["@mozilla.org/
    network/dns-service;1"].createInstance(Components.
    interfaces.nsIDNSService);
```

Dieses liefert über die Methode *resolve()* ein *nsIDNSRecord*-Objekt, das die Domain enthält [80]. Für den Internet Explorer kann die Methode *GetHostEntry()* der *Dns*-Klasse genutzt werden (C#) [74]. Als Parameter muss der Methode ein *IPAddress*-Objekt übergeben werden, als Rückgabe erhält man ein *IPHostEntry*-Objekt, in dessen Attribut *HostName* die Domain gespeichert ist. Weder in der API von Opera noch bei Safari gibt es die Möglichkeit von DNS-Auflösungen.

Der einzige Browser, der Reverse DNS-Abfragen ausführen kann, ist der Internet Explorer. Die *Dns*-Klasse enthält dafür die Methode *GetHostEntry()*.

4.4 Antwort-Header

Da der Zugriff auf alle Anfragen bei Safari und Opera über die API nicht gegeben ist, können entsprechend auch nicht die Response Header ausgelesen werden. Ein Zugriff über das „beforeload“-Event scheint ebenfalls nicht möglich. Die drei anderen Browser unterstützen das Auslesen der Response Header. Chrome bietet, wenn der Erweiterung entsprechende Rechte gewährt wurden, das Objekt *responseHeaders*. Dies ist ein Array von HTTP-Headern, die jeweils aus einem Dictionary mit den Schlüsseln *name* und *value* bestehen. Firefox stellt das Auslesen der Response Header über die Methode *getResponseHeader()* der *nsIHttpChannel*-Klasse bereit. Header können nicht nur ausgelesen, sondern an dieser Stelle auch über *setResponseHeader()* verändert werden. Im Internet Explorer können über das *IHttpNegotiate*-Interface und die Methode *szResponseHeaders()* die Response Header ausgelesen werden [75].
Antwort-Header liefern diverse Informationen, wie das Datum, um welchen Webserver es sich handelt (z. B. Apache) oder den Typ der übertragenen Daten. Im Header sind auch Cookies enthalten, die bei dem Client gespeichert werden.

4.5 Antwortzeiten

Die Antwortzeiten sind bei Chrome leicht zu ermitteln. Eine Anfrage durchläuft mehrere Schritte, die über Listener überwacht werden können. Alle Ereignisse enthalten einen Zeitstempel. Es muss also lediglich die Differenz der Zeitstempel des *onBeforeRequest*-Events und des *onResponseStarted*-Events genommen werden. Das *onResponseStarted*-Event tritt auf, sobald das erste Byte der Antwort eingetroffen ist [45]. Bei Firefox ist die Funktionsweise gleich, bis auf den Unterschied, dass der Zeitstempel selbst genommen werden muss. Es ist möglich, über *Observer Notifications* [84] die Ereignisse „http-on-modify-request“ und „http-on-examine-response“ zu überwachen. Die Funktionsweise für den Internet Explorer ist ebenfalls dieselbe. Hier wird wieder das *IHttpNegotiate*-Interface genutzt. Die Methoden *BeginningTransaction()* und *OnResponse()* benachrichtigen über Beginn und Antwort einer Anfrage.

4.6 Externe Abfragen

Die Möglichkeit, externe Abfragen zu stellen, bieten alle Browser. Bei Chrome, Opera und Safari ist die Vorgehensweise identisch. In JavaScript wird ein *XMLHttpRequest*-Objekt erstellt, über das eine externe Anfrage durchgeführt werden kann (siehe Listing 6).

Listing 6: Ausführung einer asynchronen GET-Anfrage in JavaScript

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "http://foobar.de?foo=bar", true);
xhr.onreadystatechange = function(){
    if (xhr.readyState == 4){
        //do something
    }
}
xhr.send();
```

Wichtig dabei ist, dass die Abfrage in einer dafür vorgesehenen Umgebung (in der Regel die Background Page, siehe S. 37) durchgeführt wird. In einem in die Webseite eingefügten Script sind aufgrund der Same-Origin-Policy nur Anfragen an dieselbe Domain zugelassen. Die Unterstützung des *XMLHttpRequest*-Objekts findet sich in der Dokumentation von Chrome und Opera. Für die Unterstützung bei Safari existiert lediglich ein Beitrag von „Flambino“ auf `stackoverflow.com` [36]. In der offiziellen Dokumentation [9] findet sich kein Eintrag diesbezüglich. Firefox besitzt eine eigene API (*nsIXMLHttpRequest* [83]) zur Ausführung von externen Anfragen. Das folgende Codebeispiel (Listing 7) zeigt das Übermitteln von POST-Daten.

Listing 7: Ausführung einer asynchronen Anfrage aus einer Firefox-Erweiterung

```
var req = Components.classes["@mozilla.org/xmlextras/
    xmlhttprequest;1"].createInstance();
 req.open('POST', "http://www.domainnamehere.de/path",
      true);
 req.send('foo=bar');
```

Im Internet Explorer liegt die Funktionalität in den *WebRequest*- und *WebResponse*-Klassen. Eine Abfrage für den Internet Explorer ist in Listing 8 dargestellt.

Listing 8: Ausführung einer externen Anfrage in C#

```
WebRequest request = WebRequest.Create("http://www.
    foobar.de/index.html");
HttpWebResponse response = (HttpWebResponse)request.
    GetResponse();
```

Externe Requests bieten die Möglichkeit, gesammelte Daten durch weitere Informationen aus zusätzlichen Quellen anzureichern. So kann beispielsweise das WOT-Ranking abgefragt werden.

4.7 Externe Befehle

Eine weitere Möglichkeit der Informationsanreicherung besteht in der Ausführung externer Befehle. Diese Möglichkeit kann allerdings auch missbraucht werden. So ist es denkbar, dass durch einen Befehl vertrauliche Daten auf der Festplatte ausgespäht und über die Erweiterung, per asynchronem Request, an Unbefugte versendet werden.
Nach Abschluss der Recherchen scheint die Funktionalität nur bei Chrome und dem Internet Explorer in vollem Umfang zur Verfügung zu stehen. Chrome benutzt das **N**etscape **P**lugin **A**pplication **P**rogramming **I**nterface (NPAPI) [79] zur Ausführung externer Befehle. NPAPI-Module zählen allerdings nicht als Erweiterung, sondern als eigenständige Plug-ins (zur Begriffserläuterung siehe Abschnitt 2.6, S. 16). Damit die Erweiterung auf das Plug-in zugreifen kann, muss dieses zunächst in der Konfigurationsdatei (manifest.json) eingetragen werden. Anschließend wird das Plug-in in die (virtuelle) Seite der Erweiterung eingebettet. Der Zugriff auf eine Methode geschieht wie im Folgenden dargestellt.

Listing 9: Ausführung einer Methode eines Plug-ins innerhalb einer Erweiterung

```
<embed type="application/x-my-extension" id="pid">
<script>
  var plugin = document.getElementById("pid");
  var result = plugin.method_call();
  console.log(result);
</script>
```

In der Dokumentation wird aber darauf hingewiesen, dass die Benutzung der API nicht unproblematisch ist.

Warning
NPAPI is a really big hammer that should only be used when no other approach will work. [53]

Auf den bereits erwähnten Gegenstand des Missbrauchs wird auch in der Dokumentation hingewiesen. Plug-ins werden nicht in einer Sandbox ausgeführt, sondern mit den vollen Rechten des aktuellen Benutzers. Aufgrund dieser Sicherheitsproblematik werden Erweiterungen, die auf Plug-ins zugreifen und veröffentlicht werden sollen, manuell überprüft [53].
Um externe Befehle bei dem Internet Explorer auszuführen, wird die *Process*-Klasse [76] verwendet. Das folgende Beispiel (Listing 10) zeigt die Funktionsweise.

Listing 10: Ausführung eines Befehls aus einem C# Programm

```
Process myProcess = new Process();

try{
    myProcess.StartInfo.UseShellExecute = false;
    myProcess.StartInfo.FileName = "path/to/process.
        exe";
    myProcess.Start();
}catch(Exception e){
    //exception handling
}
```

Ausgaben des ausgeführten Programms können über das Attribut *Standard-Output* ausgelesen werden.
Bei Firefox existiert zwar über die *nsIProcess* API [82] auch die Möglichkeit, externe Befehle auszuführen, allerdings wird nur ein Return Code, der anzeigt, ob die Ausführung erfolgreich war oder nicht, zurückgegeben. Beispielsweise können *whois*-Informationen damit nicht ausgelesen werden.

4.8 SSL-Zertifikate

Um einfache Verschlüsselungsinformationen auszulesen, bietet Firefox das Attribut *securityInfo* der *nsIChannel*-Klasse. Diese ist eine Oberklasse der bereits erwähnten *nsIHttpChannel*-Klasse. Über das Attribut ist ein Objekt vom Typ *nsITransportSecurityInfo* zugänglich. Dieses beinhaltet die Attribute *securityState*, das anzeigt, ob die Verbindung verschlüsselt ist, und *shortSecurityDescription*, das den Namen der Zertifizierungsstelle enthält. Um weitere Informationen bezüglich der Verschlüsselung zu erhalten, ist es möglich, eine zusätzliche asynchrone Abfrage zu stellen und innerhalb dieser den Namen, den Aussteller, den virtuellen Fingerabdruck und Validitätsdaten auszulesen. Ein ausführliches Beispiel findet sich auf den Seiten des Mozilla Developer Network [85]. Für den Internet Explorer bietet C# in *System.Security.Cryptography.X509Certificates* einige Methoden, um Zertifikate zu verarbeiten [77]. Ein komplettes Beispiel, eine Anfrage auszuführen und dabei die Zertifikatinformationen auszulesen, findet sich auf `stackoverflow.com` [127]. Dabei wird ebenfalls eine zusätzliche Anfrage gestellt. Für die anderen drei Browser konnte keine Unterstützung zur Visualisierung von Zertifikaten gefunden werden.

4.9 Tabellarische Übersicht

In der folgenden Tabelle (Tabelle 2, S. 34) sind die genannten Informationen zusammengefasst. Dabei bedeutet ein Haken, dass die Funktionalität getestet wurde oder sich entsprechende Einträge in der API befinden. Für Einträge, die in der Tabelle mit einem X markiert sind, konnte keine Unterstützung gefunden werden. Ein Fragezeichen bedeutet, dass

die Funktionalität im Ansatz gefunden (z. B. als Eintrag in einem Forum aber nicht in der offiziellen API) und/oder nicht ausreichend getestet wurde. Erläuterungen zu diesen Einträgen folgen im einzelnen.

1: Lokale Ausführung nicht getestet oder nicht möglich. Für Abfragen über Webschnittstellen ist eine externe Quelle nötig.
2: Lokale Ausführung nicht getestet. Die Abfrage über eine Webschnittstelle ist nicht sinnvoll, da Ergebnisse aufgrund unterschiedlicher Standorte verschieden sein können.
3: Externe Webschnittstelle nötig, da auf die Ausgabe bei lokaler Ausführung nicht zugegriffen werden kann.
4: Eintrag im Forum gefunden [25].
5: Nicht überprüft.
6: Lokale Ausführung laut API möglich, die Befehle whois und traceroute müssen vorhanden sein.
7: Keine Möglichkeit in der API vorhanden. Auslesen über das „beforeload"-Event im Content Script nicht getestet.
8: Siehe Punkt 7. Das Mitlesen der Requests ist Voraussetzung.
9: Funktionalität in der API nicht beschrieben, laut einem Eintrag bei `stackoverflow.com` *aber möglich [36].*
10: Siehe Punkt 9. Das Ausführen externer Anfragen ist Voraussetzung.

Um *whois* oder auch *traceroute* Befehle ausführen zu können, müssen diese beim Betriebssystem vorhanden oder nachträglich installiert worden sein. Die Einträge IP-Adresse (Tab bezogen), URL (Tab bezogen), Response Header, Antwortzeiten URL: Schema, Domain, Pfad, Header: Content/Mime-Type, Header: P3P, X-Frame-Options, Cookies, Herkunftsland und Unternehmen beziehen sich auf alle Anfragen bzw. unterschiedliche Domains. Das bedeutet, eine Funktionalität wird als vorhanden angesehen, wenn für alle Anfragen beispielsweise die Response Header ausgelesen, oder für alle angefragten Domains vorhandene Cookies angezeigt werden können.

	Google Chrome	Mozilla Firefox	Internet Explorer	Opera	Safari
Version	21.0.1180.89	15.0.1	8.0.6001.18691	12.02	5.1.7
Technische Funktionalität					
IP-Adresse (Tab bezogen)	√	√	?[4]	X	X
URL (Tab bezogen)	√	√	?[4]	√	?[7]
DNS-Auflösung	X	√	√	X	X
Reverse DNS	X	X	√	X	X
Response Header	√	√	√	X	X
Antwortzeiten	√	√	?[5]	?[5]	?[8]
Externe Requests	√	√	√	√	?[9]
Externe Befehle	√	(X)	√	X	X
SSL-Zertifikat-Informationen	X	√	√	X	X
Darstellbare Informationen					
URL: Schema, Domain, Pfad	√	√	√	√	?[8]
Header: Content/Mime-Type	√	√	√	X	X
Header: P3P, X-Frame-Options	√	√	√	X	X
Cookies	√	√	√	X	X
Anzahl verschiedener Domains	√	√	√	√	?[8]
WOT-Ranking*	√	√	√	√	?[10]
whois-Abfrage*	?[1]	?[3]	?[6]	?[1]	?[1]
traceroute-Abfrage*	?[2]	X	?[6]	X	X
Herkunftsland*	√	√	√	√	X
Unternehmen*	√	√	√	√	?[10]
Zusätzliche Informationen					
Betriebssystem	X	X		X	X
Technologien und Visualisierung	HTML,CSS,JS	HTML,CSS,JS	.NET(C#,C++,VB)	HTML,CSS,JS	HTML,CSS,JS
API-Bewertung (subjektiv)	+	−	−−	++	+
Anzahl der Erweiterungen	∼10000 [8]	∼10000 [8]	795 [78]	∼1000 [95]	∼300 [11]

√ Funktionalität sichergestellt, ? Ansätze gefunden, Durchführbarkeit nicht getestet, X Nicht unterstützt, * Externe API/Funktionalität notwendig

Tabelle 2: Browserübersicht

5 Systementwicklung

5.1 Ziele

Die Hauptziele der Systementwicklung sind die Darstellung von Authorities (Bündelung von Anfragen, die an dieselbe Domain gehen), verschlüsselten Verbindungen, gesetzten Cookies und Länderinformationen. Zudem soll die Software leicht verständlich, erlernbar und bedienbar sein. Zusätzlich soll die Möglichkeit bestehen, die Software zu konfigurieren. Dazu gehört das Blockieren von ausgewählter Kommunikation, das Benachrichtigen über Passwörter, die unverschlüsselt übertragen werden, und die Benachrichtigung über eine Kommunikation mit ausgewählten Firmen und/oder Ländern.
Um die Software verständlich zu halten, werden nicht alle anzeigbaren Informationen visualisiert. Dadurch liegt der Fokus mehr auf den gezeigten Informationen. Technische Informationen, wie beispielsweise der Server-Header sind für Benutzer ohne technischen Hintergrund nicht relevant.

5.2 Vergleich und Wahl des Systems

5.2.1 Entscheidung für ein System

Für Opera und Safari gibt es nicht die Möglichkeit, die IP-Adressen für jede Anfrage zu ermitteln, was aber die Grundvoraussetzung ist. Deshalb scheiden beide Browser für die Entwicklung aus.
Die Marktanteile der übrigen Browser unterscheiden sich bei unterschiedlichen Quellen zu stark, als dass daraus ein Favorit ermittelt werden könnte. Laut *StatCounter* [107] hat Google Chrome den Internet Explorer auf der Web-Browser-Rangliste auf Platz eins abgelöst, *Net Applications* hingegen sieht den Internet Explorer mit 54% vorne [88].
Ein anderes Maß ist die Anzahl der bereits existierenden Erweiterungen. Hier haben Chrome und Firefox die meisten Browser-Erweiterungen (siehe Tabelle 2, S. 34). Ebenfalls sind beide als Open Source verfügbar (Chrome in der Open Source Variante Chromium), dies bietet zusätzliche Transparenz. Dass der Internet Explorer nicht als Open Source zur Verfügung steht, ist ein Argument gegen ihn. Zudem ist er unter Mac OS und Linux/Unix nicht einsetzbar, was einen möglichen Nutzerkreis einschränkt

Sowohl Mozilla Firefox als auch Google Chrome verfügen ungefähr über den selben Funktionsumfang, beide sind als Open Source verfügbar und auf unterschiedlichen Plattformen einsetzbar. Die API von Chrome ist jedoch übersichtlicher und besser dokumentiert. Deshalb wir die Erweiterung für Google Chrome entwickelt.

5.2.2 System-spezifische Eigenschaften

Die Software kann im Chrome Web Store [49] veröffentlicht werden. Dort ist sie für jeden über das Internet herunterladbar.
Eine einfache Installation ist durch das System, das Google zur Programmierung und Veröffentlichung von Erweiterungen bereitstellt, gewährleistet. Erweiterungen müssen, egal aus wie vielen einzelnen Dateien sie bestehen, in einer .crx Datei *gepackt* sein. Diese wird mit einem Schlüssel, der zu Beginn der Entwicklung ausgestellt wird, signiert. Damit ist gewährleistet, dass die Erweiterung nicht fälschbar ist. Durch die eindeutige Erweiterungs-ID, die ebenfalls zu Beginn der Entwicklung vergeben wird, kann eine Erweiterung leicht aktualisiert werden. Eine vorhandene ältere Version wird dabei überschrieben. Erweiterungen für Google Chrome sind sofort nach der Installation ohne einen Neustart des Browsers nutzbar.

5.3 Softwarearchitektur

Der Aufbau der Software ähnelt einer **M**odel-**V**iew-**C**ontroller-Architektur (MVC) [55]. Dabei bildet die Funktionalität in der Datei background.js das Model. Informationen (auftretende Requests) werden dort gesammelt, angereichert und in dem Array *tabs* gespeichert. Die Visualisierung geschieht in der Datei popup/script.js. Diese bildet die View-Komponente, in der aus den gespeicherten Informationen das HTML-Dokument generiert wird. Der Controller besteht nur aus einer einzigen Funktion, die die *tabId* des aktiven Tabs ausliest und die entsprechenden Informationen aus dem Model an das Pop-up weiterreicht. Eine Übersicht ist in Abb. 4 dargestellt.

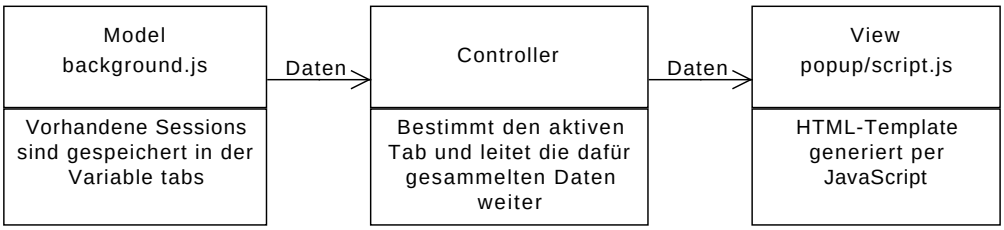

Abbildung 4: Model-View-Controller-Architektur

Eine gewisse Struktur des Projektes ist bereits durch Google vorgegeben. In der Dokumentation werden die Module, wie im Folgenden beschrieben, eingeteilt.

Die Manifest-Datei

Für jedes Projekt existiert eine Manifest-Datei (manifest.json), in der die Erweiterung beschrieben werden kann, eine Versionsnummer und der Autor vermerkt sind, welche Zugriffsrechte (z. B. auf gespeicherte Cookies) die Erweiterung benötigt und welche weiteren Dateien eingebunden werden sollen. Der Inhalt der Manifest-Datei für meine Erweiterung ist in Abb. 14, S. 63 dargestellt.
Über den Schlüssel „browser_action“ kann angeben werden, welche Aktion stattfindet, wenn das Icon der Erweiterung angeklickt wird. In diesem Fall öffnet sich die Datei *popup/index.html* in einem Pop-up.
Alle Scripte, die dauerhaft im Hintergrund aktiv sein sollen, werden unter „background“ notiert. Die Ordnung entspricht dabei der Reihenfolge der Ausführung.
Unter „content_scripts“ sind alle Scripte aufgelistet, die, ebenfalls in der notierten Reihenfolge, in die Webseite injiziert werden. Dabei können die Seiten durch Reguläre Ausdrücke von URLs selektiert werden. Drei mögliche Einstellungen, den Zeitpunkt der Ausführung festzulegen, stehen dabei zur Verfügung: „document_start“, „document_end“ und „document_idle“.
Die Zugriffsrechte (z. B. Auslesen vorhandener Cookies), die die Erweiterung benötigt, regelt „permissions“. Sie werden dem Benutzer vor der Installation der Erweiterung gezeigt, sodass er aufgrund dieser Informationen entscheiden kann, ob er die Erweiterung installieren möchte oder nicht.
Mit Manifest Version 2 wurde das Element „content_security_policy“ hinzugefügt, das bei der Einstellung „script - src self; object - src self“ das Laden externer Scripte verhindert. Alle Bibliotheken (z. B. jQuery), die die Erweiterung benutzt, dürfen nicht mehr von externen Quellen geladen, sondern müssen mit ihr ausgeliefert werden. Dadurch entsteht weniger Angriffspotential durch „Man-in-the-middle“-Angriffe. Zudem wurde die Ausführung von Inline-JavaScript (Funktionen im HTML-Code an den Elementen oder Funktionalität innerhalb von <script>-Tags) unterbunden. Damit bleiben viele Cross-Site-Scripting-Angriffe wirkungslos.

Background Page

In der Background Page (siehe Abb. 5) befindet sich in der Regel der größte Teil der Logik der Erweiterung. Dort hat man Zugriff auf die in der Chrome Extension API [43] dokumentierte Funktionalität - entsprechende Zugriffsrechte in der Manifest-Datei vorausgesetzt. Es ist möglich, Lesezeichen auszulesen, zu bearbeiten und neue anzulegen, auf neue Tabs zu reagieren, Tabs zu öffnen und zu schließen. Über die *storage*-API können Daten (z. B. Einstellungen der Erweiterung) bei Google gespeichert werden. Eine Möglichkeit, die am häufigsten besuchten URLs auszulesen, gibt es auch. Darüber hinaus existieren diverse weitere Funktionen, insgesamt befinden

sich derzeit (24.08.2012) 34 Einträge in der API. Die für diese Arbeit relevanteste Funktion, Requests mitzulesen, wird in Abschnitt 5.6.1 genauer beschrieben. Sofern die Erweiterung nicht deaktiviert ist, ist genau eine Instanz der Background Page für alle geöffneten Tabs und Fenster des Chrome Browsers aktiv.

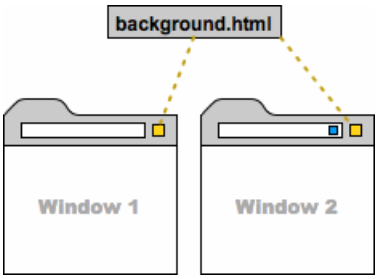

Abbildung 5: Background Page (Quelle: siehe Anhang, S. 62)

UI Pages

Unter UI Pages (siehe Abb. 6) werden alle Seiten zusammengefasst, mit denen der Benutzer interagieren kann. Dazu zählen das Pop-up, eine oder mehrere Konfigurationsseiten (Options Pages) und beliebig viele weitere individuelle Seiten (z. B. Danksagungen, Referenzen, Lizenzen).

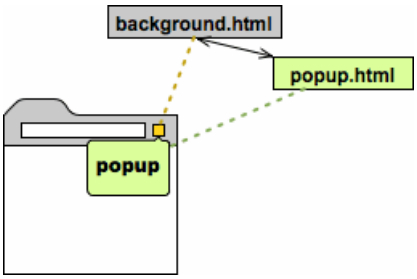

Abbildung 6: User Interface Pages (Quelle: siehe Anhang, S. 62)

Content Scripts

Content Scripts (siehe Abb. 7) werden in die Webseite eingebettet und dort wie ein mitgeliefertes Script ausgeführt. Über Content Scripts ist es möglich, das DOM zu traversieren und zu bearbeiten und damit zusätzliche Funktionalität zur Verfügung zu stellen. Ebenso kann durch ein Content Script auch Funktionalität, die nicht erwünscht ist, eingeschränkt oder deaktiviert werden.

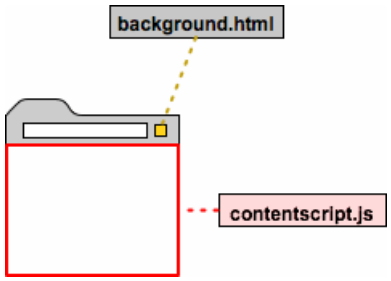

Abbildung 7: Content Scripts (Quelle: siehe Anhang, S. 62)

Kommunikation zwischen den einzelnen Komponenten

Die zuvor genannten Komponenten können über *Message Passing* [42] miteinander kommunizieren. Die Kommunikation erfolgt dabei asynchron. Listing 11 zeigt, wie eine Nachricht von einem Content Script verschickt und von einem Message-Listener entgegengenommen wird. Dabei muss der Listener registriert und aktiv sein, bevor die Nachricht eintrifft.

Listing 11: Nachrichtenaustausch innerhalb einer Chrome-Erweiterung

```
chrome.extension.sendMessage(MESSAGE, function(
    response) {
     ...
});

chrome.extension.onMessage.addListener(function(
    MESSAGE, sender, sendResponse){
     ...
});
```

Im *callback* des Listeners kann die Kommunikation durch das Aufrufen von *sendResponse* aufrecht erhalten werden.
Alternativ kann in einem Script einer UI Page auf die Background Page (genauer das *window*-Objekt der Background Page) über *chrome.extension.getBackgroundPage()* zugegriffen werden.

5.4 Entwurf

Die gesammelten Daten werden so gespeichert, dass ihre Struktur leicht in die Visualisierung im Pop-up übertragbar ist. Eintreffende *Request*-Objekte werden in mehreren Schritten um zusätzliche Daten wie Cookies, Ländercode und weitere Informationen erweitert. Der Ablauf ist in Abb. 8 dargestellt.
Requests werden Authorities zugeordnet, die in Sessions zusammengefasst sind. Eine Authority hält alle Requests, die dieselbe Domain angefragt haben. Abb. 9 zeigt das zugehörige Klassendiagramm, Abb. 15, S. 64 ein beispielhaftes Objektdiagramm.

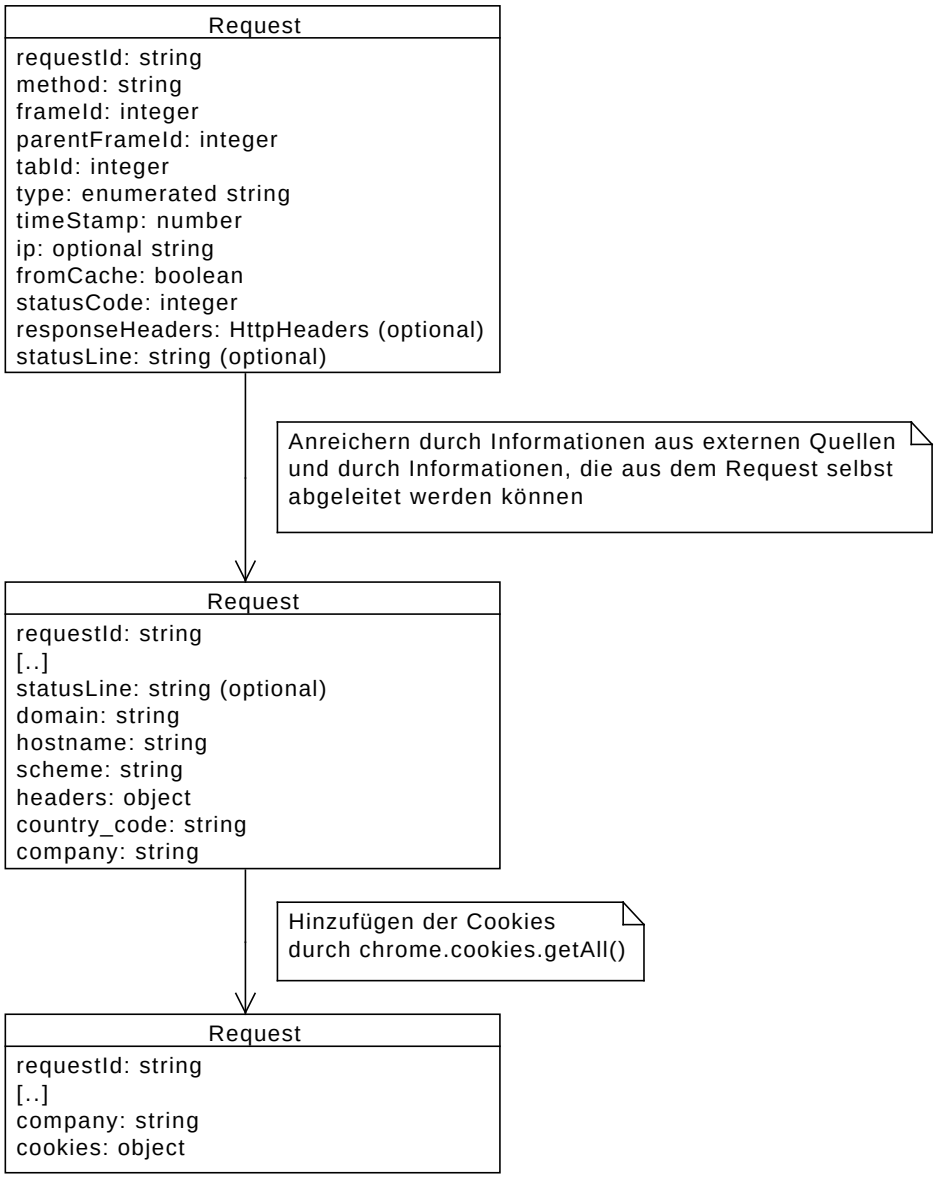

Abbildung 8: Anreicherung des Google-Request-Objekts. *Die Pfeile stellen eine Entwicklung dar und entsprechen keinem gültigen UML-Standard.*

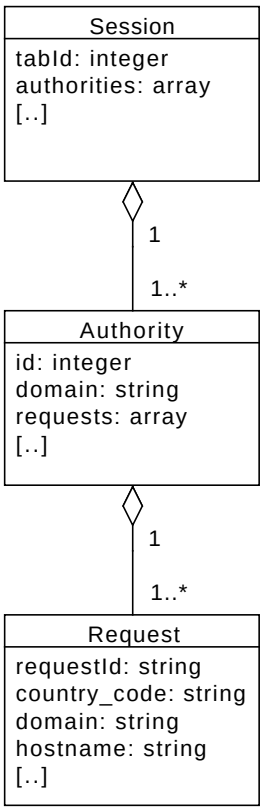

Abbildung 9: Klassendiagramm: Sessions, Authorities und Requests

Weitere Module sind z. B. das WOT-Modul (wot.js), das zu einer gegebenen Domain die WOT-Bewertung abfragt. Die komplette Funktionalität liegt dabei in dem *WOT*-Objekt. Wichtig ist außerhalb des Moduls nur die Funktion WOT.reputation(*domain*). Diese liefert die WOT-Bewertung zu der Domain. Dabei kann die Bewertung, wenn sie bereits einmal abgefragt wurde, aus dem lokalen Speicher gelesen werden. Ist die Bewertung nicht vorhanden, liefert die Funktion *undefined* zurück, fragt aber vorher den WOT-Server an, um die Bewertung für zukünftige Abfragen zu speichern. Anstatt sofort einen Wert zurückzugeben, könnte mit *callbacks* gearbeitet werden. Dabei besteht aber das Risiko, sehr lange auf die Wiederkehr der Funktion zu warten. In der Abfrage ist eine Begrenzung enthalten, sodass dieselbe Domain höchstens einmal in der Minute abgefragt wird.
Das Companies-Modul funktioniert ähnlich wie das WOT-Modul. Soll für eine Domain die Firma ermittelt werden, so wird zunächst lokal nachgeschaut, ob die Firma bereits bekannt ist. Ist sie es nicht, wird eine Anfrage an den FU-Server gestellt, der die Firma ggf. gespeichert hat. Um eine Datenbank

von Zuordnungen zwischen Domains und Firmen aufzubauen, werden eingegebene Firmen an die FU übertragen.
Da so zwei mögliche Quellen von Firmendaten existieren, gibt es zwei lokale Speicher. Sind beide gefüllt (vom Benutzer eingegeben, vom Server geholt), so wird der Wert des Benutzers bevorzugt und zurückgegeben. Ist die Firma bereits auf dem Server gespeichert, so ist die Motivation des Benutzers wahrscheinlich deutlich geringer, selbst einen Namen einzutragen.
Das Daten-Modul (data.js) enthält die zulässigen Top-Level-Domains (.de, .com), bekannte Second-Level-Domains (.co, .gov) und die Zuordnung von Ländercodes zu Ländernamen. Die Datei enthält keine weitere Logik, sondern dient nur als Speicherort für vorgegebene Standardwerte. Mit Hilfe der Domain-Einträge werden die letzten zwei (wenn Second-Level-Domains enthalten sind, drei) Teile der Domain aus der URL extrahiert.

Beispiel: http://my.ebay.co.uk/ws/eBayISAPI.dll wird zu ebay.co.uk

5.5 Visualisierung der sammelbaren Informationen

Der grafische Teil der Erweiterung kann in einem Pop-up, einem neuen Tab oder auch direkt in die Webseite über injizierte Scripte eingebunden werden. Werden die Ergebnisse in einem neuen Tab dargestellt, so müsste gekennzeichnet werden, zu welchem Aufruf die dargestellten Informationen gehören. Eine Einbettung in die aktuelle Webseite verändert diese, sowohl grafisch als auch das DOM. Dies kann im Zusammenspiel mit anderen Erweiterungen zu unerwünschten Nebeneffekten führen, da diese eingefügten Code nicht gesondert betrachten können. Deshalb werden die gesammelten Informationen in einem Pop-up angezeigt. Ein Tabwechsel ist während eines geöffneten Pop-ups nicht möglich, das Pop-up schließt sich vorher automatisch. Damit ist gewährleistet, dass ein geöffnetes Pop-up auch immer die Informationen der aktuellen Webseite zeigt.
Die Informationen sollen übersichtlich in der Erweiterung dargestellt werden. Es ist in vielen Fällen möglich, auf weitere Anbieter zu verweisen, dabei werden Benutzer aber auf eine externe Internetseiten, auf die kein Einfluss besteht, weitergeleitet. Hier können dem Benutzer falsche Informationen oder auch beispielsweise viel Werbung gezeigt werden. Beides ist nicht erwünscht. Deshalb werden alle Informationen innerhalb der Erweiterung gezeigt und es wird auf externe Verweise verzichtet.
In der Übersicht werden alle Informationen (z. B. Firma, Verschlüsselung, Land, Anzahl der Anfragen) bezüglich einer Domain gebündelt dargestellt. Die Informationen der Domain, zu der navigiert wurde, wird durch eine besondere Position (zentralisiert oder über allen anderen) hervorgehoben.
Im Folgenden wird beschrieben, welche und wie die in Kapitel 4 gesammelten Informationen visualisiert werden. Abbildung 10 zeigt zunächst einen Screenshot dieser Arbeit.

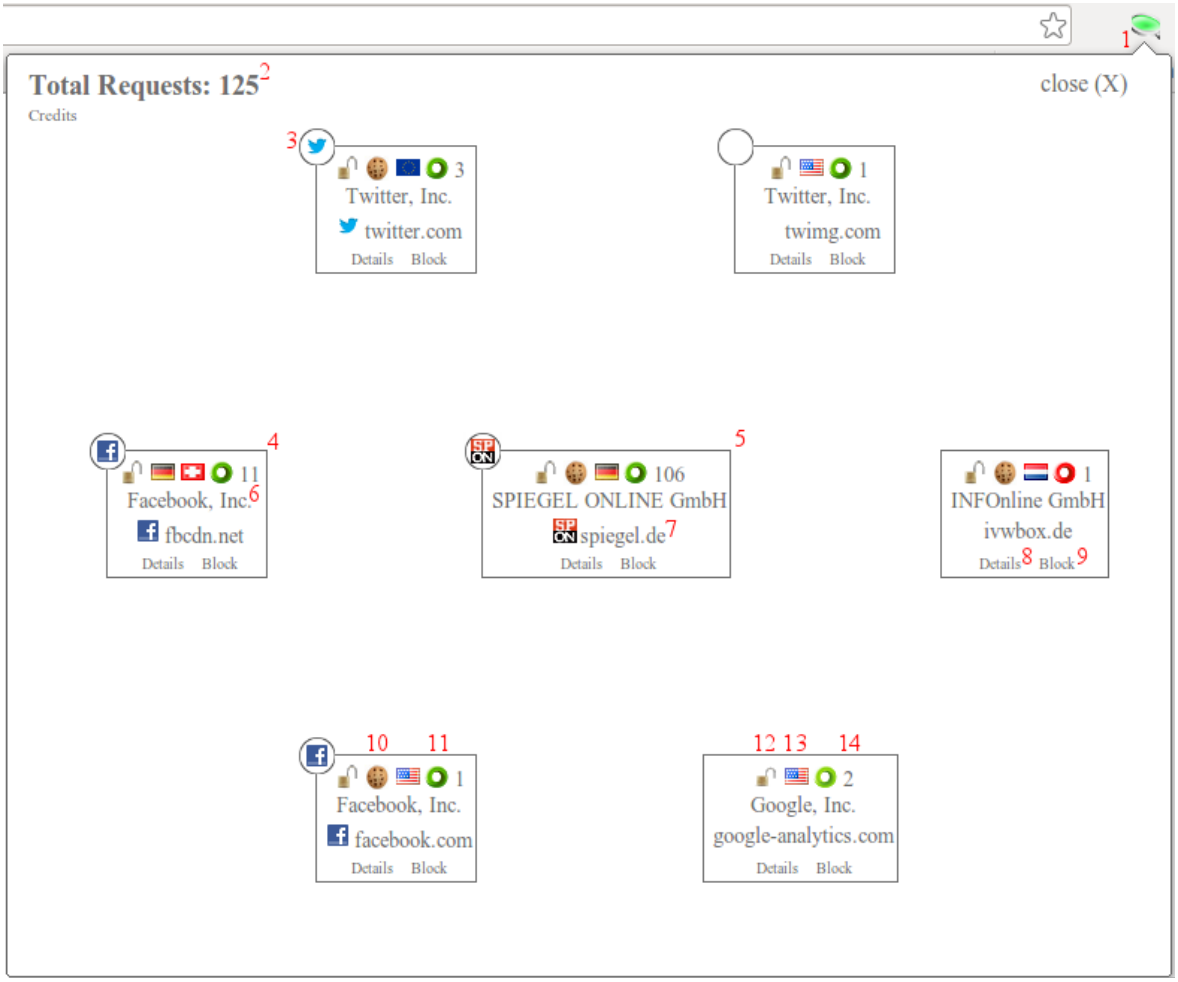

Abbildung 10: Screenshot des Pop-ups dieser Arbeit

1. *Icon der Erweiterung.*
2. *Anzahl aller Requests*
3. *Icon der Webseite*
4. *Darstellung eines Drittanbieters*
5. *Darstellung der aufgerufenen Seite*
6. *Firmenname*
7. *Domain*
8. *Link um alle Anfragen der Domain zu zeigen*
9. *Link um die Firma zu blockieren*
10. *Visualisierung, ob Cookies gesetzt werden*
11. *Web of Trust Rating*
12. *Visualisierung, ob die Kommunikation verschlüsselt ist*
13. *Landesflagge, Herkunft der Daten*
14. *Anzahl der Anfragen an diese Authority*

5.5.1 IP-Adresse

Mit der IP-Adresse kann selbst ein versierter Nutzer i.d.R. nicht viel anfangen. Auch für ihn ist es nötig, weitere Informationen, mit der IP-Adresse zu verknüpfen.
In diesem Fall kommen für die Visualisierung der IP-Adresse vor allem geografische Informationen in Frage, das Land, die Stadt und ggf. die Straße inklusive Hausnummer. Für eine Zuordnung von IP-Adresse zu Land gibt es kostenlose Datenbanken im Web. Datenbanken, die weitere Informationen wie Stadt, Straße, oder Geolocation enthalten, sind nicht kostenlos (die Kosten reichen von einmalig 25$ für einen Download [72] bis zu 1459$ für eine Lizenz im Jahr [63] - je nach Umfang der Daten).
Eine andere Möglichkeit weitere Daten bezüglich einer IP-Adresse oder Domain zu bekommen ist die Ausführung des *whois*-Befehls. Dies kann lokal, über ein weiteres Plug-in und die Abfrage der Daten aus dieser Erweiterung über die chrome plugin API [53], geschehen. Dabei treten aber gleich mehrere Probleme auf. Der *whois*-Befehl ist unter Windows nicht nativ verfügbar, müsste also mitgeliefert oder zusätzlich installiert werden, die Häufigkeit der Abfragen bei den Registrierungsstellen ist begrenzt und *whois*-Daten weisen häufig nicht dieselbe Formatierung auf. Im entsprechenden RFC [67] steht zum Format lediglich *[..] the WHOIS server replies with text content [..]*. Einen Parser für *whois*-Daten scheint es für JavaScript nicht zu geben. Die bessere Variante ist, die Daten über ein Webinterface [124] im einem definierten Format abzufragen. Hierfür ist allerdings i.d.R. ein Account notwendig [122] [24], was weitere Aktionen des Nutzers erfordern würde. Eine dritte Alternative ist eine komplette, geparste *whois*-Datenbank herunterzuladen, diese ist allerdings wiederum kostenpflichtig (200$ für eine Top-Level-Domain [123]).
Deshalb beschränkt sich die Visualisierung in dieser Arbeit auf Länder auf Basis einer kostenlosen Datenbank (`http://software77.net/geo-ip/`). Um das Unternehmen zu visualisieren soll eine eigene Funktionalität eingerichtet werden, über die der Benutzer selbst Unternehmen zu Domains zuordnen kann. Diese Informationen sollen zentral auf einem Server der Freien Universität Berlin gespeichert und abgefragt werden können. Dies ist aber nicht Teil dieser Arbeit. Anstatt der IP-Adresse soll die Domain dargestellt werden, diese ist für einen Benutzer assoziativer und verständlicher.

5.5.2 URL, Schema und Domain

Aus der URL kann mit Hilfe des Schemas festgestellt werden, ob die Kommunikation verschlüsselt ist. Komplette URLs werden dem Benutzer nicht gezeigt, da diese teilweise sehr lang und damit unübersichtlich sein können. Aus ihr wird die Domain (nur Top-Level und Second-Level-Domain werden berücksichtigt, Beispiel: `google-analytics.com`) extrahiert und angezeigt.

In der grafischen Übersicht werden alle Requests, die an dieselbe Domain gehen, zusammengefasst. Zu der Domain wird ein kleines Bild (favicon.ico) angezeigt, wenn es von dieser abrufbar ist.

5.5.3 Antwort-Header

Unter den Response Headern befinden sich hauptsächlich technische Informationen. Bezüglich der Transparenz im Web bringen „cache-control“, „Content-Encoding“ und „Content-Length“ für den Benutzer keinen Mehrwert. Interessant für den Benutzer hingegen wäre der P3P-Header (Platform for Privacy Preferences Project) [69], der dazu dient automatisch die Datenschutzinformationen des Benutzers mit denen der Webseite vor dem Besuch abzugleichen. Mittels des P3P-Standards wird definiert, welche Daten, warum, wie lange und zu welchem Zweck gespeichert werden dürfen. Voraussetzung ist allerdings, dass der Benutzer den Angaben des Webseiten-Betreibers vertraut. Das W3C hat aber aufgrund mangelnder Unterstützung und Umsetzung der Browserhersteller die Arbeit an P3P eingestellt. Entsprechend wenige Webseiten nutzen P3P und deshalb wird auch dieser Header nicht visualisiert.
Visualisiert wird hingegen der X-Frame-Options-Header. Das „X“ zeigt an, dass der Header nicht Teil des offiziellen W3C-Standards ist. Dennoch wird er von den fünf großen Browsern unterstützt [86]. Der X-Frame-Options-Header kann die Werte: *DENY* und *SAMEORIGIN* annehmen. *DENY* teilt dem Webbrowser mit, Frame-Inhalte nicht anzuzeigen, *SAMEORIGIN* erlaubt nur Inhalte die die selbe Herkunft haben, wie die Webseite selbst. Der Einsatz des X-Frame-Options-Headers bei Webservern ist eine Möglichkeit *Clickjacking* entgegenzutreten. Sollte der Header mit einem der beiden Werte gesetzt sein, so wird dies als Icon angezeigt.

5.5.4 Cookies

Die Visualisierung von gesetzten Cookies ist sehr einfach und übersichtlich gehalten. Setzt eine Domain einen Cookie, egal bei welcher Anfrage, so wird ein entsprechendes Symbol angezeigt. Welche Informationen in dem Cookie gespeichert sind, bleibt für den Benutzer verborgen. Häufig werden mit dem Nutzer verknüpfte Informationen auf dem Server zentral abgelegt und im Cookie nur eine ID zur Wiedererkennung gespeichert. Ein Tooltip weist zusätzlich darauf hin, dass der Benutzer potentiell überwacht wird.

5.5.5 Serveranzahl

Die Anzahl der verschiedenen Server mit denen kommuniziert wird, kann nicht ermittelt werden. Hinter einer IP-Adresse kann sich ein einzelner Rechner oder auch eine ganze Serverfarm verbergen. Ebenso kann es sein, dass

der selbe Server unter verschiedenen IP-Adressen erreichbar ist. Der *Server*-Header ist im Allgemeinen bei allen Anfragen zu einer Domain gleich, bietet deshalb keine Möglichkeit der weiteren Differenzierung. Lediglich die Anzahl der Anfragen kann visualisiert werden, diese wird als Zahl dargestellt.

5.5.6 Länderinformationen

Gewonnene Länderinformationen sollen so verständlich und übersichtlich wie möglich dargestellt werden. Während der Entwicklung hat sich gezeigt, dass eine Webseite selten Daten aus mehr als zehn verschiedenen Ländern bezieht. Im Durchschnitt liegt die Zahl zwischen zwei und drei.
Eine Visualisierung auf einer Weltkarte mit ca. 200 Staaten, auf der zwei bis drei grafisch hervorgehoben werden, ist nicht übersichtlich. Deshalb (und aufgrund der in Abschnitt 5.5.1 genannten Gründe) beschränkt sich die geografische Visualisierung auf die Landesflagge und einen Tooltip der den Ländernamen zeigt.

5.5.7 Tracking von Benutzerverhalten

Um herauszufinden, ob die Möglichkeit besteht, dass Benutzerverhalten überwacht wird, müssten alle Möglichkeiten (Cookies, *localStorage*, Flash Cookies) der Speicherung von Informationen beim Benutzer wie in Abschnitt 2.3.4 beschrieben, analysiert werden. Selbst wenn dann theoretisch die Möglichkeit der Überwachung gegeben ist, ist es dennoch nicht sicher, dass eine Firma diese auch nutzt. Deshalb wird Tracking von Benutzerverhalten nicht explizit visualisiert, sondern nur in einem Tooltip über dem Cookie-Icon der Hinweis gegeben, dass eine Überwachung theoretisch möglich ist.

5.6 Implementierung

Die Hauptfunktion der Software, das Visualisieren der Authorities, mit denen kommuniziert wird, ist einfach zu starten, der Benutzer klickt auf das Icon der Erweiterung im Browserfenster und das Pop-up öffnet sich.
Alle dargestellten Informationen, wie Verschlüsselung der Anfragen, geografische Herkunft der Daten, Anzahl der Anfragen und ob Cookies gesetzt werden oder nicht, werden als Symbol oder Zahl dargestellt. Ein Tooltip erklärt die einzelnen Indikatoren noch einmal zusätzlich.
Als Programmiersprache kommt ausschließlich JavaScript zum Einsatz. Die Dateien sind entsprechend ihren Modulen geordnet. Alle Dateien, die das Pop-up bilden, Stylesheets, JavaScripts und HTML-Dateien liegen im Verzeichnis (popup/).
Die Zuordnung von IP-Adressen zu Ländern befindet sich in der Datei ip2country.js im Wurzelverzeichnis der Erweiterung. Da die Datei ca. 7 Megabyte groß ist, ist es möglich, dass sich der Start des Browsers etwas verzögert. Sie ist mit Abstand die größte Datei im Projekt, umfasst

aber auch über 100.000 Datensätze. Die Daten stammen von der Webseite `http://software77.net/geo-ip/`. Dort kann eine CSV-Datei kostenlos heruntergeladen werden. Mittels eines Python-Scripts (siehe Listing 19, S. 61) wird diese in eine JavaScript-Datei umgewandelt. Dieses stellt jedoch eine externe Aktion dar, die nicht Bestandteil der Erweiterung ist.
In der Chrome API gibt es keine Möglichkeit weitere Informationen über eine aktuelle Verschlüsselung abzufragen. Eine Kommunikation wird als verschlüsselt betrachtet, wenn das Schema „https“ ist.

Bei Requests gibt es das Attribut *fromCache*, das anzeigt, ob die Ressource aus dem Chrome on-disk cache geladen wurde. Chrome verfügt über einen zweiten schnelleren in-memory cache. Wird eine Ressource aus dem in-memory cache geladen, so wird dies keinem der *webRequest*-Listener (siehe Abschnitt 5.6.1) mitgeteilt. Daher kann es bei mehrmaligem Aufruf einer Seite zu unterschiedlichen Resultaten kommen.
In den Optionen der Erweiterung kann eingestellt werden, Anfragen an den on-disk cache ebenfalls auszuwerten. Per Default werden auch diese Anfragen gezeigt, weil sie eigentlich stattfinden würden und so der Nutzer die Möglichkeit hat, sie zu blockieren.

Das Pop-up wird wiederholt aktualisiert. In einem zunächst festen Intervall wird überprüft, ob neue Informationen vorliegen (WOT-Bewertung, Firmenname für eine Domain). Gibt es Änderungen, wird das Intervall halbiert, gibt es keine, verdoppelt sich das Intervall. Damit wird eine zu seltene oder zu häufige Abfrage reguliert.

5.6.1 Erstellung einer neuen Session - Zusammenspiel der webRequest-Events

Die *chrome.webRequest*-API stellt Listener bereit, um Requests während ihres Auftretens zu überwachen. Den Lebenszyklus von Requests zeigt Abb. 11. Das erste Event, das auftritt, ist das *onBeforeRequest*-Event. Mit dem entsprechenden Listener können Requests blockiert werden, bevor sie stattfinden. Der *onBeforeSendHeaders*-Listener bietet die Möglichkeit, HTTP-Header zu verändern. Für jeden Request tritt zuletzt immer ein *onCompleted*- oder ein *onErrorOccured*-Event auf.

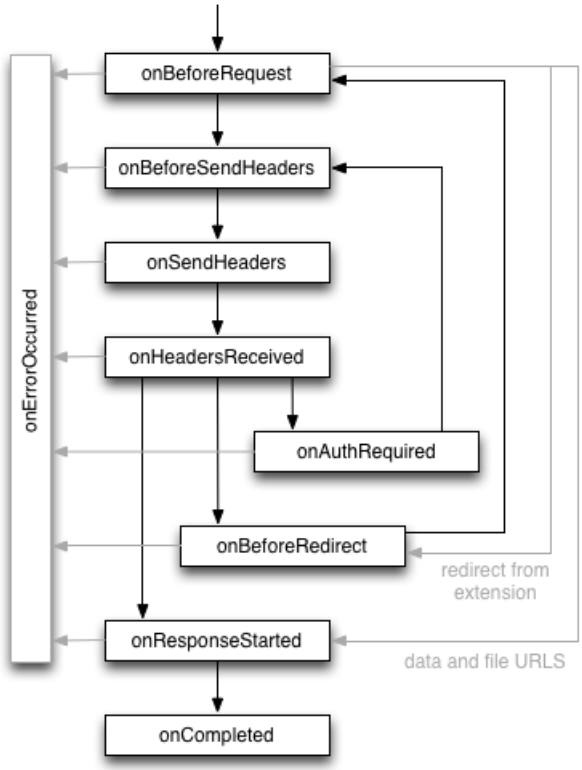

Abbildung 11: Lebenszyklus von Requests (Quelle: siehe Anhang, S. 62)

Listing 12 zeigt die Funktionsweise des *onBeforeRequest*-Listeners. Die anderen Listener funktionieren analog.

Listing 12: Listener, um auf *onBeforeRequest*-Events zu reagieren

```
chrome.webRequest.onBeforeRequest.addListener(function
    (details){
...
});
```

Das *details*-Objekt besitzt die Attribute *requestId*, *url*, *tabId*, *type* und weitere. Anhand der *requestId* können Anfragen über mehrere Listener hinweg verfolgt werden. Die *tabId* gibt an, in welchem Tab der Request aufgetreten ist. Anhand des *type*-Attributs kann identifiziert werden, um welche Art Anfrage es sich handelt. Folgende Werte kann das Attribut annehmen:

```
["main_frame", "sub_frame", "stylesheet", "script", "
    image", "object", "xmlhttprequest", "other"]
```

Eine neue Session wird erstellt, wenn das *onBeforeRequest*-Event mit dem *type* „main_frame“ auftritt. Um alle weiteren Requests zu sammeln, wird der *onCompleted*-Listener verwendet. Dabei werden nur Requests berücksichtigt,

bei denen kein Fehler auftritt (Requests, die durch andere Erweiterungen, wie z. B. Adblock blockiert werden, werfen einen Fehler).

Die Möglichkeit auf das *onBeforeRequest*-Event zu verzichten, gibt es nicht, da *onCompleted*-Events zwar auch über das Attribut *type* verfügen und so das „main_frame“-Event ermittelt werden kann, die Ereignisse aber nicht synchron auftreten, Listing 18 (S. 60) zeigt ein Beispiel. Damit könnten auftretende Requests zunächst keiner Session zugeordnet werden, weil noch keine existiert. Bei einer Speicherung und späteren Zuordnung entfällt die Möglichkeit, Requests zu blockieren (siehe Abschnitt 5.6.3), weil sie dann schon ausgeführt wurden.

Exkurs: In einer früheren Version der Implementierung wurden zunächst die Requests gespeichert, weil statt dem onBeforeRequest-Event das webNavigation-Event für die Erstellung einer neuen Session verwendet wurde. Dieses zeigt an, wenn eine neue Navigation (folgen einer URL, neues oder initiales Laden einer Seite) gestartet wurde. Leider treten webNavigation-Events ebenfalls nicht synchron auf, und einen eindeutigen Zusammenhang zwischen webRequest- und webNavigation-Events gibt es nicht. So ist nicht gewährleistet, dass alle Anfragen der richtigen Session zugeordnet werden. Die Chrome-Dokumentation sagt dazu Folgendes [44]:

> There is no defined ordering between events of the webRequest API and the events of the webNavigation API. It is possible that webRequest events are still received for frames that already started a new navigation, or that a navigation only proceeds after the network resources are already fully loaded.

Die Problematik findet in einem Bugreport [99] bereits Erwähnung. Die Resultate bei dem Einsatz des webNavigation-Events im Vergleich zu dem onBeforeRequest-Event waren bei mehreren Ausführungen die selben. Allerdings bietet, wie erwähnt, der Einsatz des webNavigation-Events nicht die Möglichkeit, Anfragen zu blockieren.

Sessions gelten als abgeschlossen, wenn eine neue Session in demselben (virtuellen) Tab erstellt wird, oder wenn ein Tab geschlossen wird. Obwohl Requests in demselben Tab stattfinden, ändert sich gelegentlich die *tabId* bei einer neuen Navigation. In diesem Fall wird das Abschließen einer Session über den *tabs.onRemoved*-Listener, siehe Listing 13 registriert. Abgeschlossene Sessions werden zur Übertragung gespeichert.

Listing 13: Listener, auf geschlossene Tabs zu reagieren

```
chrome.tabs.onRemoved.addListener(function(tabId,
    removeInfo){
...
});
```

5.6.2 Konfigurationsmöglichkeiten

Die Software bietet mehrere Möglichkeiten zur Konfiguration. Besteht die Gefahr, dass Passwörter unverschlüsselt übertragen werden, so ist es möglich, die Farbe des Icons ändern zu lassen. Je nach Einstellung kann die Farbe von Grün auf Gelb oder Rot wechseln. Für die Abschnitte

- „Communication Companies“, Benachrichtigung über die Kommunikation mit ausgewählten Firmen,
- „Communication Countries“, Benachrichtigung über die Kommunikation mit bestimmten Ländern, und
- „Information Storage“, Benachrichtigung über die lokale Speicherung von Daten,

sind die Einstellungsmöglichkeiten vielfältiger. Eine Menge von Elementen (Länder, Firmen) kann entweder auf eine Black- oder Whitelist gesetzt werden. Für eine so definierte Bedingung kann die Farbe des Icons geändert werden.

Beispiel: Der Benutzer setzt eine Firma, z. B. Facebook, Inc. auf die Blacklist im Abschnitt Communication Companies und stellt die Benachrichtigungsstufe auf Gelb. Dann wird das Icon bei jeder Webseite, die Daten von Facebook, Inc. bezieht, gelb. Würde die Firma auf der Whitelist stehen, so würde das Icon bei jeder Webseite gelb werden, die Daten von einer anderen Firma als Facebook, Inc. bezieht.

Die Funktionsweise für Communication Companies und Information Storage ist analog. Sind Einstellungen in mehreren Abschnitten getroffen, so ändert sich die Farbe auf Rot, wenn auch nur für eine Einstellung Rot gewählt wurde. Wurde Rot gar nicht gewählt, aber Gelb, so ändert sich die Farbe auf Gelb. In allen anderen Fällen bleibt das Icon grün. An der veränderten Farbe ist bei mehreren Einstellungen nicht abzulesen, welche Bedingung eingetreten ist. Weitere Informationen liefert in diesem Fall der Tooltip über dem Icon, siehe Abb. 12.

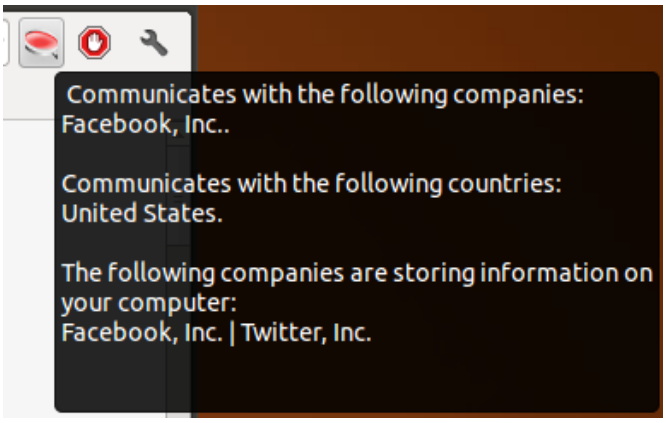

Abbildung 12: Tooltip dieser Erweiterung

Dabei gelten die Einstellungsmöglichkeiten Information Storage und Communication Companies nur für Drittanbieter. Bezieht man Daten von einer Firma, die auf der Blacklist steht, aber direkt aufgerufen wurde, zeigt die Erweiterung dies nicht an. Ebenso ist es, wenn die Firma, deren Webseite aufgerufen wurde, Cookies lokal speichert. Communication Countries informiert hingegen bei allen Kommunikationspartnern.
Die Optionen werden bei Änderungen direkt im *localStorage*-Objekt der Erweiterung gespeichert, ohne dass der Benutzer einen Speichern-Knopf drücken muss.
Da die Einstellungen für jeden Request geprüft werden müssen, werden diese ausgewertet, sobald ein neuer Request einer Session zugeordnet wird. Das bedeutet, dass die Optionen bei einigen Webseiten häufig (wenn die Anzahl der Anfragen hoch ist) ausgewertet werden.

5.6.3 Blockieren von Elementen

Über das Pop-up kann die Kommunikation mit ausgewählten Firmen blockiert werden. Dafür ist Voraussetzung, dass zu den gezeigten Domains die Firmen bekannt sind. Wenn keine weitere Navigation in dem Tab stattgefunden hat, kann auch nach einem Schließen und nochmaligem Öffnen des Pop-ups eine blockierte Firma wieder zugelassen werden. Nachdem eine Firma blockiert wurde und eine neue Navigation stattgefunden hat, wird die Kommunikation mit dieser Firma nicht mehr angezeigt - weil sie nicht mehr stattfindet. Um eine solche Kommunikation wieder zu erlauben, muss in den Optionen der Eintrag in der Liste der blockierten Elemente entfernt werden.
Das Blockieren geschieht bei Chrome in dem *onBeforeRequest*-Listener über Listen von URLs, siehe dazu Listing 14.

Listing 14: Listener zur Blockierung von Requests

```
block_listener = function(details){
    return {cancel: true}
}

chrome.webRequest.onBeforeRequest.addListener(
    block_listener,
    {urls: blocked_domains()},
    ["blocking"]
);
```

Der Listener setzt an, bevor Anfragen gestartet werden. Dabei werden alle URLs blockiert, die im zweiten Parameter als Liste unter „urls:“ übergeben werden und für die die Funktion (erster Parameter) ein Objekt mit dem Attribut *cancel*, das den Wert *true* besitzt, zurück gibt. Die URLs können auch Wildcards (*) enthalten. Da sich die Liste der zu blockierenden URLs bei jedem Hinzufügen oder Entfernen einer Firma ändert, existiert die Funktion *blocked_domains()*, die die Liste der zu blockierenden Domains generiert. In dem Beispiel werden alle Anfragen an URLs der übergebenen Liste blockiert.
Die Funktionalität kann aber dahingehend erweitert werden, dass nur die Kommunikation mit Drittanbietern blockiert wird.

Beispiel: Blockieren der Firma Facebook, Inc. (angefragte Domains: facebook.com, fbcdn.net) bei einem Besuch der Webseite `spiegel.de`*. Der Besuch der Webseite* `facebook.com` *(angefragte Domains: facebook.com, fbcdn.net) ist aber uneingeschränkt möglich.*

Dies ist mit der gegebenen Funktionalität möglich, weil die Session, zu der Requests zugeordnet werden, bereits existiert und so bei jeder Anfrage feststeht, ob sie an einen Drittanbieter geht oder nicht.
Eine Unstimmigkeit tritt allerdings auf, wenn Daten von einer Domain geladen werden, der noch keine Firma zugeordnet wurde, die Domain aber zu einer bereits blockierten Firma gehört. Anfragen werden, bis die Firma feststeht, zugelassen. Ein Tooltip im Pop-up weist in diesem Fall darauf hin, dass die Kommunikation erst nach einem Aktualisieren der Webseite blockiert wird.

Die Erweiterung Facebook Disconnect [23] nutzt eine andere Funktionalität um die Kommunikation mit Facebook zu unterbinden, wenn Daten von Facebook von einer anderen Seite aus abgefragt werden. Dafür wird ein Content Script eingesetzt, das auf das „beforeload“-Event reagiert. Listing 15 zeigt die Funktionsweise.

Listing 15: Funktionsweise der Erweiterung Facebook Disconnect

```
const DOMAINS = ['facebook.com', 'facebook.net', '
    fbcdn.net'];

/* Determines whether any of a bucket of domains is
    part of a URL, regex free. */
function isMatching(url, domains){
    [..]
}

/* Traps and selectively cancels a request. */
if (!isMatching(location.href, DOMAINS)) {
  document.addEventListener("beforeload", function(
      event) {
    if (isMatching(event.url, DOMAINS)) event.
        preventDefault();
  }, true);
}
```

Hier wird die Kommunikation nur blockiert, wenn *location.href* (die URL der aktuellen Seite) ungleich einer der auszuschließenden Domains ist und eine Anfrage an eine dieser Domains geht. Damit wird erreicht, dass die Domains `facebook.com`, `facebook.net` und `fbcdn.net` immer blockiert werden, wenn nicht eine der Domains direkt aufgerufen wird.
Allerdings gibt es keine Möglichkeit, die Liste der Domains dynamisch zu verändern. Wenn über das Pop-up die Kommunikation mit ausgewählten Firmen blockiert wird, müsste die Datei, die das Content Script enthält, neu geschrieben werden. Dies ist aber mit den gegebenen Funktionalitäten nicht möglich. Die Domain-Liste kann zwar über Message Passing (siehe Abschnitt 5.3) an das Content Script gesandt werden, aber die Kommunikation zwischen den Komponenten verläuft asynchron, sodass nicht garantiert werden kann, dass die Liste vor Einrichtung des „beforeload“-Listeners im Content Script verfügbar ist. Im Literaturverzeichnis findet sich unter [126] ein Foreneintrag, der auf die fehlende Unterstützung des synchronen Nachrichtenaustauschs hinweist. Ein entsprechender Bugreport [28] ist eröffnet.
Der Zugriff auf die Background Page über *chrome.extension.getBackgroundPage()* ist von einem injizierten Script ebenfalls nicht möglich, folgende Fehlermeldung erscheint:

Uncaught Error: „getBackgroundPage“ can only be used in extension processes. See the content scripts documentation for more details.

5.6.4 Cookies

Cookies werden im HTTP-Protokoll über die Header-Felder „Set-Cookie“ und „Cookie“ gesetzt und ausgelesen. Bei Chrome (getestet: Version 18 und 21) ist es möglich, auf diese Felder zuzugreifen, obwohl in der Manifest-Datei die Rechte für die Cookies API nicht gesetzt sind. Damit können Erweiterungen Cookies mitlesen, auch wenn der Benutzer davon ausgeht, dass dies nicht möglich ist, weil er bei der Installation nicht darauf hingewiesen wurde (Chrome informiert bei der Installation einer Erweiterung über benötigte Zugriffsrechte). In dieser Arbeit wird die Cookies API (siehe Listing 16) genutzt, weil es einfacher ist, Cookies auf diese Weise auszulesen, als Request- und Response-Objekte zu überwachen.

Listing 16: Funktionsweise der Chrome Cookies API

```
chrome.cookies.getAll(object details, function
    callback);
```

Dabei werden in dem übergebenen *details*-Objekt die Filterkriterien wie URL, Name und Domain festgelegt. Dem *callback* wird anschließend ein Array von Cookies übergeben. Wird in den Einstellungsmöglichkeiten von Chrome das Akzeptieren von Cookies von Drittanbietern abgeschaltet, so sind die Cookies trotzdem noch lokal vorhanden und werden durch die Erweiterung weiterhin visualisiert, bis sie gelöscht werden.

5.6.5 Übertragung der Daten an die Freie Universität Berlin

Sessions, die abgelaufen sind, werden zur Übertragung temporär gespeichert, d. h. bis der Browser geschlossen oder die Session übertragen wird. Eine Session wird als abgeschlossen betrachtet, wenn entweder im selben Tab eine neues *onBeforeRequest*-Event auftritt oder der Tab, in dem die Session stattgefunden hat, geschlossen wird. Führt ein Nutzer viele Navigationen schnell hintereinander aus, können Sessions unvollständig sein, weil nicht alle Requests stattfinden.
Gespeicherte Sessions werden in regelmäßigen Abständen an den FU-Server übertragen. Damit verfallen alle zuletzt gespeicherten Sessions, wenn der Browser geschlossen wird.
Für den Fall, dass noch keine Sessions gespeichert sind, wird dem Benutzer in den Optionen als Beispiel eine Anfrage an die Freie Universität Berlin gezeigt, die beinhaltet, welche Daten übertragen werden könnten. Sonst sieht der Benutzer die zuletzt gespeicherte Session im JSON-Format.

5.7 Wiederverwendbarkeit der Module

Das Modell von Sessions, Authorities und Requests kann auch auf eine andere Plattform übertragen werden, sofern die dort auftretenden Request-Objekte dieselben Attribute haben wie bei Google-Chrome. Ist dies nicht der Fall, so muss ein Adapter erstellt werden, der die Requests anpasst. Eine weitere Voraussetzung für die Wiederverwendbarkeit ist der erweiterungseigene *localStorage*, der auch auf anderen Plattformen zur Verfügung stehen muss, wenn an dieser Stelle nicht eine weitere Abstraktionsschicht implementiert wird.
Sofern die Visualisierung auf der anderen Plattform ebenfalls mit HTML, CSS und JavaScript geschieht, kann die komplette Logik für das Pop-up übernommen werden. Lediglich die Art der Datenübertragung muss angepasst werden.
Das Vorhandensein des *localStorage* und die Möglichkeit, asynchrone Anfragen zu stellen, sind notwendige Voraussetzungen für das WOT- und das Companies-Modul.

6 Fazit und Ausblick

6.1 Umsetzung der Ziele

Im Zuge dieser Arbeit ist ein Browser-Plug-in für transparentes Surfen entstanden. Die Erweiterung kann so konfiguriert werden, dass über das Erweiterungs-Icon visuell angezeigt wird, ob Passwörter auf der Seite unverschlüsselt übertragen werden, ob mit ausgewählten Ländern und/oder Firmen kommuniziert wird sowie ob Cookies bei der Kommunikation mit einer der Quellen der Webseite gesetzt werden. Eine Übersicht über die komplette Kommunikation, die mit einer Webseite verbunden ist, zeigt das Pop-up. Weitere Informationen wie Verschlüsselung, Web of Trust-Rating und beteiligte Firmen werden hier angezeigt. Zusätzlich ist es möglich, die Kommunikation mit Drittanbietern einzuschränken.
Das Plug-in visualisiert Informationen, die sonst für den Benutzer nicht ohne weitere Hilfsmittel sichtbar gewesen wären.
Damit wurden viele der in Kapitel 5.1 genannten Eigenschaften und Konfigurationsmöglichkeiten implementiert.
Um die Verständlichkeit, Erlernbarkeit und Bedienbarkeit zu testen, müsste Rücksprache mit verschiedenen Testpersonen gehalten werden. Die Ausarbeitung einer Umfrage hätte aber den Rahmen dieser Arbeit überstiegen.
Vergleicht man das Ergebnis der Arbeit mit den Voraussetzungen, die nötig sind, um einer Webseite zu vertrauen (siehe 2.1.1), so schafft die Erweiterung mehr Transparenz bezüglich des vierten genannten Punktes:
Der Benutzer erhält Kenntnis über alle Informationen (z. B. Anfragen, Speicherung von Cookies), die im Hintergrund ablaufen.

6.2 Kritik am gewählten System

Da alle funktionalen Ziele umzusetzen waren, hat sich die Entscheidung für Chrome als richtig herausgestellt.
Die Dokumentation zu Browser-Erweiterungen von Google ist sehr übersichtlich und verständlich. Sie ist gut strukturiert, und es gibt für viele Funktionalitäten ausreichend Beispiele. Zudem gibt es ein Tutorial zur Erstellung der ersten eigenen Erweiterung.
Die API wird stetig weiterentwickelt, seit dem ersten stabilen Release von Chrome Ende 2008 sind 20 neue Versionen erschienen. So wurde auch während der Entwicklung dieser Arbeit der Browser aktualisiert. Dabei wurden Features wieder entfernt und Altlasten beseitigt. So ist die erste Version des Manifest-File inzwischen *deprecated*, ab Chrome 18 wird empfohlen, auf die zweite Version umzusteigen, worauf Chrome auch hinweist, siehe Abb. 13. Deshalb musste während der Entwicklung die Software geändert werden, sodass Inline-JavaScript in eigenständige JavaScript-Dateien ausgelagert wurde. Die zusätzliche Arbeit wird aber durch ein sichereres System gerechtfertigt, das weniger anfällig gegenüber Cross-Site-Scripting-Angriffen ist.

Für die Manifest-Datei gibt es einen Zeitplan [52], der zunächst vorsieht, keine Updates für Erweiterungen mit Version 1 mehr auszuliefern. Anschließend können keine solchen Erweiterungen mehr in den Chrome Web Store eingestellt werden. Für 2013 ist angekündigt, diese Erweiterungen aus dem Web Store zu entfernen. Sie werden dann auch nicht mehr von Chrome unterstützt.
Damit erreicht Google ein langfristig sauberes und übersichtliches System, nicht mehr gepflegte Erweiterungen verschwinden aus dem Web Store.

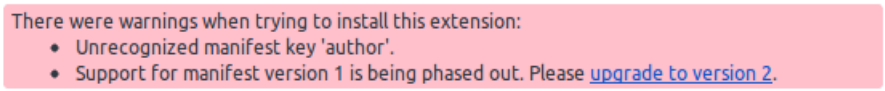

Abbildung 13: Hinweis auf eine veraltete Manifest-Datei

Werden neue Features der API hinzugefügt, befinden diese sich zunächst in dem Paket *experimental*, das nur in der Development-Version von Chrome zur Verfügung steht.

Nicht nachvollziehbar ist allerdings, dass für Requests, die von demselben Tab ausgehen, nach einer Navigation manchmal eine neue *tabId* vergeben wird. Die Funktionalität der Erweiterung wird dadurch nicht beeinflusst. Sessions werden richtig angezeigt, da sie auf Basis der *tabId* des aktiven Tabs gewählt werden. Erwartungsgemäß sollte dennoch ein Tab eine ID für seine komplette Lebenszeit behalten.
Anfragen, die aus dem in-memory cache beantwortet werden, sollten ebenso wie Anfragen, die aus dem on-disk cache stammen, über die API erreichbar sein.

6.3 Erfahrungen

Zunächst wurde ein Prototyp für Chrome und Firefox entwickelt, der die Mindestanforderung, die IP-Adresse für jede Anfrage mitzulesen, erfüllt. Mit Hilfe der Übersicht über die technischen Möglichkeiten der verfügbaren Browser wurde anschließend die Entscheidung für eine Entwicklung für Chrome gefällt. Diese Vorgehensweise hat sich, rückblickend betrachtet, bewährt. Der zunächst entwickelte Prototyp bildete eine gute Grundlage für das weitere Vorgehen.
Setzt man die Erweiterung ein, so fällt auf, dass in Webseiten oft Inhalte von den Firmen Facebook, Inc., Twitter, Inc. und Google, Inc. eingebunden sind. Im deutschsprachigen Raum ist auch häufig ein Zählpixel der INFOnline GmbH von der Domain `ivwbox.de` zu finden.
Eingebunden werden von Drittanbietern hauptsächlich iFrames, Scripte oder Zählpixel. Webseiten, die ihre Inhalte von mehr als zehn verschiedenen Do-

mains beziehen, sind selten, ebenso Webseiten, die nur eine einzige Quelle besitzen.

6.4 Weiterführende Arbeit

Die Arbeit ist nur für IP-Adressen der Version 4 ausgelegt. Eine IPv6-Adresse würde beim Nachschlagen in der Ländercodes-Tabelle kein Ergebnis liefern. Es könnte eine zusätzliche Tabelle hinzugefügt und zwischen IPv4 und IPv6 unterschieden werden. Eine Tabelle für IPv6-Adressen wäre allerdings sehr groß und würde die Ladezeit der Erweiterung stark negativ beeinflussen.

Google bietet eine API an, über die Informationen zentral gespeichert werden können [46]. Wenn die lokalen Einstellungen der Erweiterung bei Google gespeichert würden, stünden sie auch nach einer Neuinstallation von Chrome wieder zur Verfügung.
In dieser Arbeit werden bezüglich des Trackings nur Cookies berücksichtigt. Das Plug-in könnte dahingehend erweitert werden, dass auch andere Mittel (*localStorage*, Flash Cookies) zur lokalen Speicherung von Daten geprüft und entsprechend visualisiert werden.
Wie in Kapitel 3.6 beschrieben, kann auf Basis der in dieser Arbeit gesammelten Informationen ein System entstehen, das zentral Webdaten speichert und für Abfragen bereitstellt. Stimmt der Benutzer der Datenübertragung zu, werden in regelmäßigen Abständen die abgelaufenen Sessions im JSON-Format an den Server geschickt.

Ein ähnliches Browser-Plug-in könnte für Mozilla Firefox entwickelt werden. Dabei kann das Modell von Sessions, Authorities (siehe Abb. 9, S. 41) sowie der komplette Code für das Pop-up wiederverwendet werden. Allerdings stellt jede Plattform unterschiedliche Objekte, Mechanismen und Informationen bereit. Hier müssen bei einer Übertragung auf eine andere Plattform Anpassungen vorgenommen werden. Der Quellcode ist dokumentiert. Alle Funktionen werden beschrieben, und an wichtigen Stellen sind zusätzliche Erklärungen eingefügt. Dies unterstützt sowohl die Erweiterbarkeit, weil es für einen anderen Programmierer leichter ist, sich in der Struktur zurechtzufinden, als auch die Wartbarkeit.

Technisch ist es möglich, Requests, die von anderen Erweiterungen getätigt werden, ebenfalls zu protokollieren. Diese werden aber nicht visualisiert. Dafür müsste die bisherige Struktur geändert werden, sodass im Pop-up oder in einem neuen Tab deutlich ist, welche Erweiterung welchen Request stellt.
Blockiert eine andere Erweiterung (z. B. Adblock) ausgewählte Anfragen, so zeigt die im Rahmen dieser Arbeit entwickelte Erweiterung eine Kommuni-

kation nicht an, weil diese nicht stattfindet.
Bei anderen Erweiterungen wird die Kommunikation mit Hilfe von mitgelieferten Listen von URLs blockiert. Das ist für den Benutzer einfacher und komfortabler. Es muss nichts mehr konfiguriert werden. Die hier entwickelte Erweiterung könnte ebenfalls um eine Liste von Firmen ergänzt werden, deren Kommunikation, wenn sie als Drittanbieter auftreten, blockiert wird.

6.5 Fazit

Es gibt diverse Erweiterungen, die das Surfen im Netz transparenter machen und die Privatsphäre schützen. Die Wahl liegt im Umfang der Funktionalität, der Benutzerfreundlichkeit und des eingesetzten Browsers. Sollen z. B. nur Scripte von bestimmten Domains nicht mehr geladen werden, reicht die Installation der Erweiterung JavaScript Blacklist. Soll gezielt das Tracking durch Facebook verhindert werden, sollten Erweiterungen wie Facebook Disconnect und Facebook Blocker installiert werden. Mehr Informationen über blockierte Inhalte bieten Erweiterungen wie Do Not Track Plus und Ghostery. Die meisten Erweiterungen im Bereich Transparenz existieren für die Browser Chrome und Firefox.
Um alle Anfragen an Drittanbieter zu visualisieren, kann die in dieser Arbeit entwickelte Erweiterung genutzt werden. Sie zeigt übersichtlich wichtige Informationen bezüglich der Kommunikation. Zudem enthält sie zusätzliche umfangreiche Konfigurationsmöglichkeiten, um über ausgewählte Kommunikation zu benachrichtigen. Diese Möglichkeiten bietet keine andere der vorgestellten Erweiterungen.

7 Appendix

7.1 Listings

Listing 17: Beispiel für Requests, die zu unterschiedlichen Sessions gehören, aber dieselbe *tabId* besitzen

```
152 onBeforeRequest: ebay.de
152 onCompleted: ebaystatic.com stylesheet
[..]
152 onCompleted: ebay.de main_frame
152 onCompleted: ebayimg.com image
152 onCompleted: ivwbox.de image
152 onCompleted: 2o7.net image
152 onCompleted: ebaystatic.com script
[..]
152 onCompleted: ebaystatic.com image
152 onBeforeRequest: amazon.de
152 onCompleted: images-amazon.com image
[..]
152 onCompleted: amazon.de main_frame
152 onCompleted: images-amazon.com image
```

Listing 18: Beispiel für das verspätete Laden des „main_frames“

```
98 onCompleted: spiegel.de stylesheet
98 onCompleted: spiegel.de script
98 onCompleted: spiegel.de script
98 onCompleted: spiegel.de sub_frame
98 onCompleted: spiegel.de main_frame
98 onCompleted: cdn1.spiegel.de image
98 onCompleted: cdn3.spiegel.de image
98 onCompleted: cdn4.spiegel.de image
[..]
```

Listing 19: Python-Script zur Umwandlung der ip2country-Daten in JavaScript

```
import os

def to_int(i):
    i = i.replace('"','')
    return int(i)

js = open('ip2country.js','w')
js.write('ip2country = {\n')

last_ip = -1
f = open('IpToCountry.csv','r')
for line in f:
    if not line.startswith('#'):
        s = line.split(',')
        assert len(s) == 7, 'incorrect length: %s' %s
        s0 = to_int(s[0])
        s1 = to_int(s[1])
        s4 = s[4].lower()
        assert (s0 > last_ip), 'IPs not ordered'
        last_ip = s0
        js.write('%s: {"boundary":%s, "country_code":
           %s},\n' %(s0,s1,s4));

js.write('};')
f.close()
js.close()
```

7.2 Quellen verwendeter Abbildungen

Exemplarische Abbildung eines DOM:
`http://javascript.info/files/tutorial/browser/dom/simpledom2.png`
Linux Logo:
`http://upload.wikimedia.org/wikipedia/commons/3/35/Tux.svg`
Windows Logo:
`http://upload.wikimedia.org/wikipedia/en/b/b7/Windows_logo.svg`
Mac OSX Logo:
`http://upload.wikimedia.org/wikipedia/commons/e/e8/OS_X_Logo.png`

Abb. 3, Facebook Like-Button:
`http://www.b2b-grosshaendleradressen.de/blog/wp-content/uploads/2011/11/Facebook-Like-Button-big.jpg`
Abb. 5, Background Page:
`http://code.google.com/chrome/extensions/images/overview/arch-1.gif`
Abb. 6, User Interface Pages:
`http://code.google.com/chrome/extensions/images/overview/arch-2.gif`
Abb. 7, Content Scripts:
`http://code.google.com/chrome/extensions/images/overview/arch-3.gif`
Abb. 11, Lebenszyklus von Requests:
`http://developer.chrome.com/extensions/images/webrequestapi.png`

7.3 Weitere Abbildungen

```
{
  "name": "Transparency",
  "version": "0.5.7",
  "manifest_version": 2,
  "description": "Shows information about traffic
     origins.",
  "icons": {"16": "images/icon_16.png", "48": "images/
     icon_48.png", "128": "images/icon_128.png"},
  "browser_action": {
      "default_title": "Transparency",
      "default_icon": "images/browser_action_grey.png"
         ,
      "default_popup": "popup/index.html"
  },
  "background": {
      "scripts": ["jquery.js", "debug.js", "tools.js",
          "data.js", "companies.js", "wot.js", "
         background.js"]
  },
  "content_scripts": [{
      "matches": ["http://*/*", "https://*/*"],
      "js": ["jquery.js", "injection.js"],
      "run_at": "document_start"
  }],
  "options_page": "options/index.html",
  "permissions": ["webRequest", "webRequestBlocking",
     "tabs", "cookies", "http://*/*", "https://*/*"],
  "content_security_policy": "script-src 'self';
     object-src 'self'"
}
```

Abbildung 14: Manifest-Datei der Transparency-Erweiterung

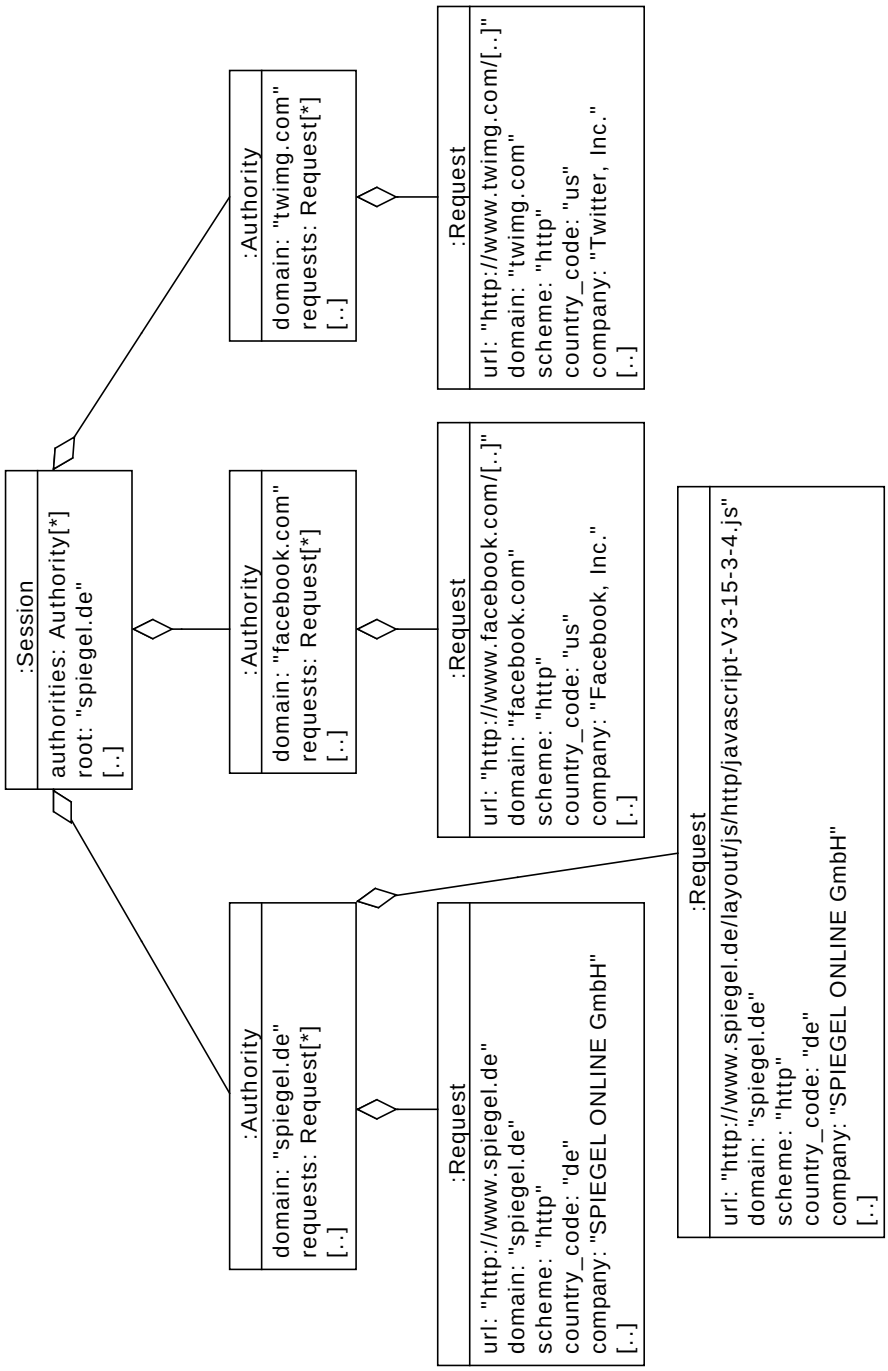

Abbildung 15: Objektdiagramm: Sessions, Authorities und Requests

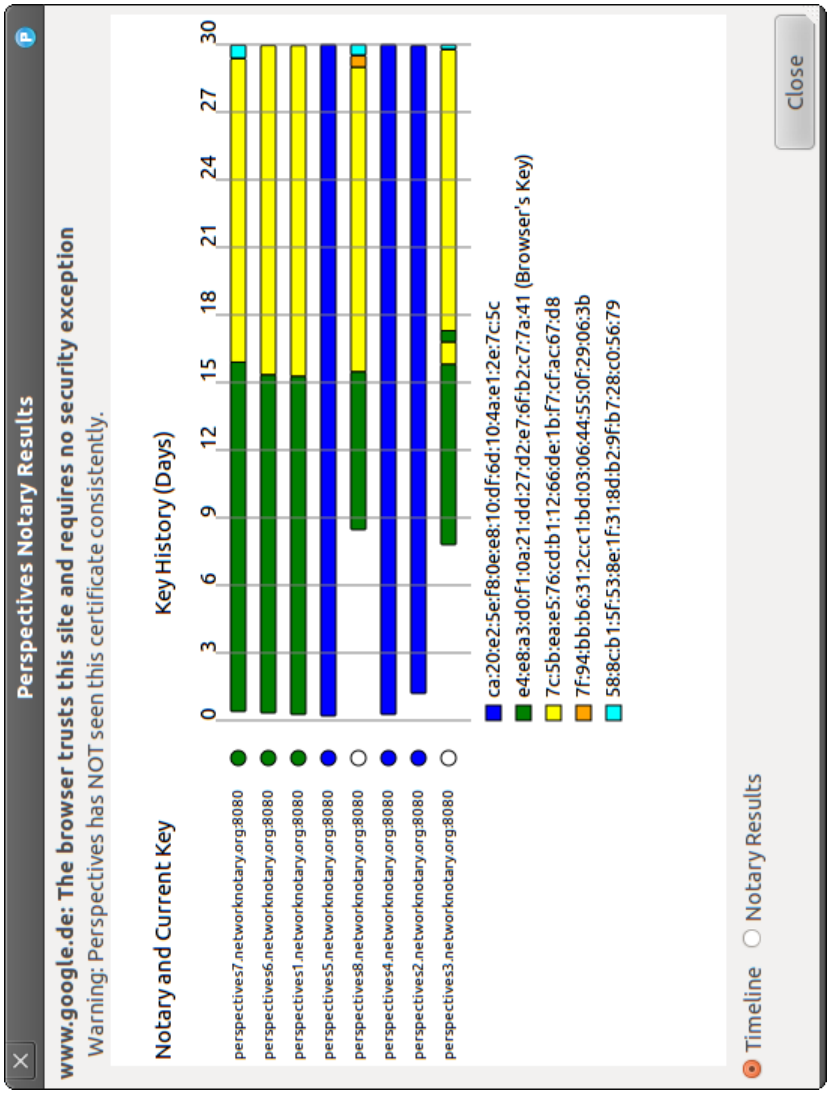

Abbildung 16: Historie des Zertifikats von `https://www.google.de` der Perspectives-Erweiterung

Abbildung 17: Pop-up der NoScript-Erweiterung

Abbildung 18: Pop-up der Vanilla Cookie Manager-Erweiterung

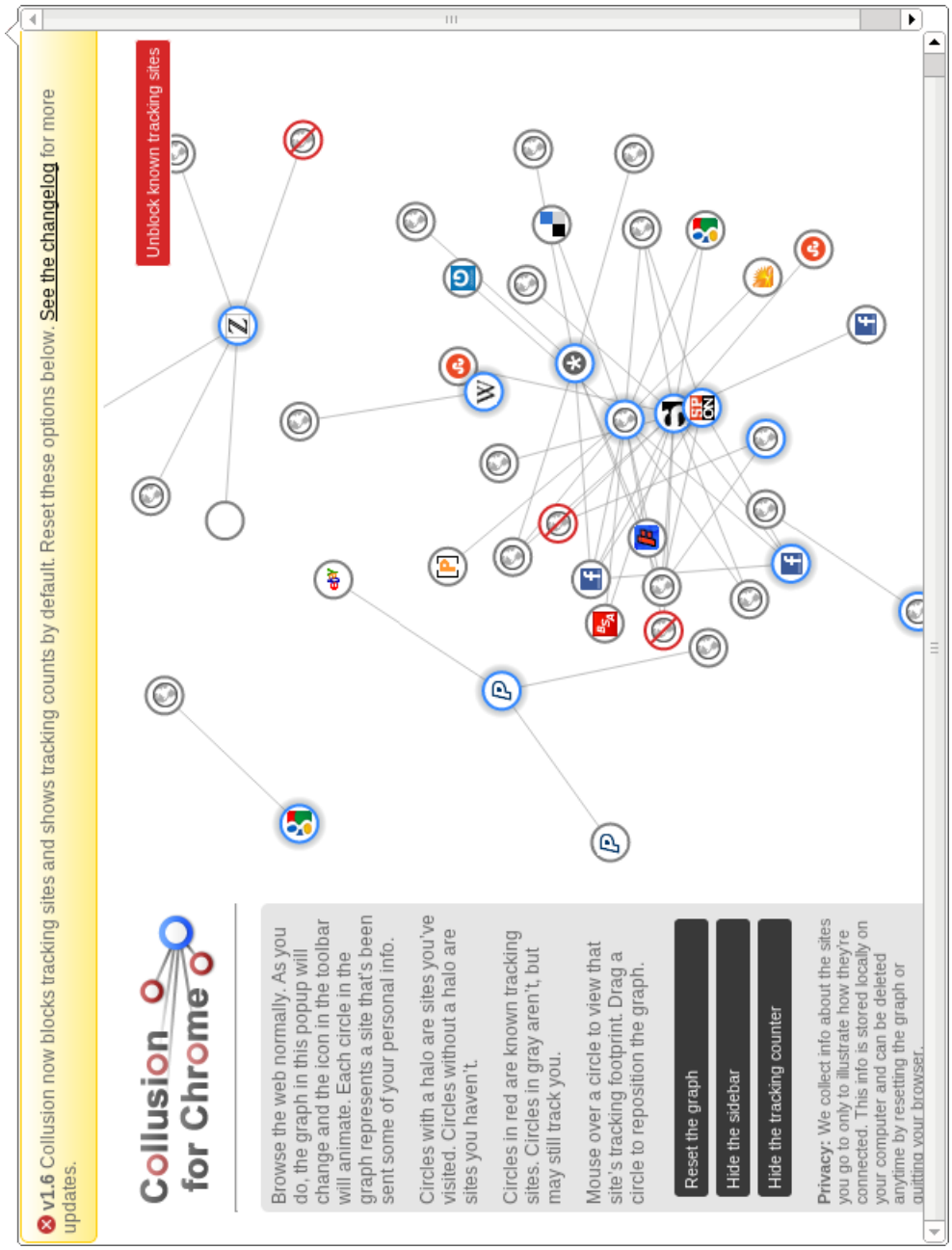

Abbildung 19: Pop-up der Collusion-Erweiterung

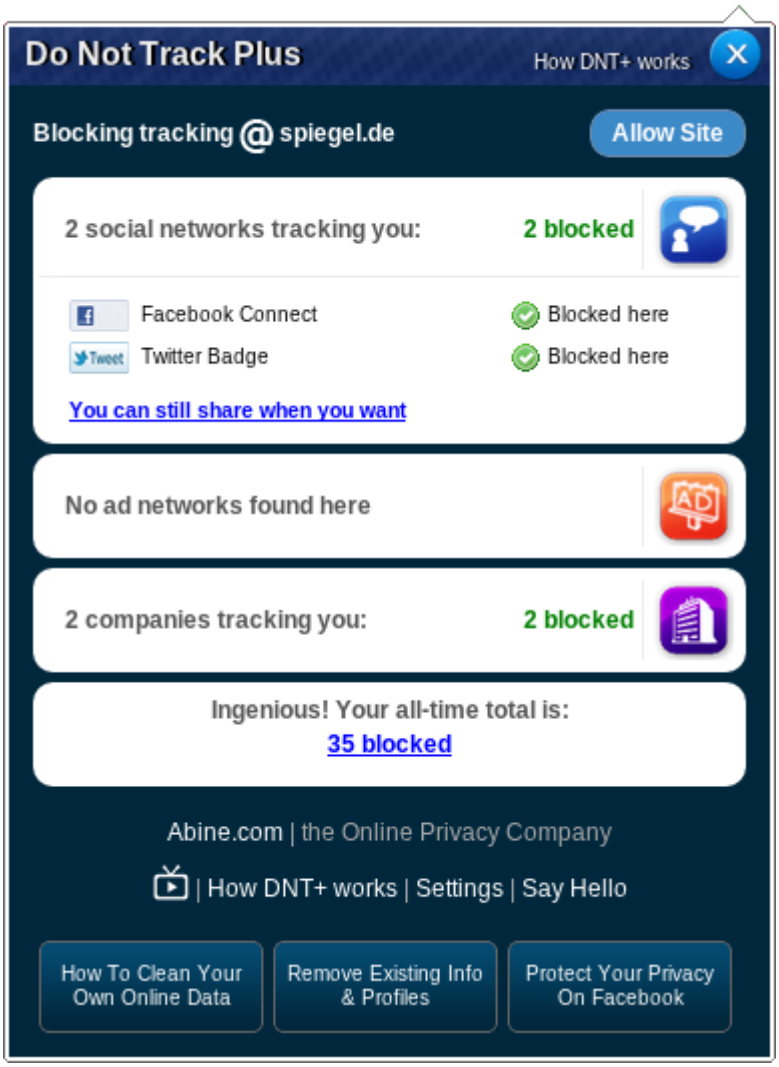

Abbildung 20: Pop-up der Do Not Track Plus-Erweiterung

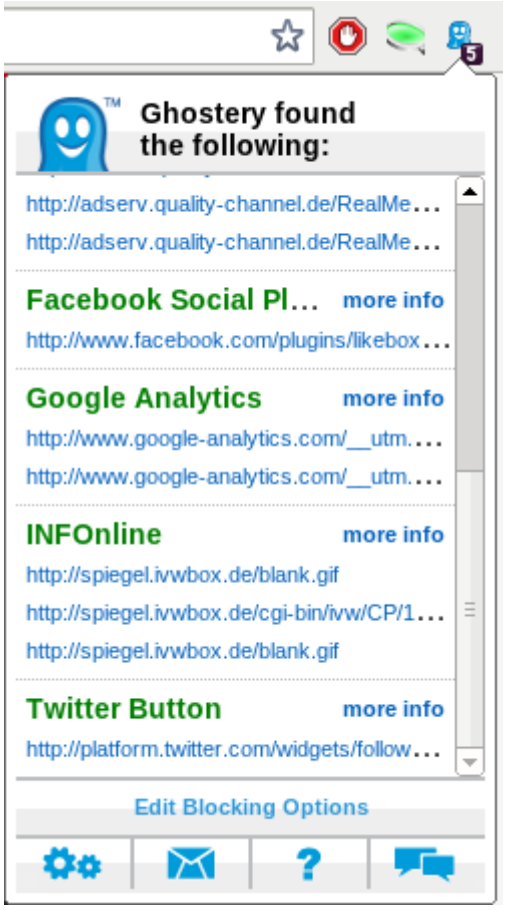

Abbildung 21: Pop-up der Ghostery-Erweiterung

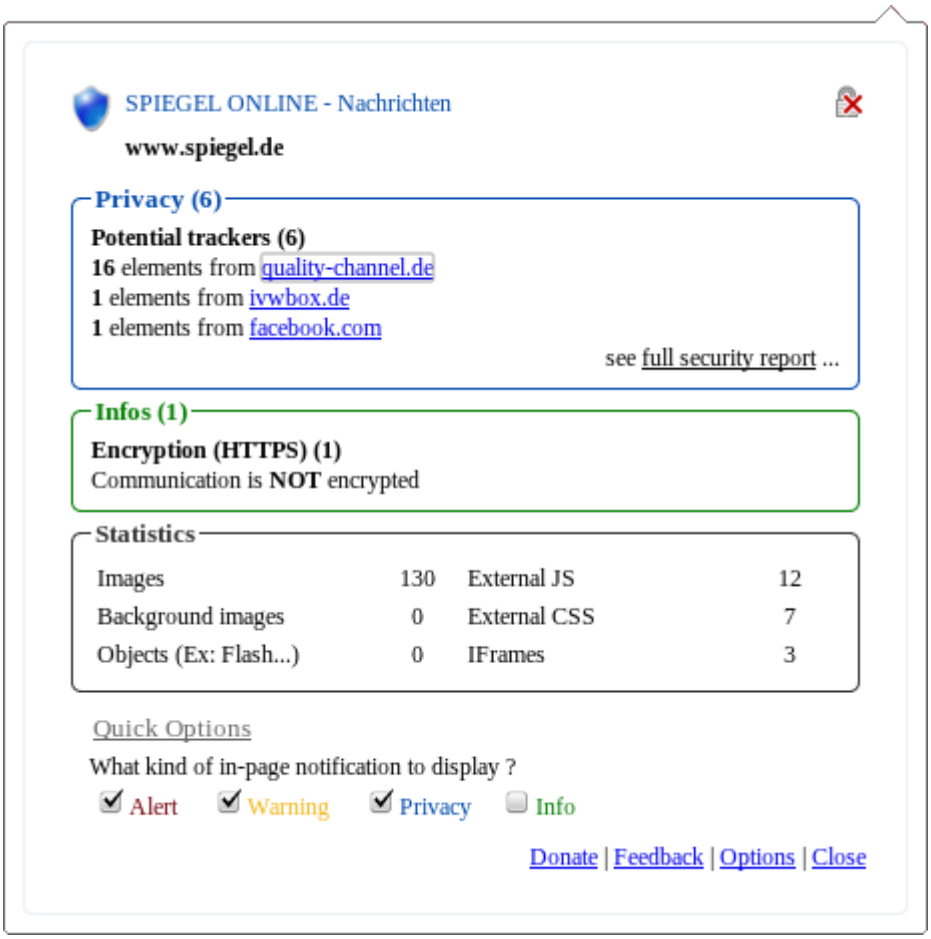

Abbildung 22: Pop-up der SaferChrome-Erweiterung

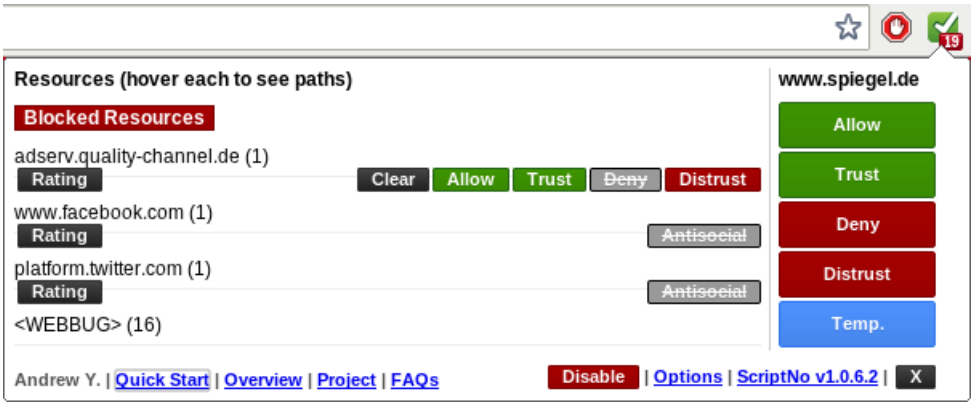

Abbildung 23: Pop-up der ScriptNo-Erweiterung

Mir ist bewusst, dass URLs nicht beständig sind und sich deshalb eigentlich nicht als Referenzen in einer Diplomarbeit eignen. Diese Arbeit ist aber stark Web-affin und die meisten der Quellen sind nur Online verfügbar. Alle Online-Quellen wurden am 1. Oktober 2012 überprüft.

Literatur

[1] Abine, Inc. Do Not Track Plus Erweiterung. `https://chrome.google.com/webstore/detail/epanfjkfahimkgomnigadpkobaefekcd`.

[2] Abine, Inc. Protect your Privacy with Abine. `http://www.abine.com`.

[3] Adobe Systems Incorporated. Adobe Flash Player. `http://www.adobe.com/products/flashplayer.html`.

[4] Alexa Internet, Inc. Alexa the Web Information Company. `http://www.alexa.com/`.

[5] Alrond, `http://www.alrond.com/`. WorldIP - flag and datacenter, ping and traceroute Add-on für Firefox. `https://addons.mozilla.org/en-US/firefox/addon/worldip-flag-and-datacenter-pi/`.

[6] Amazon EU S.a.r.l. Amazon Web Services. `http://aws.amazon.com/`.

[7] Andrew - andryou.com. ScriptNo Erweiterung. `https://chrome.google.com/webstore/detail/oiigbmnaadbkfbmpbfijlflahbdbdgdf`.

[8] Apple, Inc. `http://www.apple.com`.

[9] Apple, Inc. About Safari Extensions. `https://developer.apple.com/library/safari/#documentation/Tools/Conceptual/SafariExtensionGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009977-CH1-SW1`.

[10] Apple, Inc. SafariBeforeNavigateEvent Class Reference. `http://developer.apple.com/library/safari/#documentation/Extensions/Reference/SafariBeforeNavigateEventClassRef/SafariBeforeNavigateEvent.html#//apple_ref/doc/uid/TP40010860`.

[11] ARD/ZDF. Onlinestudie - Internetnutzer.
http://www.ard-zdf-onlinestudie.de/index.php?id=onlinenutzungprozen0.

[12] ARD/ZDF. Onlinestudie - Internetzugang.
http://www.ard-zdf-onlinestudie.de/index.php?id=361.

[13] BetaFish, Inc. AdBlock Erweiterung für Safari.
http://safariadblock.com/.

[14] Celince und Elie Bursztein. SaferChrome Erweiterung.
https://chrome.google.com/webstore/detail/lgpkjjingioekjianemgdobchenebhek.

[15] Celine Bursztein. Personal Web Page.
http://cealtea.tumblr.com/.

[16] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* Oldenbourg Wissenschaftsverlag; Auflage: überarbeitete und erweiterte Auflage; ISBN-13: 978-3486706871, 2011.

[17] Compete, Inc. http://www.compete.com.

[18] Coralllama. Vanilla Cookie Manager Erweiterung.
https://chrome.google.com/webstore/detail/gieohaicffldbmiilohhggbidhephnjj.

[19] Dagger. IPvFox Add-on für Firefox.
https://addons.mozilla.org/de/firefox/addon/ipvfox/.

[20] Dan Connolly. Standard Generalized Markup Language.
http://www.w3.org/MarkUp/SGML/.

[21] Dan Wendlandt, Dave Anderson und Adrian Perrig. Perspectives Project. http://perspectives-project.org/.

[22] Dave G, http://blog.flagfox.net/. Flagfox Add-on für Firefox.
https://addons.mozilla.org/de/firefox/addon/flagfox/.

[23] Disconnect, Inc. Facebook Disconnect Erweiterung.
https://chrome.google.com/webstore/detail/ejpepffjfmamnambagiibghpglaidiec.

[24] DomainTools, LLC. DomainTools API Account.
https://secure.domaintools.com/api/setup/pro/.

[25] don. IE extensions: Manipulating any request or viewing any response.
http://www.blackfishsoftware.com/blog/don/passthroughapp_bho_toolbar_intercepting_requests_responses.

[26] dschaefer. Perspectives Add-on für Firefox.
https://addons.mozilla.org/en-US/firefox/addon/perspectives/.

[27] EasyList. https://easylist.adblockplus.org/.

[28] eburszt...@gmail.com. Synchronous Message Passing - Bug Report.
http://code.google.com/p/chromium/issues/detail?id=54257.

[29] Elie Bursztein. Personal Web Page. http://elie.im/.

[30] Elika J. Etemad. Cascading Style Sheets (CSS) Snapshot 2010.
http://www.w3.org/TR/css-2010/.

[31] Erik van Kempen. BlockSite Add-on für Firefox.
https://addons.mozilla.org/en-US/firefox/addon/blocksite/.

[32] Evidon, Inc. http://www.evidon.com/.

[33] Evidon, Inc. Ghostery Erweiterung. http://www.ghostery.com/
Download: www.ghostery.com/download
FAQ: http://www.ghostery.com/faq.

[34] Facebook, Inc. http://www.facebook.com/.

[35] Facebook, Inc. Cookies, Pixels, and Similar Technologies.
http://www.facebook.com/help/cookies.

[36] Flambino. XMLHttpRequest for Safari Extension.
http://stackoverflow.com/questions/7045550/coding-with-xmlhttprequest-for-safari-extension.

[37] Francesco Capano. Forget Me Erweiterung.
https://chrome.google.com/webstore/detail/gekpdemielcmiiiackmeoppdgaggjgda.

[38] Frank Kölmel und André Stolze. Was bringt IPv6?
http://www.kes.info/archiv/online/05-5-012.htm.

[39] Franz Petermann. *Psychologie des Vertrauens*. Hogrefe-Verlag; Auflage: 3., korr. A; ISBN-13: 978-3801709327, 1996.

[40] Giorgio Maone. NoScript Add-on für Firefox.
https://addons.mozilla.org/de/firefox/addon/noscript/.

[41] Google, Inc. Advertising privacy FAQ.
http://www.google.com/policies/privacy/ads/.

[42] Google, Inc. Chrome Extensions - Message Passing.
http://code.google.com/chrome/extensions/messaging.html.

[43] Google, Inc. Chrome Extensions API.
http://code.google.com/chrome/extensions/api_index.html.

[44] Google, Inc. Chrome Extensions API - webNavigation.
http://code.google.com/chrome/extensions/webNavigation.html.

[45] Google, Inc. Chrome Extensions API - webRequest.
https://developer.chrome.com/extensions/webRequest.html.

[46] Google, Inc. Chrome External Storage.
http://developer.chrome.com/extensions/storage.html.

[47] Google, Inc. Chrome Web Store - Extensions.
https://chrome.google.com/webstore/category/extensions.

[48] Google, Inc. Google+. https://plus.google.com.

[49] Google, Inc. Google Web Store.
https://chrome.google.com/webstore.

[50] Google, Inc. IBA Opt-out Erweiterung.
https://chrome.google.com/webstore/detail/gbiekjoijknlhijdjbaadobpkdhmoebb.

[51] Google, Inc. Keep My Opt-Outs Erweiterung.
https://chrome.google.com/webstore/detail/hhnjdplhmcnkiecampfdgfjilccfpfoe.

[52] Google, Inc. Manifest Support Schedule.
http://developer.chrome.com/extensions/manifestVersion.html.

[53] Google, Inc. NPAPI Plugins.
http://developer.chrome.com/extensions/npapi.html.

[54] Google, Inc. Personal Blocklist Erweiterung.
https://chrome.google.com/webstore/detail/nolijncfnkgaikbjbdaogikpmpbdcdef.

[55] Helmut Balzert. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb, 3. Auflage.* Spektrum Akademischer Verlag; ISBN-13: 978-3827417060, 2011.

[56] Ian Hickson. HTML5 - A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/TR/html5/.

[57] Ian Hickson. Web SQL Database.
http://www.w3.org/TR/webdatabase/.

[58] Ian Hickson. Web Storage.
http://www.w3.org/TR/webstorage/.

[59] Ian Hickson. Web Workers - W3C Candidate Recommendation.
http://www.w3.org/TR/workers/.

[60] Ian Hickson et al. Web Hypertext Application Technology Working Group. http://www.whatwg.org/.

[61] Individual mozilla.org contributors. The Mozilla Foundation.
http://www.mozilla.org/foundation/.

[62] Information Sciences Institute, University of Southern California. RFC 791, Internet Protocol.
http://www.ietf.org/rfc/rfc791.txt, 1981.

[63] IP2Location.com. Complete Database.
http://www.ip2location.com/databases/
db20-ip-country-region-city-latitude-longitude-zipcode-\
timezone-isp-domain-netspeed-areacode-weather-mobile.

[64] JavaScript Object Notation. http://www.json.org/.

[65] Jiayong Ou. JavaScript Blacklist Erweiterung.
https://chrome.google.com/webstore/detail/
emcepjkdiiaenmoaghcfghjjppbkbhnf.

[66] josorek. Chrome Flags Erweiterung.
https://chrome.google.com/webstore/detail/
jhejngphiacapbgllhagbpdkkdieeaej.

[67] L. Daigle. RFC 3912, Whois - Protocol.
http://tools.ietf.org/html/rfc3912.

[68] Liam Quin. Extensible Markup Language (XML).
http://www.w3.org/XML/.

[69] Lorrie Cranor & Rigo Wenning. Platform Privacy Preferences - P3P.
http://www.w3.org/P3P/.

[70] Majestic-12 Limited. http://www.majesticseo.com.

[71] Mark Birbeck et al. XHTML™ 2.0.
http://www.w3.org/TR/xhtml2/.

[72] MaxMind, Inc. IP - Organization Database.
http://www.maxmind.com/app/organization.

[73] Michael, http://chromeadblock.com. AdBlock Erweiterung. https://chrome.google.com/webstore/detail/gighmmpiobklfepjocnamgkkbiglidom.

[74] Microsoft. Dns Class. http://msdn.microsoft.com/en-us/library/system.net.dns.aspx.

[75] Microsoft. IHttpNegotiate - OnResponse. http://msdn.microsoft.com/en-us/library/ie/ms775055(v=vs.85).aspx.

[76] Microsoft. Process - Klasse. http://msdn.microsoft.com/de-de/library/system.diagnostics.process(v=vs.100).aspx.

[77] Microsoft. X509Certificates Namespace. http://msdn.microsoft.com/en-us/library/system.security.cryptography.x509certificates.aspx.

[78] Mozilla Developer Network and individual contributors. Getting Started with Firefox Extensions. https://developer.mozilla.org/en/XUL_School/Getting_Started_with_Firefox_Extensions.

[79] Mozilla Developer Network and individual contributors. NPAPI Plugins. https://developer.mozilla.org/en-US/docs/Plugins.

[80] Mozilla Developer Network and individual contributors. nsIDNSService. https://developer.mozilla.org/en/XPCOM_Interface_Reference/nsIDNSService.

[81] Mozilla Developer Network and individual contributors. nsIObserverService. https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsIObserverService.

[82] Mozilla Developer Network and individual contributors. nsIProcess. https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsIProcess?redirectlocale=en-US&redirectslug=nsIProcess.

[83] Mozilla Developer Network and individual contributors. nsIXMLHttpRequest. https://developer.mozilla.org/en-US/docs/nsIXMLHttpRequest.

[84] Mozilla Developer Network and individual contributors. Observer Notifications.
https://developer.mozilla.org/en-US/docs/Observer_Notifications.

[85] Mozilla Developer Network and individual contributors. Security of an XMLHTTPRequest over SSL.
https://developer.mozilla.org/en-US/docs/How_to_check_the_security_state_of_an_XMLHTTPRequest_over_SSL.

[86] Mozilla Developer Network and individual contributors. X-Frame-Options-Response-Header.
https://developer.mozilla.org/en-US/docs/The_X-FRAME-OPTIONS_response_header#Browser_compatibility.

[87] NC. BetterPrivacy Add-on für Firefox.
https://addons.mozilla.org/de/firefox/addon/betterprivacy/.

[88] Net Applications. Browser market share - July 2012.
http://netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0.

[89] Nicolas Sylvain. Chromium Blog: A new approach to browser security: the Google Chrome Sandbox.
http://blog.chromium.org/2008/10/new-approach-to-browser-security-google.html.

[90] Opera Software ASA. http://www.opera.com/.

[91] Opera Software ASA. Extensions - Opera add-ons.
https://addons.opera.com/en/extensions/.

[92] ORACLE. Programmiersprache Java. www.java.com.

[93] Oracle Corporation. Java Applets. http://java.sun.com/applets/.

[94] P. Mockapetris. RFC 1034, Domain Names - Concepts and Facilities.
http://www.ietf.org/rfc/rfc1034.txt, 1987.

[95] P. Mockapetris. RFC 1035, Domain Names - Implementation and Specification. http://www.ietf.org/rfc/rfc1035.txt, 1987.

[96] P. Srisuresh et al. RFC 3022, Traditional IP Network Address Translator (Traditional NAT). http://www.ietf.org/rfc/rfc3022.txt, 2001.

[97] Paul Grebe, Wolfgang Müller. *Der Große Duden in 9 Bänden*. Dudenredaktion, Leitung Dr. phil. habil. Paul Grebe, 1964.

[98] Philippe Le Hégaret und Ian Jacobs. FAQ for HTML5 Last Call. http://www.w3.org/2011/05/html5lc-faq.html.

[99] pma...@google.com. WebRequest/WebNavigation events fire in the wrong order - Bug Report. http://code.google.com/p/chromium/issues/detail?id=88625.

[100] Prototype Core Team. prototype - JavaScript framework. http://www.prototypejs.org/.

[101] R. Arends et al. RFC 4033, DNS Security Introduction and Requirements. http://www.ietf.org/rfc/rfc4033.txt, 2005.

[102] R. Fielding et al. RFC 2616, Hypertext Transfer Protocol - HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt, 1999.

[103] S. Deering und R. Hinden. RFC 1883, Internet Protocol, Version 6 (IPv6) - Specification. http://www.ietf.org/rfc/rfc1883.txt, 1995.

[104] School of Computer Science, Carnegie Mellon. http://www.cs.cmu.edu/.

[105] Solvium. Browser Statistik. http://www.browser-statistik.de/.

[106] StatCounter. Browser Statistics. http://gs.statcounter.com/.

[107] StatCounter. StatCounter Global Stats - Browser, OS, Search Engine including Mobile Market Share. http://gs.statcounter.com/#browser-ww-weekly-201121-201221.

[108] Steven Pemberton et al. XHTML2 Working Group Home Page. http://www.w3.org/MarkUp/.

[109] The Dojo Foundation. Dojo Toolkit. http://dojotoolkit.org/.

[110] The jQuery Foundation. jQuery - write less, do more. http://jquery.com/.

[111] Tim Berners-Lee. Uniform Resource Locators (URL). http://www.w3.org/Addressing/URL/url-spec.txt.

[112] Tim O'Reilly. What Is Web 2.0. http://oreilly.com/web2/archive/what-is-web-20.html.

[113] Twitter, Inc. https://twitter.com/.

[114] Ulrich Druwe-Mikusin. *Moralische Pluralität. Grundlegung einer Analytischen Ethik der Politik.* Königshausen & Neumann; ISBN-13: 978-3884796306, 1997.

[115] `disconnect.me`. Collusion Erweiterung.
`https://chrome.google.com/webstore/detail/ganlifbpkcplnldliibcbegplfmcfigp`.

[116] `pmarks.net`. IPvFoo Erweiterung.
`https://chrome.google.com/webstore/detail/ecanpcehffngcegjmadlcijfolapggal`.

[117] `www.tcpiputils.com`. IP Address and Domain Information Erweiterung.
`https://chrome.google.com/webstore/detail/lhgkegeccnckoiliokondpaaalbhafoa`.

[118] W3C - World Wide Web Consortium. `http://www.w3.org/`.

[119] W3Schools. Browser Statistics.
`http://www.w3schools.com/browsers/browsers_stats.asp`.

[120] Webgraph, Inc. Facebook Blocker Erweiterung.
`http://webgraph.com/resources/facebookblocker/`.

[121] WHATWG. FAQ - WHATWG Wiki.
`http://wiki.whatwg.org/wiki/FAQ`.

[122] Whois API LLC. Terms of Service.
`http://whoisxmlapi.com/sla.php`.

[123] Whois API LLC. Whois Database Download.
`http://www.whoisxmlapi.com/whois-database-download.php`.

[124] Whois API LLC. Whois fu-berlin.de - JSON.
`http://www.whoisxmlapi.com/whoisserver/WhoisService?outputFormat=json&domainName=fu-berlin.de`.

[125] WOT Services Ltd. Web of Trust. `http://www.mywot.com`
Terms of Service: `http://www.mywot.com/en/terms/api`
FAQ: `http://www.mywot.com/en/faq/website/rating-websites`
Wiki: `http://www.mywot.com/wiki/Main_Page`.

[126] x64igor. Forenbeitrag zu Synchronous Message Passing.
`http://stackoverflow.com/questions/3996281/synchronous-message-passing-in-chrome-extensions`.

[127] Yo Momma. Open Certificate Information from Web Browser Control.
`http://stackoverflow.com/a/2941934`.

Software