Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik
Lehrstuhl für Systemarchitektur

Unverknüpfbare browserbasierte Authentifizierungen durch eine dritte Partei

Studienarbeit

eingereicht von: Jan Birkholz

Betreuer: Dr. rer. nat. Wolf Müller
Gutachter: Prof. Dr. rer. nat. Jens-Peter Redlich

eingereicht am: 14.11.2016

Zusammenfassung

Lässt ein Dienst im Internet die Authentifizierung eines Benutzers durch einen Dritten ausführen, so kann dieser aus der stattfindenden Kommunikation mitunter die Identität des Diensts erfahren. Aus Gründen der Datensparsamkeit und um Profilbildung entgegenzuwirken, kann das nicht wünschenswert sein. Es werden Möglichkeiten gesucht, welche ein Benutzer mit einem gängigen Webbrowser verwenden kann.

Inhaltsverzeichnis

1 Einführung **5**
- 1.1 Motivation . . . 5
- 1.2 Problemstellung . . . 6
- 1.3 Ziele . . . 6
- 1.4 Vorgehensweise . . . 7

2 Grundlagen **8**
- 2.1 Begriffe . . . 8
- 2.2 Rechtliche Rahmenbedingungen . . . 9
  - 2.2.1 Volkszählungsurteil . . . 10
  - 2.2.2 Bundesdatenschutzgesetz (BDSG) . . . 11
  - 2.2.3 Telemediengesetz (TMG) . . . 12
  - 2.2.4 Richtlinie 95/46/EG . . . 13
  - 2.2.5 Internationale Regelungen . . . 14
- 2.3 Verwandte Lösungen . . . 15
  - 2.3.1 Kerberos . . . 15
  - 2.3.2 OpenID . . . 16
  - 2.3.3 Auth^2(nPA) . . . 17
- 2.4 Zusammenfassung . . . 18

3 Lösung **19**
- 3.1 Techniken . . . 19
  - 3.1.1 Konzepte Browser . . . 19
  - 3.1.2 Cross-Origin Kommunikation . . . 26
  - 3.1.3 Unverknüpfbare Cross-Origin Kommunikation . . . 32
  - 3.1.4 Zusammenfassung . . . 37
- 3.2 Implementation Kommunikation . . . 37
  - 3.2.1 Versuchsaufbau . . . 37
  - 3.2.2 Webbrowser Spezifika . . . 38
  - 3.2.3 Kommunikation über IFrame mit HTTP POST und Web Messaging 43
  - 3.2.4 Weitere Kommunikations-Ansätze . . . 44
- 3.3 Authentifizierung . . . 46
- 3.4 Zusammenfassung . . . 47

4 Ergebnis **49**
4.1 Vergleich mit verwandten Lösungen 50
4.1.1 Kerberos 50
4.1.2 OpenID 51
4.1.3 Auth²(nPA) 51
4.2 Schwachstellen und Angriffsvektoren 52
4.3 Datenschutz 54

5 Fazit **55**
5.1 Schlussfolgerungen 55

1 Einführung

1.1 Motivation

Mit fortschreitender Digitalisierung, die in immer weitere Bereiche des Lebens vordringt, werden zunehmend mehr Informationen und sensiblere Daten verarbeitet. Damit einher geht eine gestiegene Bedeutung des Schutzes dieser Daten [Mas86, LB15]. Das stellt wiederum steigende Anforderungen an technische Systeme zur Datenverarbeitung. Einem Anbieter eines Diensts, der ein solches Schutzniveau einzuhalten hat, entsteht entsprechender Mehraufwand. Sind die Kosten für den Mehraufwand unverhältnismäßig hoch und der Dienst damit nicht mehr rentabel, wirkt dies innovationshemmend. Eine Lösung wäre, technischen Mehraufwand auszulagern und zwischen mehreren Anbietern zu teilen. Dabei darf jedoch das Schutzniveau nicht durch die zusätzliche Partei abgesenkt werden. Neben finanziellen Gründen, sind auch rechtliche und organisatorische Vorgaben vorstellbar, wegen denen der Dienst vom Anbieter nicht vollständig selbst erbracht werden kann.

Ein wichtiger Mechanismus, der zum Schutz sensibler Daten beiträgt, ist zugriffsberechtigte Benutzer zu authentifizieren. Dazu muss der Benutzer seine Identität nachweisen, worauf seine Angaben geprüft und er im Erfolgsfall authentifiziert wird. Einem authentifizierten Benutzer kann dann Zugriff auf Dienste gewährt werden, zu denen er berechtigt ist. Je sensibler Daten oder Systeme sind, desto stärker sollte die Authentifizierung gewählt werden. Wird versucht die Authentifizierung aus einem Datenverarbeitungssystem herauszulösen und durch einen Dritten durchzuführen, treten neben technischen Aspekten zur Funktionsweise auch Fragestellungen zur Vertraulichkeit der Benutzerinformationen auf. Ein Bestandteil ist die hinzugekommene Möglichkeit für den Dritten, Benutzer und Dienst zu verknüpfen.

Bestehende Lösungen im Bereich der Dritt-Partei Authentifizierung, sind die Protokolle Kerberos [SNS88] und OpenID [RR06]. Kerberos auf Netzwerkebene und OpenID webbasierend. Beiden Lösungen ist gemeinsam, dass der Dritte neben den Anmeldeinformationen die Identität des verwendeten Diensts erfährt. Mit dem Konzept Auth^2(nPA)[MRJ11] wird eine Authentifizierung durch einen Dritten bei fehlender Verknüpfungsmöglichkeit von Benutzer zu Dienst, unter Nutzung des Neuen Personalausweises und mittels eines anonymisierenden Proxys, beschrieben.

1.2 Problemstellung

Bei der Authentifizierung eines Benutzers durch eine dritte Partei, soll diese keine oder möglichst wenig unnötige Informationen zum Benutzer sammeln können. Vom Benutzer erhält der Dritte bereits, die für die Authentifizierung notwendigen Anmeldeinformationen. Der Dritte soll nicht zusätzlich die Identität des verwendeten Diensts kennen. Die Bedeutung des Wissens über die Identität des Diensts, lässt sich am Beispiel mit mehreren Diensten, die den gleichen Dritten zur Authentifizierung nutzen, veranschaulichen. Werden dann noch vom Benutzer personenbezogene Daten oder biometrische Merkmale für die Authentifizierung erhoben, also die Qualität der identifizierenden Merkmale erhöht, gewinnt das Wissen des Dritten um die Kenntnis der zweiten Partei noch mehr an Bedeutung. Dann könnte der Dritte auf stark identifizierenden Merkmalen Profilbildung betreiben.

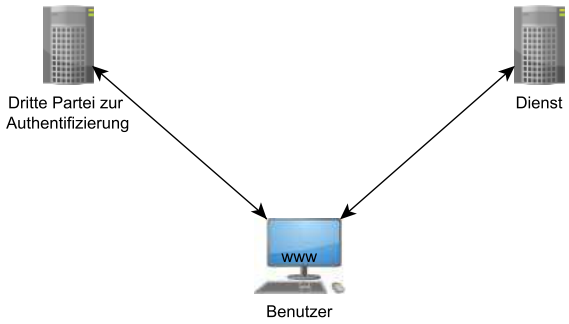

Abbildung 1.1: Beteiligte Parteien der Problemstellung.

Ein reines Verbot der Profilbildung, wäre eine Verschiebung von der technischen auf die organisatorische Ebene. Denn ein wirksames Verbot müsste regelmäßig überprüft werden. Weiterhin besteht für den Dritten immer noch die Möglichkeit die Informationen zu erlangen. Ein Angreifer oder Lauscher, der nicht an ein Verbot gebunden ist, könnte immer noch an die Informationen kommen. Aus diesem Grund scheiden auch Lösungen aus, die auf einer vertrauenswürdigen zentralen Instanz basieren.

Ein mögliches Verfahren sollte für Benutzer unproblematisch nutzbar sein. Es sollte also mit handelsüblichen Geräten, ohne spezielle Anpassung funktionieren.

1.3 Ziele

Es soll untersucht werden welche Möglichkeiten für Verfahren zur Dritt-Partei Authentifizierung bestehen, bei denen der Dritte Benutzer und Dienst nicht verknüpfen kann, und die von Benutzern im digitalen Alltag verwendet werden können. Weil mit Auth^2(nPA)

bereits eine Lösung mit einem anonymisierenden Proxy existiert, konzentriert sich diese Arbeit auf Lösungen ausschließlich mit den drei Beteiligten Benutzer, Dienst und authentifizierenden Dritten, wie in Abbildung 1.1.

Um möglichst viele Benutzer zu erreichen, soll eine weit verbreitete Plattform, sowie Standard-Technologien verwendet werden. Deswegen wird eine Nutzung des Internets, Web Technologien und beim Benutzer das Vorhandensein einer Software zur Anzeige von Webseiten, einem Webbrowser oder kurz Browser, angenommen. Allerdings ist es ausreichend einen Großteil der Browser zu unterstützen und für relevante moderne Schnittstellen alte Browser auszuschließen.

1.4 Vorgehensweise

Nach einer Einführung in relevante Begriffe, der datenschutzrechtlichen Perspektive zur Begründung der Notwendigkeit und in verwandte Techniken aus der Dritt-Partei Authentifizierung, werden im dritten Teil die relevanten Bestandteile und Techniken für einen Lösungsansatz aufgezeigt und eingegrenzt. Ein vielversprechender Lösungsansatz wird dann implementiert und dokumentiert. Abschließend werden die Ergebnisse zusammengefasst, diskutiert und im Kontext der aufgeworfenen Problematik eingeordnet.

2 Grundlagen

Um die weitere Arbeit vorzustellen werden kurz die verwendeten Begriffe abgesteckt und gegebenenfalls voneinander abgegrenzt. Darauffolgend wird die Datenschutzthematik behandelt, welche ursächlich für die Fragestellung dieser Arbeit ist. Zuletzt werden die bekannten Dritt-Partei Authentifizierungs-Lösungen Kerberos und OpenID, sowie das speziell auf Datenschutz ausgerichtete Auth^2(nPA) kurz erläutert und auf ihre Eignung für die Problemstellung eingegangen.

2.1 Begriffe

Ein *Benutzer* ist im Kontext der Arbeit eine natürliche Person, die ein mögliches Verfahren zur Lösung der Problemstellung in 1.2 benutzt. Also jemand, der einen Dienst nutzen möchte, welcher wiederum eine Authentifizierung durch einen Dritten voraussetzt.
Er besitzt dabei, neben seiner natürlichen *Identität* als Person und *Entität*[1], zwei verschiedene (partielle) Identitäten[2]. Eine für seine Rolle beim Dienstanbieter und eine weitere für seine Rolle bei der dritten Partei, zur Authentifizierung. Die Authentifizierung soll, wie eingehend in 1.3 „Ziele“ beschrieben, nicht durch den Dienst, sondern durch einen Dritten erbracht werden. Diese Art der Authentifizierung wird als *Dritt-Partei Authentifizierung* bezeichnet.

Eine Zuordnung zur Identität erfolgt mit einem eindeutigen *Identifikator*, wie zum Beispiel einem Benutzernamen. Um zu verifizieren, dass der Benutzer die Identität tatsächlich innehat, können charakteristische Eigenschaften, wie zum Beispiel ein Passwort, abgefragt werden. Bei Erfolg wird der Benutzer authentifiziert und (seine Identität) besitzt *Authentizität*.
Der Vorgang des Nachweises, bei dem überprüft wird, ob die behauptete Identität mit ihren charakteristischen Eigenschaften übereinstimmt, nennt sich *Authentifikation*. Dabei wird eine Identität an ein Subjekt gebunden [Bis02, 12. Authentication].

Authentifizierung wird in dieser Arbeit synonym mit Authentifikation verwendet. Im Unterschied dazu ist *Authentisierung*, das Aufstellen einer Behauptung über eine Identität [Hü08]. Eine Authentisierung ist im Normalfall Voraussetzung für eine darauffolgende Authentifikation.

[1] Vgl. Entität in [Hü08], wonach eine Entität aber auch ein Objekt sein kann.
[2] Vgl. partielle Identität in [Hü08].

Vertraulichkeit ist gegeben, wenn keine unautorisierte Informationsgewinnung möglich ist [Eck12, 1.2].

In Abgrenzung dazu ist *Datensicherheit* gegeben, sobald neben der grundsätzlichen Funktion eines Systems, keine unautorisierten Zugriffe (auf Daten) stattfinden können und Datenverlusten, beispielsweise mit Sicherheitskopien vorgebeugt wird [Eck12, 1.2]. Bei Datensicherheit geht es gegenüber Vertraulichkeit weniger darum die Informationen zu schützen, sondern mehr die Daten als Repräsentation vor Verlust und unberechtigten Zugriffen zu schützen.

Informationssicherheit dagegen gilt, sobald neben der grundsätzlichen Funktion eines Systems, unautorisiert weder Informationsgewinnung noch Informationsveränderung möglich ist [Eck12, 1.2]. Eine Definition über die Begriffe Vertraulichkeit, Integrität und Verfügbarkeit beschreibt den gleichen Sachverhalt[3]. Wobei *Integrität* den Schutz der Daten vor unautorisierten Veränderungen oder Datenverlust meint und *Verfügbarkeit* gewährleistet, dass die Informationen abrufbar sind und genutzt werden können [Bis02, 1.1]. Damit ist für Informationssicherheit sowohl Vertraulichkeit (der Informationen), als auch Datensicherheit Voraussetzung.

Datenschutz wird im Folgenden behandelt. Dabei wird auf die Herleitung aus dem Volkszählungsurteil eingegangen, als auch ein Einblick in das BDSG und damit dessen Definition gegeben.

2.2 Rechtliche Rahmenbedingungen

Die gesetzlichen Vorgaben legen den legalen Handlungsrahmen fest, in dem sich Benutzer und Datenverarbeitungssysteme bewegen. Sie regeln die Rechte des Benutzers und die Anforderungen an Datenverarbeitungssysteme. Für den Kontext dieser Arbeit erklären sie das Grundverständnis, aus dem die spezielle Problemstellung aus 1.2, der Nicht-Weitergabe des verwendeten Diensts bei Authentifizierung durch einen Dritten, erwächst. Der Gesetzgeber unterstreicht damit die Bedeutung des Schutzes der Benutzerdaten und fordert von Entwicklern von Datenverarbeitungssystemen eine Auseinandersetzung mit diesem Thema. Sowohl das Volkszählungsurteil des Bundesverfassungsgerichts aus dem Jahr 1983, als auch das Bundesdatenschutzgesetz (BDSG) und die europäischen Datenschutzvorgaben in Form der Richtlinie 95/46/EG sind maßgebend für den Schutz der Daten des Benutzers. Für deutsches Recht gilt zuerst die Spezialnorm und falls nicht vorhanden eine allgemeine, darüberliegende Norm, bis zur obersten Ebene, dem Grundgesetz.

[3] Vgl. [Bis02, 1.1] zu „computer security“ mittels confidentiality, integrity und availability.

2.2.1 Volkszählungsurteil

Ein Benutzer hat als natürliche Person zunächst das Recht auf informationelle Selbstbestimmung. Dieses Grundrecht wurde im Volkszählungsurteil des Bundesverfassungsgerichts aus dem Jahr 1983 entwickelt und begründet sich aus dem allgemeinen Persönlichkeitsrecht, welches wiederum aus dem Grundgesetz[GG49] Art 2 Abs. 1 in Verbindung mit Art 1 Abs. 1 abgeleitet wird [BSH+84][4].

> Das Grundrecht gewährleistet insoweit die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen. [BSH+84]

Hintergrund des Urteils sind die für den Einzelnen nicht durchschaubaren Möglichkeiten der modernen Datenverarbeitung und der damit einhergehenden Angst vor unkontrollierter Erfassung der Persönlichkeit, sowie fehlende Aufklärung zu Umfang und Verwendungszweck der Erhebung. Ein weiterer Aspekt war die Kritik Sachkundiger an Vorschriften und Bestimmungen zu Erhebung und Verwertung der Daten. [BSH+84]

Für den Einzelnen bedeutet es, dass er vor unbegrenztem Erheben, Speichern, Verwenden und Weitergeben seiner persönlichen Daten geschützt ist. Welche Daten schützenswert sind hängt vom jeweiligen Einzelfall der Erhebung ab, denn die Bedeutsamkeit der Daten ist abhängig davon, ob und wie sie zusammengeführt werden können [BSH+84, Fußnote 177].

Das bedeutet für einen Benutzer eines möglichen Verfahrens erst einmal Entscheidungsgewalt darüber, ob und wenn ja, welche Daten er bereit ist preiszugeben, als auch wie sie zu verwenden sind. Das gilt auch für die Weitergabe an Dritte.

Ein Benutzer ist wahrscheinlich eher gewillt seine Daten preiszugeben, wenn diese aus seiner Sicht vertraulich behandelt werden. Das enthält zum einen die technisch-organisatorische Komponente der Umsetzung, als auch die Art der Vermittlung der Vertraulichkeit.

Eine Einschränkung der informationellen Selbstbestimmung ist nur im „überwiegenden Allgemeininteresse“ und dann nur auf Basis einer gesetzlichen Grundlage möglich. Dabei hat ein solches Gesetz sowohl verfassungsmäßig, verständlich und verhältnismäßig zu sein. Dem Schutz vor Verletzungen des Persönlichkeitsrechts sollen zugehörige Datenschutzbestimmungen dienen. Ziel ist es Daten zweckgebunden und dafür in geringstmöglichem Umfang zu erfassen. Für diese Arbeit ist die Einschränkung der informationellen Selbstbestimmung im überwiegenden Allgemeininteresse nicht relevant, weil es sich um eine individuelle Möglichkeit zur Identitätsprüfung durch ein Datenverarbeitungssystem handelt. Das ist vergleichbar mit dem Prüfen des Ausweises beim Grenzübertritt. Deswegen finden auch die Anforderungen für statistische Erhebungen an dieser Stelle keine Anwendung.

[4] Oft zitiert als BVerfGE 65,1.

2.2.2 Bundesdatenschutzgesetz (BDSG)

In Folge des Volkszählungsurteils wurde 1990 das BDSG veröffentlicht, dessen Zweck es ist den Schutz personenbezogener Daten sicherzustellen, um das Persönlichkeitsrecht des Einzelnen zu wahren. Personenbezogene Daten, hier synonym verwendet mit persönlichen Daten, sind „Einzelangaben über persönliche oder sachliche Verhältnisse einer ... bestimmbaren natürlichen Person“[BDG90]. Also jede Angabe, die in Zusammenhang mit einer Person steht oder durch die enthaltenen Informationen gebracht werden kann. Das sind zum Beispiel Name, Post-Adresse, E-Mail-Adresse, Telefonnummer oder IP-Adresse[5]. Das BDSG dient als Auffanggesetz, falls keine gültigen bereichsspezifischen Vorschriften oder Landesdatenschutzgesetze existieren. Da ein Dienstanbieter nicht als staatliche Stelle mit besonderen Datenschutzanforderungen auftreten soll, fällt er in die Kategorie „nicht-öffentliche Stelle“.

Grundlegend sind das Verbot mit Erlaubnisvorbehalt, der Grundsatz der Datensparsamkeit und Datenvermeidung, sowie die unabdingbaren Rechte.
Das Verbot mit Erlaubnisvorbehalt verbietet erst einmal generell personenbezogene Daten zu erheben, zu verarbeiten oder zu nutzen und erlaubt dies im Nachgang nur bei klarer Rechtsgrundlage oder mit Zustimmung des Betroffenen. Eine Zustimmung ist dabei nur gültig, wenn sie aus freiem Willen abgegeben wird, sowie den Informationspflichten zu Zweck der Erhebung, Verarbeitung und Nutzung nachgekommen wurde.
Datensparsamkeit und Datenvermeidung dienen dazu, hinsichtlich eines Zwecks nur so viele Daten zu erheben, zu verarbeiten und zu nutzen, wie zu dessen Erfüllung gebraucht werden. Dies gilt auch für die einzelnen Schritte der Verarbeitung. Das Löschen, von für den Zweck nicht mehr notwendigen Daten, gehört explizit dazu. Das Prinzip gilt über den einzelnen Anwendungsfall hinaus und verlangt entsprechende technisch-organisatorische Verfahren.
Mehr auf den Einzelfall bezogen ist das Erforderlichkeitsprinzip, welches eine Erhebung, Verwendung oder Nutzung nur erforderlich macht, insofern die Aufgabe sonst nicht erfüllt werden kann.
Zu den unabdingbaren Rechte, die vertraglich gerade nicht ausgeschlossen werden können, gehört das Recht auf Auskunft, Berichtigung, Löschung und Sperrung der persönlichen Daten. Ausnahmen sind nur in Sonderfällen wie öffentlichem Interesse, bei Geschäftsgeheimnissen oder einem Geheimhaltungsinteresse Dritter möglich.

Entsteht durch Fehler bei Erhebung, Verarbeitung oder Nutzung persönlicher Daten ein Schaden, so ist ein privates Unternehmen zu Schadenersatz verpflichtet. Es sei denn es ist der entsprechend notwendigen Sorgfalt nachgekommen.

Da ein Benutzer das vertraglich nicht ausschließbare Recht auf Sperrung seiner Daten hat, können diese in Folge daran auch nicht an einen Dritten weitergegeben werden; zumindest ist das im Regelfall so. Zu beachten ist, dass bei der Auftragsdatenverarbeitung

[5]Nach Richtlinie 95/46/EG des Europäischen Parlaments und des Rates vom 24. Oktober 1995, Erwägung 26, gilt dies auch für Telemedienanbieter/Webseiten-Betreiber. Eine Entscheidung durch den EuGH (Rechtssache C-582/14) ist zum Zeitpunkt November 2015 noch offen.

durch einen Dritten keine Übermittlung im Sinne des BDSG stattfindet. Eine Auftragsdatenverarbeitung bringt vertragliche Pflichten mit, und Haftung und Verantwortung verbleiben beim Auftraggeber. Ohne Auftragsdatenverarbeitung wird für eine Weitergabe die Zustimmung des Benutzers oder eine andere Rechtsgrundlage benötigt. In dem Fall darf der Auftragnehmer nicht weisungsgebunden sein.

Die dritte, authentifizierende, Partei ließe sich sowohl als Auftragsdatenverarbeiter realisieren, als auch funktional getrennt. Funktional getrennt könnte ein Anbieter solch eines reinen Authentifizierungsdiensts, basierend auf einem Protokoll, unabhängig agieren. Dann wäre er selbst die verarbeitende Stelle. Zum Beispiel könnte er per Pauschalgebühr oder pro Authentifizierung für seine Diensterbringung Gebühren verlangen.
Werden allerdings nur nicht-personenbezogene Daten übermittelt, fällt eine Weitergabe nicht mehr unter das BDSG. Ein Verfahren ohne die Übermittlung personenbezogener Daten wäre wünschenswert.

Die vom Anbieter zu erfüllenden technischen und organisatorischen Maßnahmen zur Datensicherung sollen dem Gebot der Verhältnismäßigkeit folgen. Daher sollen Maßnahmen entsprechend dem Grad der Gefährdung des Persönlichkeitsrechts, bei einem Missbrauch der personenbezogenen Daten erfolgen. Verhältnismäßigkeit ist dabei so zu verstehen, dass die Maßnahmen Zutrittskontrolle, Zugangskontrolle, Zugriffskontrolle, Weitergabekontrolle, Eingabekontrolle, Auftragskontrolle, Verfügbarkeitskontrolle, jeweils durchgeführt werden müssen, aber nur insoweit, wie ihre Schutzwirkung in einem angemessenen Verhältnis zum Aufwand steht [PG07].

2.2.3 Telemediengesetz (TMG)

Ein Anbieter von IT Diensten fällt innerhalb Deutschlands unter das TMG [TMG07], denn er erbringt elektronische Informations- und Kommunikationsdienste, auch Telemedien genannt. Damit einher gehen Anforderungen an Informationspflichten, der Haftung und des Datenschutzes. Das verwandte Telekommunikationsgesetz (TKG) findet indes keine Anwendung bei netzwerk- und webbasierenden Diensten. Es regelt stattdessen Dienste, die hauptsächlich der reinen Übermittlung von Signalen über Telekommunikationsnetze dienen.

Die grundlegende Informationspflicht zur Kennzeichnung und Selbstauskunft des Anbieters ist auch bekannt als Impressumspflicht. Sie ermöglicht dem Benutzer zu erkennen, wer hinter dem Angebot steht und erleichtert die Kontaktaufnahme. Auch ist vor Nutzungsbeginn über Art, Umfang und Zweck von Erhebung und Verwendung von persönlichen Daten zu informieren. Die Informationspflicht betrifft auch eine eventuelle Weitergabe an Dritte, sowie spätere Identifizierung ein. Dazu gehören auch Cookies. Innerhalb der Europäischen Union gelten für einen Anbieter die Vorschriften des Herkunftslandes. Anbieter außerhalb der Europäischen Union sind nicht an das TMG gebunden. In dem Fall gilt aber immer noch internationales Privatrecht.

Für veröffentlichte Inhalte gilt zusätzlich zur Verantwortung eigener Veröffentlichungen, die Pflicht fremde Inhalte unverzüglich zu sperren, sobald Kenntnis über Rechtswidrigkeit eines Inhalts erlangt wird.
Die technische Infrastruktur ist gegen unerlaubten Zugriff abzusichern, um persönliche Daten und Verfügbarkeit zu schützen. Dabei gilt der Stand der Technik, auch hinsichtlich eines Verschlüsselungsverfahrens.

Für den Schutz persönlicher Daten sind zusätzlich zum BDSG Regelungen vorgesehen, die auf dem BDSG aufbauen oder daran angelehnt sind. Zum Beispiel gilt für die Erhebung, Verarbeitung und Nutzung von personenbezogenen Daten ein Verbot mit Erlaubnisvorbehalt durch: das BDSG, das TMG, eine andere Rechtsvorschrift oder freiwillige Einwilligung des Benutzers. Die Einwilligung ist dabei jederzeit widerrufbar.

Das TMG unterscheidet letztendlich zwischen personenbezogenen Bestands- und Nutzungsdaten. Bestandsdaten für ein Vertragsverhältnis, beispielsweise bei einem kostenpflichtigen Telemedium. Erforderliche Nutzungsdaten dienen zur Inanspruchnahme des Telemediums. Nutzungsdaten müssen im Regelfall nach der Nutzung gelöscht werden. Ausnahmen existieren für gesetzliche-, satzungsmäßige-, vertragliche- oder Abrechnungszwecke. Bei Verwendung von Pseudonymen dürfen Nutzungsprofile für bedarfsgerechte Gestaltung, Werbung oder Marktforschung erstellt werden. An Dritte dürfen die Daten, ausgenommen Auftragsdatenverarbeitung, anonymisiert für die Marktforschung weitergegeben werden. Für die Nutzungsprofile gilt zusätzlich, dass sie nicht mit Merkmalen zusammengeführt werden dürfen, die eine erneute Identifikation ermöglichen.

Das Telemediengesetz erlegt einem Dienstanbieter Pflichten zu einem rechtssicheren Anbieten von Telemedien auf und regelt über das BDSG hinausgehende datenschutzrechtliche Vorschriften bezogen auf Telemedien. Der Benutzer profitiert neben Kenntnis und Kontaktmöglichkeiten zum Anbieter, von klaren Regelungen im Umgang mit entstehenden Nutzungsdaten.

2.2.4 Richtlinie 95/46/EG

Die Richtlinie 95/46/EG der Europäische Union (EU) [EP95] dient dem Schutz natürlicher Personen bei der Verarbeitung ihrer personenbezogenen Daten, als auch dem freien Datenverkehr (personenbezogener Daten) innerhalb der EU. Sie setzt über die Mitgliedsstaaten hinweg Mindest-Standards, die in nationale Gesetze umgesetzt werden müssen und in Deutschland auch im BDSG und TMG umgesetzt sind. Ausgenommen sind der private Bereich, sowie zur Erfüllung staatliche Aufgaben hinsichtlich Außen- und Sicherheitspolitik und Polizei. Ihre Grundprinzipien entstammen der Europäischen Menschenrechtskonvention (EMRK). Die zu verhängenden Strafen legt jedes Land selbst fest.

Neben grundsätzlichen Anforderungen wie Rechtmäßigkeit von Verarbeitung und Zweck, sowie minimaler, zweckgebundener Verwendung, sind auch Betroffenenrechte wie Datenzugang, Widerspruchsrecht und Schadenersatz enthalten. Der Datenverarbeiter ist in der

Pflicht entsprechende Maßnahmen zum Schutz der Daten zu ergreifen und gegenüber einer unabhängigen nationalen Kontrollstelle meldepflichtig. Die Richtlinie enthält auch Regelungen zur Übermittlung in Drittländer außerhalb der EU.

Ergänzt wird die Richtlinie 95/46/EG durch die Datenschutzrichtlinie für elektronische Kommunikation (Richtlinie 2002/58/EG[6]) [EP02], welche Datenschutz für Telekommunikationsdienste regelt. Sie kann als europäisches Analog gesehen werden, zu dem, was in Deutschland mittels TMG und TKG umgesetzt ist.
Dazu gehört die Verpflichtung von Anbietern Informationssicherheit und ein Sicherheitskonzept für die Verarbeitung von personenbezogenen Daten umzusetzen. Für den Fall einer Schutzverletzung muss die nationale Kontrollstelle informiert werden, als auch der Benutzer, wenn sich die Verletzung auf dessen Privatsphäre auswirkt.
Weiterhin gibt es Vorschriften zur Überwachung von Nachrichten, die nur aufgrund gesetzlicher Vorgaben oder mit Einwilligung des Benutzers erfolgen darf. Für die aus dem TMG bekannten Nutzungs- und Abrechnungsdaten, ist auch eine Löschung nach Nutzung beziehungsweise Abrechnung vorgesehen.
Zum Schutz vor unerwünschten Nachrichten, Aufnahme in öffentliche Verzeichnisse oder Zugriff auf Informationen auf dem Endgerät des Benutzers (z.B. Cookies), muss der Benutzer vorher seine Einwilligung gegeben haben.

Da die Richtlinien bereits in nationale Gesetze umgesetzt wurden, sind sie vor allem als länderübergreifender Mindeststandard interessant, an dem sich Anbieter orientieren können. Benutzer können sich bei Anbietern innerhalb der EU sicher sein, dass gewisse Grundregeln eingehalten werden müssen.

2.2.5 Internationale Regelungen

International existieren seit 1980 die OECD-Datenschutzempfehlungen[Oec02] und das im Jahr darauf veröffentlichte „Übereinkommen zum Schutz des Menschen bei der automatischen Verarbeitung personenbezogener Daten (Konvention 108)“[Eur81] vom Europarat.
Davon ist nur die Konvention 108 des Europarats, als völkerrechtlicher Vertrag, für die unterzeichnenden Staaten, bindend [Eur81].

Die Datenschutzempfehlungen der Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD) [7] sollen in erster Linie gemeinsame Standards für den freien Datenaustausch schaffen [Oec02]. Die Ursprüngliche Empfehlung von 1980 wurde mit dem OECD Privacy Framework von 2013 aktualisiert, wobei die Grundprinzipien erhalten geblieben sind [OEC13]. Bei der Erarbeitung der Konvention 108 des Europarats, wurde mit der OECD und ihren nicht-europäischen Mitgliedsländern auf unterschiedlichen Ebenen zusammengearbeitet [CoE81, S. 4]. Deswegen verfolgen beide kohärente Ansätze. Die Grundsätze der Konvention 108 sind wiederum Ausgangspunkt der Prinzipien von

[6]Eine Aktualisierung der Richtlinie 2002/58/EG ist mittels Richtlinie 2009/136/EG erfolgt.
[7]Im Original: Organisation for Economic Co-operation and Development (OECD).

der Richtlinie 95/46/EG [CoE14, S. 18]. Deswegen lassen sich die OECD-Empfehlungen in der Konvention 108 genauer spezifiziert wiederfinden und noch mehr ausgestaltet in der Richtlinie 95/46/EG.

Allen gemeinsam ist das Ziel, personenbezogene Daten zumindest bei der automatisierten Verarbeitung zu schützen. Dazu zählt die rechtmäßige Beschaffung der Daten, die Datenqualität, definiert durch einen für den Erhebungszweck minimalen Umfang, sowie die Korrektheit und damit auch Aktualität der Daten. Es gilt personenbezogene Daten für vorher festgelegte Zwecke zu verwenden, mit Ausnahme der Einwilligung des Benutzers oder von Gesetzes wegen. Der Anbieter hat für die Informations- und Datensicherheit zu sorgen, mit dem Ergebnis, dass die Daten erhalten bleiben und nur befugten zur Verfügung stehen. Daneben bestehen für einen Anbieter Informationspflichten gegenüber den Betroffenen bezüglich des Vorhandenseins, der Art und des Hauptverwendungszwecks von Daten über sie, als auch Informationen über den Verantwortlichen der Datenverarbeitung. Für Betroffene soll die Möglichkeit des Einspruchs auf Richtigstellung der eigenen personenbezogenen Daten bestehen. Zur Durchsetzung dieser Anforderungen, steht der Anbieter in der entsprechenden Haftung.

Die 2013 angenommene Resolution 68/167[UNG13] der UN-Generalversammlung (UNGA), zielt in erster Linie darauf ab, das Menschenrecht der Privatsphäre[8] , trotz der möglich gewordenen technischen Überwachungsmöglichkeiten, durchzusetzen. Damit zielt die Resolution mehr auf den Schutz vor Überwachung ab, als Regeln aufzustellen, wie mit personenbezogenen Daten umgegangen werden kann.

2.3 Verwandte Lösungen

Um einen Überblick über bestehende Lösungen zur Dritt-Partei Authentifizierung zu gewinnen, werden neben dem Konzept Auth^2(nPA)[MRJ11], dessen Anforderungen der Nichtweitergabe der Dienst-Identität an den Dritten, denen der Problemstellung in 1.2 entsprechen, zwei weitere bekannte und verbreitete Lösungen aufgegriffen. Einerseits Kerberos [SNS88], welches die Dritt-Partei Authentifikation seit Ende der 1980er begleitet hat und mit Erweiterungen in Unix und Microsoft Betriebssystemen eingesetzt wird. Andererseits OpenID, dass aufgrund Verbreitung und Ausrichtung auf webbasierte Dienste den Anforderungen in den Zielen aus 1.3 entgegenkommt.

2.3.1 Kerberos

Kerberos [SNS88] ist ein Open Source Authentifikations-Dienst für offene Computer Netzwerke. Dienste und Benutzer vertrauen dabei auf Kerberos korrekte Identitätsfeststellung und sichere Authentifizierungsmechanismen. Es entstand aus der Problematik,

[8] Vgl. Artikel 12 der Allgemeinen Erklärung der Menschenrechte.

dass Dienste nicht der Identitätsfeststellung fremder Computer vertrauen konnten.
Basierend auf dem Needham-Schroeder Protokoll[9] für gegenseitige Authentifikation und Schlüsselaustausch, von jeweils zwei Kommunikationsparteien, werden sichere Verbindungen aufgebaut. Nach einer Authentifikation des Benutzers bei Kerberos, erhält der Benutzer ein Ticket, mit dem er wiederum ein Ticket für seinen gewünschten Dienst erfragen kann, welches er bei diesem zur Authentifikation einlösen kann.
Kerberos erfüllt die Funktion der Dritt-Partei Authentifikation und bietet auf dem Needham-Schroeder Protokoll aufbauend, Sicherheit gegen verschiedene Angriffsvektoren wie erneutes Senden abgehörter Tickets oder den Versuch Tickets selbst zu erzeugen. Grundsätzliche Schwierigkeiten, wie das Aufzeichnen des Schlüssels für die Authentifikation bei Kerberos, vergleichbar mit dem Ausspähen des eingegebenen Passworts, bleiben bestehen. Eine Besonderheit ist die Abhängigkeit von möglichst synchronen Uhrzeiten der beteiligte Computer, weil im Protokoll Zeitstempel verwendet werden.
Zur Lösung der Problemstellung in 1.2, einer Authentifikation durch den Dritten ohne Weitergabe der Identität des Diensts, ist Kerberos nicht geeignet. Denn der Dienst muss genau wie der Benutzer eine Authentifikation bei Kerberos durchführen muss. Damit gibt der Dienst Kerberos gegenüber seine Identität preis[10]. Außerdem kommunizieren Benutzer und Dienste auf Netzwerk-Ebene, was eine Software-Anpassung seitens des Benutzers und des Diensts notwendig macht. Das wiederum widerspricht dem Ziel aus 1.3, einer möglichst einfachen und hürdenlosen Verwendung durch den Benutzer.

2.3.2 OpenID

Mit OpenID[RR06] besteht eine webbasierte Lösung zur Dritt-Partei Authentifizierung. Aufgrund der höheren Abstraktionsebene gegenüber Kerberos entfallen eigene Implementationen von Verschlüsselung, Anwendungsintegration und Kommunikation. Schließlich setzen webbasierte Lösungen auf einem Browser mit dem Hypertext Transfer Protocol (HTTP) auf, welches in Form von HTTPS eine Verschlüsselung auf Transportebene enthält. Eine Kommunikation kann auf Basis von HTTP Requests wie zum Beispiel GET, POST oder PUT stattfinden.
Durch den dezentralen Aufbau ist keine Registrierung der Dienste wie bei Kerberos notwendig. Stattdessen erhält der Dienstanbieter, *relying party* genannt, alle notwendigen Informationen vom Benutzer.

Der Benutzer gibt dem Dienstanbieter seinen Identifikator bekannt, welcher Informationen zur Kommunikation mit dem Identity Provider enthält. Der *Identity Provider* ist die Dritte Partei, welche die Authentifizierung durchführt. Vom Identity Provider werden dann mögliche Identitäts-Dienste für diesen Benutzer abgefragt, wie zum Beispiel der OpenID authentication service. Mittels Weiterleitung zu diesem, kann sich der Benutzer bei der Dritten Partei authentifizieren, worauf er eine Weiterleitung zum Dienstanbieter mit dem verschlüsselten Identitätsnachweis erhält. Mit Erhalt des Nachweises durch

[9] Vgl. [NS78] mit Erweiterung [NS87].
[10] Vgl. [SNS88] S. 20.

den Dienstanbieter ist der Benutzer durch diesen authentifiziert und kann zum Beispiel weitere gesicherte Dienste in Anspruch nehmen. Nach erfolgreicher Authentifizierung kann das OpenID Data Transport Protocol für weitere Datenübertragungen zwischen Dienstanbieter und Identity Provider genutzt werden.

Da OpenID sowohl ein offener Standard ist und freie Implementierungen verfügbar sind[RR06, S. 12], als auch dezentral organisiert ist, besteht die Möglichkeit eines eigenen Identity-Providers. Eine mögliche Lösung kann also auch auf Basis einer Erweiterung von OpenID erfolgen. In der Standard-Implementierung kommuniziert der Dienstanbieter mittels im Identifikator enthaltenen Informationen mit dem Identity Provider und gibt damit der Dritten Partei seine eigene Identität Preis. Deshalb kann OpenID nicht ohne Anpassungen zur Problemlösung genutzt werden. Eine Änderung müsste dann auch mögliche weitere Kommunikationsmöglichkeiten zwischen Dienstanbieter und Identity Provider anders lösen, denn es besteht zum Beispiel nach Authentifikation für den Identity Provider die Möglichkeit mittels OpenID Data Transport Protocol, Kommunikation mit dem Dienstanbieter zu initiieren und so dessen Identität zu erfahren.

2.3.3 Auth^2(nPA)

Mit dem Konzept Auth^2(nPA)[MRJ11] wird eine Zweifaktorauthentifizierung basierend auf dem neuen Personalausweis (nPA) und dessen *eID-Funktion*[11] vorgestellt, bei der sich mehrere Dienstanbieter[12] die Infrastruktur zur Authentifizierung teilen. Der nPA ist ein hoheitliches Dokument und es werden durch den Gesetzgeber entsprechende Anforderungen an Sicherheit und Datenschutz gefordert, bevor ein Berechtigungszertifikat zum Auslesen erteilt wird. Dieses Schutzniveau soll auch bei einer geteilten Authentifizierungslösung eingehalten werden. Der hinzugekommene Dritte soll keine Daten über verwendete Dienste des Benutzers sammeln können, über die Profilbildung erfolgen könnte. Der verwendete Dienst soll keinen Zugriff auf das mittels eID-Funktion ausgelesene Pseudonym, die Restricted Identification (rID), bekommen.
Einer Profilbildung wird entgegengewirkt, indem die Dienst-Identität gegenüber dem authentifizierenden Dritten anonymisiert wird. So ist vorgesehen, dass der Benutzer die Daten vom Dienst an den Dritten in einem Web-Formular ohne Hinweis auf den Dienst sendet. Das Ergebnis der Authentifizierung beim Dritten wird über einen anonymisierenden Proxy an den Dienst übermittelt. Zur Wahrung der Datensicherheit und Vertraulichkeit werden die Nachrichten zwischen Dienstanbieter und Dritter Partei verschlüsselt.
Das vorgestellte Konzept entspricht einer Dritt-Partei Authentifizierung mit Unverknüpfbarkeit des Diensts, wie in der Problemstellung, in 1.2, beschrieben. Im Unterschied

[11] Die eID-Funktion ermöglicht digitales Auslesen der Datenfelder eines nPA. Der Anbieter benötigt ein behördliches Berechtigungszertifikat für die Datenfelder und der Benutzer bestätigt das Auslesen, bei angeschlossenem nPA, mit seiner PIN.

[12] In Auth^2(nPA)[MRJ11] entspricht die dortige Einrichtung einem Dienstanbieter und der dortige Dienstanbieter, der Dritten Partei zur Authentifizierung.

zu den Beteiligten der Problemstellung, in Abbildung 1.1, ist für die Kommunikation zwischen Drittem und Dienst der anonymisierende Proxy vorgesehen. Entsprechend seiner Funktion darf der Proxy sein Wissen nicht weitergeben.

2.4 Zusammenfassung

Die verwendeten und verwandten Begriffe um den Themenbereich der Authentifizierung durch einen Dritten wurden erläutert. Je nach Literatur unterscheidet sich die Bedeutung in einigen Begriffen. So sind Begriffe aus dem Datenschutz als abstrakter zu verstehen als äquivalente Begriffe aus der Datenverarbeitung.

Der Überblick über die rechtlichen Rahmenbedingungen hinsichtlich des Datenschutzes soll die Relevanz der Problemstellung, in 1.2, hervorheben. Denn in der Problemstellung wird gefordert, dass der authentifizierende Dritte nicht erfährt welcher Dienst benutzt wird. Dies geschieht vor dem Hintergrund einer Profilbildung. Sonst würden sich verbreitete und bestehende Verfahren, zum Beispiel OpenID, besser eignen.
Aus Datenschutzperspektive handelt es sich bei der gewünschten Eigenschaft der Problemstellung, die Identität des Diensts nicht an den Dritten weiterzugeben, um Datenvermeidung und Datensparsamkeit.

Für einen Anbieter eines Diensts sind innerhalb Deutschlands das Bundesdatenschutzgesetz und Telemediengesetz relevant. Sie setzen die entsprechenden EU Richtlinien um, welche wiederum Teile der Konvention 108 des Europarats als Ausgangspunkt haben, an deren Erstellung auch die OECD mitgewirkt hat.
Ein Dienst benötigt nach dem BDSG für persönliche Daten die Einwilligung des Benutzers zur Datenverarbeitung. Für den angegebenen Zweck dürfen die dafür notwendigen Daten dann genutzt werden. Dabei gilt für eine Ausgestaltung eines Datenverarbeitungssystems so wenig wie möglich persönliche Daten zu nutzen und stattdessen auf anonymisierte oder pseudonymisierte Daten zurückzugreifen. Im Rahmen des Telemediengesetzes werden ihm als Telemedienanbieter weitere Pflichten, wie ein Impressum, Haftung und Datenschutz im Zusammenhang mit entstehenden Bestands- und Nutzungsdaten, auferlegt. Weitere Details sind abhängig von den jeweils übermittelten Informationen bei der Authentifikation durch den Dritten.

Kerberos und OpenID sind als bekannte Vertreter für eine Authentifikation durch einen Dritten, für die spezielle Anforderung, der Nicht-Weitergabe der Dienst-Identität an den Dritten, aus der Problemstellung in 1.2, nicht geeignet. Auth^2(nPA) löst dies, indem die Kommunikation zwischen Drittem und Dienst über eine anonymisierende Proxy Komponente stattfindet. Deswegen konzentriert sich diese Arbeit auf Möglichkeiten ohne eine Proxy Komponente für die Kommunikation. Insbesondere wird versucht eine Kommunikation nur mit den drei Beteiligten wie in Abbildung 1.1 zu lösen.

3 Lösung

Entsprechend den Zielen aus 1.3 wird untersucht, welche Möglichkeiten zur Dritt-Partei Authentifizierung mit einem standardkonformen Browser ohne Weitergabe der Identität des Diensts an den authentifizierenden Dritten bestehen. Einschränkend kommt hinzu, dass ausschließlich die drei Parteien gemäß Abbildung 1.1 beteiligt sind. Deswegen beziehen sich die verwendeten Techniken auf die in gängigen Browsern angebotenen Programmierschnittstellen und deren Beschränkungen.
Ausgehend vom Wissen über verwandte Lösungen aus 2.3, werden Techniken gesucht und auf ihre Eignung geprüft. Das in 2.3.3 vorgestellte Konzept Auth^2(nPA) erfüllt die Anforderungen mittels eines anonymisierenden Proxy. Deswegen konzentriert sich diese Arbeit auf Techniken ohne eine anonymisierende Proxy Komponente, mit Kommunikation über den Benutzer, wie in Abbildung 1.1 der Problemstellung.
Unter Einsatz der gefundenen Techniken wird eine passende Kommunikation für eine Authentifizierung implementiert, auf deren Basis die Authentifizierung beschrieben wird.

3.1 Techniken

Weil Dienst und authentifizierender Dritter nicht direkt miteinander kommunizieren sollen, bestimmt der Benutzer mit seinem Browser die zur Verfügung stehende Umgebung und ihre Techniken. Diese werden in diesem Abschnitt untersucht.

3.1.1 Konzepte Browser

Der Webbrowser, als Anwendung um meist über HTTP übertragene Inhalte zu lokalisieren und anzuzeigen, wurde als zentrale Anwendung des Benutzers in der Problemstellung in 1.2 angenommen. Es ist davon auszugehen, dass der Browser nach erfolgter Authentifizierung, zur Nutzung der angebotenen Dienste des Dienstanbieters weiterverwendet wird.

Vom Browser unterstützte Technologien werden gemäß der Open Web Platform angenommen. Die Open Web Platform [W3C][Jaf14] ist eine Sammlung von offenen Web Standards, definiert durch Standards der World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF) und Ecma International. Die Hypertext Markup Language (HTML) wird dabei als zentrale Sprache des Web definiert [HBF$^+$14]. Von einer Beschreibungssprache für wissenschaftliche Dokumente, hat sich die HTML Spezifikation zu einer ganzheitlicheren Sprache für Web Anwendungen entwickelt. Dazu gehören

Programmierschnittstellen zur Kommunikation, Datenverarbeitung und Darstellung von Interaktiven Inhalten, wie auch Konzepte zur Sicherheit und Ausführungsumgebung. HTML ist dabei abhängig von weiteren Standards, um einige zu nennen: Unicode, URL, Cookies, JavaScript[1] und Document Object Model (DOM). Neben HTML und davon abhängigen Standards enthält die Open Web Platform noch weitere Standards. Dazu zählen unter anderem Cascading Style Sheets (CSS), XMLHttpRequest oder auch die File Api.
Die aktuelle HTML5 Version ist die Empfehlung vom 28.10.2014, auch wenn der HTML Standard sich fortlaufend in Entwicklung befindet und bereits ein Entwurf für HTML 5.1, vom 21.06.2016 existiert [FEL+16].
Die Verwendung von HTTP als Netzwerk-Protokoll, CSS als Gestaltungssprache und JavaScript als Skriptsprache ist dabei nicht vom HTML5 Standard vorgeschrieben [HBF+14, 2.2.2 Dependencies].

Kontext

Der Kontext [HBF+14, 5.1] eines Webbrowsers ist eine abgeschlossene Umgebung, in die Dokumente geladen werden. Typischerweise ist das ein HTML Dokument bestehend aus `body` und `head`.
Ein Webbrowser Fenster oder Reiter enthält jeweils einen eigenen Kontext, welcher durch ein jeweils eigenes `window` Objekt repräsentiert wird. Mittels `iframe` Element können darin weitere ineinander verschachtelte Kontexte erstellt werden. Die Variante eines `frame` Elements in einem `frameset` gilt dabei als obsolet.
Zum Kontext gehört auch eine session history mit dem aktuellen Verlauf, samt Zustände der Dokumente, sowie das aktive Dokument. Das aktive Dokument-Objekt ist über `window.document` erreichbar. Initialisiert wird ein Kontext mit dem leeren Dokument `about:blank`. Mit dem Initialisierungsprozess wird für das Dokument *Sandboxing* konfiguriert, dessen Referrer auf die Url des Ersteller-Dokuments gesetzt und die `origin` vom Ersteller-Dokument übernommen. Falls kein Ersteller-Kontext existiert wird die `origin` auf einen Globally Unique Identifier (GUID) und für Skripte auf `null` gesetzt.
Für Zugriffe von einem fremden Kontext, mit anderer effektiver Skript Origin, auf Objekte des eigenen `window` wird ein `SecurityError` erzeugt. Davon sind einige wenige Ausnahmen ausgenommen. Für die Ausnahmen gilt aber auch, dass jede effektive Skript Origin ihre eigenen Instanzen davon erhält.

Ein Kontext kann über seine Eigenschaft `window.name` benannt werden. Aus dem einbettenden, übergeordneten, Kontext heraus kann er folgendermaßen angesprochen werden. Entweder in Form eines Index über `window[Index]`, oder über seinen umschließenden Container, wie einem IFrame, mit `Container.contentWindow`, oder über dessen `window["KontextName"]`. Für letzteres gilt, dass der Name des Kontexts sich nicht mit Eigenschaften auf dem `window` Objekt überschneiden sollte. Schließlich gilt in JavaScript,

[1] JavaScript bezieht sich auf die ECMA-262 Spezifikation[Ecm11].

dass `window.Eigenschaft === window["Eigenschaft"]`. Die Anzahl der direkt untergeordneten Browser-Kontexte wird über `window.length` zurückgegeben.
Spezielle Referenzen auf Browser-Kontexte sind in `window.top`, `window.parent` und `window.opener` festgehalten. Das `window` Objekt des aktuellen Browser-Kontexts ist auch über die Referenzen `window.frames` und window.self erreichbar [HBF+14, 5.2].

Sandboxing Generell wird als Sandbox ein Sicherheitsmechanismus bezeichnet, der darin laufende Programme von der außerhalb laufenden Umgebung isoliert. Dabei können durchaus Ressourcen geteilt oder Schnittstellen zur Kommunikation vorhanden sein.
Innerhalb einer Sandbox ausgeführter, fehlerhafter oder schädlicher Code, kann damit nur begrenzten Schaden anrichten. Zum Beispiel nutzt der, auf Chromium basierende, Chrome Webbrowser für jedes Fenster und jeden Reiter einen eigenen Renderer Prozess [2].

Für Sandboxing innerhalb eines Kontexts ist im HTML5 Standard [HBF+14, 5.4] für jedes Dokument ein Satz von Sandboxing-Flags[3] vorgesehen. Für unverschachtelte Kontexte werden die vordefinierten Flags für Popups übernommen. Ist der Kontext verschachtelt, gelten die Attribute des übergeordneten Kontexts und IFrames, falls verwendet. Zusätzlich gibt es noch den Satz von erzwungenen[4] Sandboxing-Flags, die zum Beispiel von der *Content-Security-Policy* [WBVS15] verwendet werden.
Bei Verwendung des `sandbox` Attributs ohne Parameter gelten alle anwendbaren Beschränkungen. Dazu gehören:

- Kein Verwenden von Plugins.
- Formulare können nicht gesendet werden.
- Neue Browser-Fenster können nicht geöffnet werden.
- Mauszeiger Beschränkungen, z.B. dass der Zeiger nur innerhalb eines Elements nutzbar ist.
- Zurechnung zu neuer, eigener Origin, womit immer Same-Origin Beschränkungen gelten.
- Skripte können nicht ausgeführt werden.
- Nur die Url des eigenen Kontexts kann geändert werden.

Mit der Ausnahme von Plugins, die nativen Code außerhalb der Sandbox ausführen, können die Beschränkungen durch Angabe von Flags im `sandbox` Attribut gelockert

[2] Vgl. `https://www.chromium.org/developers/design-documents/sandbox/Sandbox-FAQ`. Letzter Zugriff: 08.11.2016.
[3] Ein Flag ist dabei vordefiniertes Attribut der Sandbox.
[4] Englisch forced.

werden. Eine Anwendung aller Ausnahmen auf dem `sandbox` Attribut sieht folgendermaßen aus: `sandbox = 'allow-forms allow-popups allow-pointer-lock allow-same-origin allow-scripts allow-top-navigation'`.

Same-Origin Policy

Weil Webbrowser Aktionen im Namen von entfernten Akteuren ausführen, können Aktionen gegen die Interessen des Benutzers oder eines Dritten durchgeführt werden. Deswegen haben webverwandte Technologien mit der Zeit zu einem gemeinsamen Sicherheitsmodell tendiert, der Same-Origin Policy [Bar11]. Dabei werden Inhalte abhängig von ihrer Quelle (englisch origin), in Form ihrer Uniform Resource Identifier (URI), voneinander getrennt behandelt. Zugriffe sind nur auf die eigenen Inhalte erlaubt. Werden fremde Inhalte, wie Skripte, eingebunden oder sind Aktionsziele von zum Beispiel Skripten, Formularen oder Links, dann vertraut man ihnen die mitgelieferten Informationen und den Umgang damit an. Eingebundene Skripte einer anderen Origin werden von dort geladen und im eigenen Kontext ausgeführt. Damit werden ihnen die eigenen Inhalte anvertraut, soweit die Möglichkeiten der vom Browser bereitgestellten Schnittstellen reichen. Allerdings unterliegt ein fremdes Skript dann den Einschränkungen auf die eigenen Inhalte.

Auf die Same-Origin Policy (SOP) existieren Angriffe, bei denen das Vertrauen des Benutzers in die Webseite missbraucht wird, und Angriffe, bei denen das Vertrauen der Webseite in den Benutzer missbraucht wird [Eck12, 3.5.3]. Bei ersterem, Cross-Site-Scripting (XSS), wird vom Angreifer Schadcode in die Webseite injiziert, der durch den Benutzer mit dem Aufruf der Webseite ausgeführt wird. Bei zweiterem, Cross-Site-Request-Forgery (CSRF), führt dem Benutzer untergeschobener Schadcode, im Namen des Benutzers, ungewollte Anfragen an eine Webseite aus.

Origin

Die Origin einer Ressource soll ihre Quelle identifizieren und besteht, für einen Benutzer sichtbar, aus Protokoll/Schema, Host und Port, zum Beispiel `https://www.w3.org:8080`. Mit der Origin werden die Sicherheitsbeschränkungen, aufgrund unterschiedlicher Origin, durch die Same-Origin Policy möglich [HBF+14][Bar11]. Dabei ist nicht die volle Url enthalten, wie beim *Referrer*[5], sondern nur die notwendige Information zur Unterscheidung der Quelle. Als Ergänzung können optional weitere Daten, wie das Zertifikat einer verschlüsselten Webseite in der Origin enthalten sein.
Abweichend zu Protokoll, Host und Port, kann die Origin auch ein eindeutiger Identifikator, zum Beispiel eine GUID sein, die für Skripte als `null` sichtbar ist. Für die Festlegung der Origin wird unterschieden zwischen Dokument- und *effektiver Skript-Origin*[6]. Die

[5] Siehe 3.1.1.
[6] Vgl. effective script origin in [HBF+14]

ursprüngliche Origin des Dokuments bleibt erhalten, wogegen sich die effektive Skript-Origin durch bestimmte Skript-Operationen ändern kann und damit die „effektive“ Origin zum Zeitpunkt nach einer Skript-Operation darstellt. Wenn im Folgenden von der Origin gesprochen wird, ist die aktuelle Origin zum Beschreibungszeitpunkt gemeint, die bei Änderung der effektiven Skript-Origin entspricht.

Die Origin eines HTML5 Dokuments wird in `document.origin` gespeichert, als auch bei Anfragen im Origin-HTTP-Header mitgesendet.

Der beim Aufruf von netzwerkbasierten Webdiensten reguläre Fall, einer Url mit serverbasierter Namensvergabe für den Hostnamen, nutzt diesen für seine Origin. Dieser Fall ist für eine weitere Betrachtung zur Problemlösung weniger interessant. Der technisch vorgesehene Standardfall ist die Vergabe einer GUID. Er tritt zum Beispiel bei data:Urls ein, die manuell eingegeben oder am Ende einer HTTP-Weiterleitung stehen. Für ein Skript ist eine GUID Origin als `null` sichtbar. Sonst greifen folgende Mechanismen:

- Sandboxing setzt die Origin auf eine GUID.
  Davon ausgenommen ist der Spezialfall von gesetztem `allow-same-origin` Flag und gleicher Origin wie der übergeordnete Kontext.
- Für data:Urls entspricht die Origin dem aufrufenden Dokument beziehungsweise der Origin des aufrufenden Skripts. Wird die data:Url manuell eingegeben, ist ihre Origin eine GUID.
- Das `about:blank` Dokument erhält die Origin vom Ersteller-Dokument. Falls keines existiert eine GUID.
- Für javascript:Urls wird die Origin vom aktiven Dokument bei Aufruf übernommen.
- Beim Verwenden eines `iframe`-Elements mit `srcdoc` Attribut wird die Origin vom `iframe`-Container verwendet.

Dass die Origin mittels `document.domain` auf eine übergeordnete Domäne geändert werden kann, ist nicht relevant für eine Problemlösung.

Die aktuelle Origin kann als HTTP-Header in Anfragen gesendet werden [Bar11, 7.3]. Praktisch wird der Origin Header bei Cross-Origin Resource Sharing (CORS)[7] Anfragen an eine fremde Origin mitgesendet [Kes14b, 5.7]. Das sind Anfragen an eine fremde Origin, die über `HTTP HEAD`, `GET` oder `POST`, mit `Content-Type application/x-www-form-urlencoded` oder `multipart/form-data`, hinausgehen oder modifizierte HTTP Header[8] verwenden [Kes14b, 4]. Explizit nicht gesendet werden soll der Origin-Header aus vertraulichen Kontexten. Wann ein vertraulicher Kontext existiert ist in „The Web Origin Concept“ [Bar11] nicht definiert.

[7] Cross-Origin Resource Sharing wird in 3.1.2 vorgestellt.

[8] Modifiziert bezieht sich auf HTTP Header, die nicht automatisch durch den Browser gesetzt werden. Einige HTTP Header dürfen manuell verändert werden, vergleiche [Kes14b, 3].

Referrer

Der Referrer eines HTML5-Dokuments, in `document.referrer`, ist eine absolute Url, die gesetzt werden kann, wenn das Dokument erstellt wird. Dabei sind der mit `#` eingeleitete Fragment Identifier, am Ende einer Uniform Resource Locator (URL), und die dem Schema folgende Benutzerinfo[9] nicht enthalten. In einer Anfrage über das HTTP Protokoll, kann der Referer [sic] HTTP-Header vom Webbrowser gesetzt werden [FR14, 5.5.2].

Bei Anfragen wird der Referrer vom aktuellen Dokument genommen. Ist die Anfrage durch ein Skript erzeugt, wird der Referrer der Skript API[10] gesendet [HBF+14, 2.6.2]. Üblicherweise wird der Referer-Header [sic] vom Browser nicht mitgesendet, wenn das Protokoll mit der Anfrage von einem verschlüsselten zu einem unverschlüsselten wechselt [FR14, 5.5.2]. Auf einem `a` oder `area` Element kann das `rel` Attribut auf `rel="noreferrer"` gesetzt werden um dem Browser zu signalisieren, den Referer-Header [sic] nicht zu senden. Falls der Referrer in `document.referrer` eine `data:Url`, `about:blank` oder der leere String `""` ist, wird der Referer [sic] Header nicht gesetzt. Vielfach wird der Referrer für file:Urls nicht gesendet[FR14, 5.5.2]. Generell ist der Referrer für Dokumente, deren Origin nicht Protokoll, Host und Port enthält leer [HBF+14, 2.6.2].

Kommunikation

Die Kommunikation eines Webbrowsers zum Abruf von HTML Seiten von einem *Web-Server*, findet klassischerweise über das HTTP Protokoll statt. Der Web-Server ist ein Dienst auf dem Host, der Web Inhalte über das HTTP Protokoll kommuniziert. HTTP unterstützt die Methoden `OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT` [FR14].

Innerhalb eines HTML-Dokuments können `GET` Anfragen, zum Beispiel durch Anklicken eines Links, und `POST` Anfragen, durch Absenden eines Formulars erzeugt werden. `GET`, als die standardmäßige HTTP-Methode zum Abrufen von Informationen, wird sowohl bei Eingabe einer Url in die Adresszeile eines Browsers, als auch beim Laden weiterer, eingebetteter Ressourcen verwendet. Solche eingebetteten Ressourcen können in HTML-Elemente wie `iframe` oder `img` über ihr `src` Attribut eingebunden werden und erzeugen bei Verwendung des HTTP-Protokolls eine entsprechende `GET` Anfrage. Die dazugehörige Antwort kann vom *Web-Server* zurückgegeben werden.
Daten können bei `GET`-Anfragen in Teile der Url wie die dafür vorgesehene Query Komponente [BLM05, 3.4], oder in den Pfad kodiert werden. Für `POST`-Anfragen können zusätzlich noch Daten in den Response-Body kodiert werden.

[9] Folgende Muster Url zur Veranschaulichung:
`schema://Benutzer:Passwort@Host:Port/Pfad?QueryKomponente#FragmentIdentifier` .
[10] Vgl. API referrer source in [HBF+14, 6.1.3.1].

Werden Skripte verwendet können die `GET` und `POST` Anfragen auch automatisiert, ohne Benutzerinteraktion, erzeugt werden. Mittels weiterer API Schnittstellen können auch die anderen HTTP-Methoden genutzt werden.

Der Fetch Standard [Kes16] zur Vereinheitlichung von Datenabrufen existiert Stand August 2016 als „Living Standard“ bei der Web Hypertext Application Technology Working Group (WHATWG). Dessen Ziel ist es den HTML5 fetch Algorithmus [HBF+14, 2.6], CORS [Kes14b] und das Web Origin Konzept [Bar11] zu vereinheitlichen. Allerdings wird auch bei einem Fetch entweder die Origin im Header mitgesendet oder die Antwort wird entsprechend leer zurückgegeben, um der SOP Rechnung zu tragen. Aus den genannten Gründen wird der Fetch Standard nicht für weitere Betrachtungen herangezogen.

XMLHttpRequest spezifiziert eine API [Kes14a], mit der skriptgesteuert Daten zwischen Client und Server übertragen werden können. Damit können alle HTTP-Methoden verwendet, zusätzliche Header festgelegt und auf festgelegte Ereignisse des Ladevorgangs reagiert werden.

XMLHttpRequest ist wichtiger Bestandteil von Asynchronous JavaScript and XML (AJAX) [Gar05] und wird für dynamische Webseiten genutzt, die Informationen im Hintergrund mit dem Server austauschen.

Web Messaging Web Messaging oder cross-document messaging [Hic15] spezifiziert zwei Mechanismen, mit denen Browser-Kontexte miteinander kommunizieren können. Dabei verbleiben die Daten innerhalb des Webbrowsers. Ein Kontext kann dabei Daten an einen anderen senden, wenn dieser auf dem `message` Event lauscht. Mit dem MessageChannel wird dieses Prinzip so erweitert, dass Nachrichten über Anschlüsse des MessageChannel Kanals gehen und nicht mehr an den Kontext direkt.

WebSocket Das WebSocket Protokoll [FM11] bietet einen einfachen bidirektionalen Nachrichtenkanal vom Browser zu einem Server und ähnelt einer fürs Web angepassten TCP Verbindung. Tatsächlich baut WebSocket auch auf TCP auf, fügt allerdings noch ein Origin-basiertes Sicherheitsmodell, eine namensbasierte Zuordnung von Verbindungen, eigene Datenframes und handshakes zum Eröffnen und Schließen der Verbindung ein. Ziel ist es die Möglichkeiten eines TCP Kanals für Web-Technologien [Hic12] verfügbar zu machen. Dafür wird ein entsprechender WebSocket Server benötigt, da im Gegensatz zum XMLHttpRequest nur für den handshake HTTP genutzt wird.

Server-Sent Events Mit Server-Sent Events [Hic14] kann der Browser (push) Nachrichten vom Server über HTTP empfangen, ohne selbst den einzelnen Nachrichtenaustausch zu initiieren. Es muss nur einmalig ein Empfänger für den entsprechenden Sender eingerichtet werden.

WebRTC WebRTC [BBJ16][Alv16], für Web Real-Time Communication, ist eine Browser-Schnittstelle, die Browser-zu-Browser Kommunikation in Echtzeit für Sprachanrufe, Video Chat und Peer to Peer Datenaustausch ermöglicht. Zum Aufbau der Verbindung und dem Austausch von Kontrollnachrichten kann ein signaling Server verwendet werden, der per XMLHttpRequest oder Web Sockets angesprochen werden kann. Für eine Peer to Peer Kommunikation über Netzwerke mit Network adress translation (NAT) hinaus, können entsprechende Protokolle wie ICE, STUN oder TURN[11] genutzt werden.

3.1.2 Cross-Origin Kommunikation

Innerhalb eines geladenen HTML-Dokuments und mit dessen Herkunftsserver kann, entsprechend der Same-Origin Policy, frei kommuniziert werden. Für einen Nachrichtenaustausch mit einer fremden Origin, können nicht beschränkte Kommunikationstechniken oder eingeführte Erweiterungen, die mögliche Sicherheitsbedenken behandeln, genutzt werden. Im Folgenden sollen entsprechende Techniken vorgestellt werden.

Proxy Der Web-Server kann als Proxy für einen Dritten agieren und Anfragen, sowie Antworten von Webbrowser oder Drittem, an den jeweils anderen weiterleiten.

Die Proxy-Funktion kann auch nur ins Dokument eingebettete Elemente betreffen, sodass eigene Inhalte parallel dazu übertragen werden können. Ob der Ausgestaltung der Kommunikationsmethoden, besteht die freie Wahl.
Ein klassischer Ansatz wäre eine Anfrage an den Same-Origin Web-Server per HTTP `GET`, mit in die Query Komponente kodierten Daten. Der Same-Origin Web-Server leitet die Anfrage an den Web-Server der fremden Origin weiter, wertet dessen Antwort aus und liefert dem Benutzer ein HTML-Dokument mit Inhalten der fremden Origin aus.

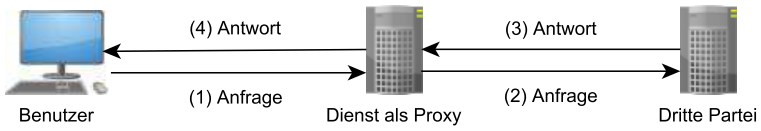

Abbildung 3.1: Nachrichtenaustausch mit dem Dienst als Proxy.

HTTP GET und POST mit HTML-Elementen Einige HTML-Elemente erzeugen unter bestimmten Bedingungen HTTP `GET` und `POST` Anfragen, die auch an eine fremde Origin gerichtet sein können. Eine Detaillierte Auflistung erfolgt in der Tabelle 3.1.
Für `GET` Anfragen können Daten in die Url kodiert werden, speziell dafür gedacht ist

[11] Session Traversal Utilities for NAT (STUN) ist ein Protokoll zur Adresserkennung (der öffentlichen IP-Adresse) und NAT Klassifizierung. Traversal Using Relays around NAT (TURN) nutzt eine dritte Partei, die Nachrichten der beiden weiterleitet. Interactive Connectivity Establishment (ICE) nutzt STUN und TURN, um einen IP Kanal zwischen den Kommunikationspartnern zu ermöglichen.

Element	Url-Attribut	Aufruf	Effekt
a	href	`click` Event	Die Url wird per `GET` abgerufen und der Inhalt angezeigt.
form	action	`submit` Event	Formulardaten werden per `GET` oder `POST`, definiert im `method` Attribut, an die Url gesendet und das erhaltene Ergebnis angezeigt.
img	src	bei einbetten ins Dokument	Auf die `src` Url wird ein `GET` ausgeführt und das erhaltene Bild im unterstützten Format im Dokument angezeigt.
iframe	src	bei einbetten ins Dokument	Auf die `src` Url wird ein `GET` ausgeführt und der Inhalt in einem eigenen Browser Kontext zur Verfügung gestellt.
embed	src	bei einbetten ins Dokument	Auf die `src` Url wird ein `GET` ausgeführt und mit dem entsprechenden Plugin ausgeführt.
object	data	bei einbetten ins Dokument	Auf die Url im `data` Attribut wird ein `GET` ausgeführt. Je nach Inhalt wird ein Plugin darauf ausgeführt, das Bild angezeigt oder der Inhalt in einen neuen Browser Kontext geladen.
video/audio	src oder source/track Element	bei einbetten ins Dokument	Auf die Url im `src` Attribut, oder einem optionalen `source` oder `track` Element mit `src` Attribut, wird ein `GET` ausgeführt.
script	src	bei einbetten ins Dokument	Auf die Url im `src` Attribut wird ein `GET` ausgeführt. Ist der Inhalt ein ausführbares Skript, so wird es im Namen des einbindenden Kontexts ausgeführt.

Tabelle 3.1: HTML Elemente mit HTTP `GET`/`POST` Verhalten. [HBF+14, 4.]

die Query Komponente der Url. `POST` Anfragen liefern die Formulardaten kodiert als `application/x-www-form-urlencoded` oder `multipart/form-data` im Nachrichten Body mit.

Besitzt ein Element das `target` Attribut, so kann darüber, statt dem standardmäßigen aktuellen Browser Kontext, ein anderer Ziel-Kontext festgelegt werden. Optionen sind `_blank`, `_self`, `_parent`, `_top` oder der Name eines Browser Kontexts. `_blank` bezieht sich auf ein neues Browser-Fenster mit neuem Browser Kontext, `_self` auf den aktuellen Browser-Kontext, `_parent` in einem verschachtelten Browser Kontext den darüber liegenden Browser Kontext, und `_top` in einem verschachtelten Browser Kontext den obersten Browser Kontext.

In Kombination mit Skripten sind vor allem `a`, `form` und `iframe` interessant.
Mit `a` lassen sich über die Url Daten an eine fremde Origin übergeben, die Inhalte in

einem neuen Kontext anzeigen kann. Das Problem stellt dann allerdings der Zugriff auf diesen neuen Browser-Kontext dar.
Mit dem `form` Element können (Formular-)Daten an eine fremde Origin übermittelt werden, die als Antwort eigene Inhalte in einem neuen Browser Kontext ausliefern kann. Die Anfrage kann mittels `POST` deutlich mehr Daten als eine `GET` Anfrage enthalten, deren Länge von Browser und Web-Server beschränkt ist. Wie bei `a` besteht die Schwierigkeit im Zugriff auf diesen neuen Browser-Kontext. Das `iframe` Element stellt einen eigenen Browser Kontext und ermöglicht das Einbetten eines weiteren HTML Dokuments. Dieser Sonderfall wird in 3.1.2 weiter behandelt.

Mittels `embed` und `object` lassen sich Browser-Plugins in einem eigenen Kontext einbinden. Allerdings sind Plugins, laut Problemstellung in 1.2, nicht gewünscht. Eine Antwort auf ein erzeugtes `GET` ist nur über andere Wege möglich. Bindet `object` ein HTML-Dokument in einen neuen Browser Kontext ein, wird dieses in einen dafür erstellten, separaten, neuen Kontext geladen.

Um mit dem `script` Element über ein `GET` hinaus Inhalte auszutauschen, können Inhalte in den Skript Quellcode integriert werden. In das Skript können Daten eingebettet werden, die bei Ausführung an eine vorher definierte Funktion als Parameter übergeben werden. Diese Rücksprung-Funktion innerhalb der eigenen Skripte ist für eine weitere Verarbeitung der erhaltenen Daten zuständig. Die Technik wird JavaScript Object Notation with Padding (JSONP) genannt und im Folgenden (Siehe 3.1.2) behandelt.

Mit dem Einbinden von Inhalten fremder Origin über das `src` Attribut von `img`, `video` und `audio`, erfolgen `GET` Anfragen an die entsprechenden Urls [HBF+14, 4.7]. Darüber lassen sich Daten in eine Richtung übertragen. Eine Antwort auf die Nachricht müsste aber über einen anderen Weg übertragen werden. Der Grund ist die Same-Origin Policy und ihre Trennung von Inhalten unterschiedlicher Origin.
Mögliche Zugriffe auf die empfangenen Inhalte sind mit der Verwendung des `crossorigin` Attributs und CORS (Siehe 3.1.2) möglich.
Nur mit CORS lassen sich `img` Elemente von einer fremden Origin auf einen eigenen Canvas zeichnen und mittels `.toDataURL()`, `.toBlob()`, `.getImageData()`, `.getImageDataHD()` in verarbeitbares Datenmaterial umwandeln, mit dem in Bilder kodierte Daten ausgelesen werden können. Für `audio` und `video` Elemente geht es um den Zugriff auf Untertitel.

JavaScript Object Notation with Padding (JSONP) Werden `script` Elemente in ein Dokument eingefügt, so wird das Skript im Kontext des Dokuments und mit dessen Origin ausgeführt, unerheblich von welcher Origin das Skript geladen wird. Daten für einen Dritten können in die Url vom Skript kodiert werden und der Dritte stellt seine Antwort in dem eingefügten Skript als JavaScript Objekt zur Verfügung. Damit die im Antwort-Skript-Objekt enthaltenen Daten übergeben werden können, braucht es noch einen Aufruf auf eine bereits vorhandene Rücksprung Funktion im Kontext des Dokuments, an welche das Antwort-Objekt übergeben wird. Das Umschließen der JSON

Daten mit der Rücksprung Funktion ist das „Padding“ in JSONP. Hat die Rücksprung-Funktion die JSON-Daten erhalten, können sie weiter verarbeitet werden.
Die Gefahr besteht darin, dass ein eingebundenes Skript beliebige JavaScript Operationen ausführen kann und damit auch unerwünschte oder schädliche. Zum Beispiel können Daten aus dem eigenen Kontext an Dritte gesendet werden. Dafür können Techniken aus Tabelle 3.1 verwendet werden. Deswegen ist es wichtig nur von vertrauenswürdigen Dritten Skripte einzubinden.

IFrame (Inline Frame) Das `iframe` Element, stellt einen eingebetteten Browser-Kontext zur Verfügung und kann darin ein Dokument mit fremder Origin laden. Durch den getrennten Kontext in Verbindung mit der SOP werden Zugriffe auf den jeweils anderen Kontext beschränkt. Eine Einschränkung darüber hinaus ist mit dem in 3.1.1 vorgestellten Sandboxing möglich.
Wegen der Same-Origin Policy (SOP) entfällt die Variante, dynamisch `iframe` Elemente zu erstellen, Daten in die Url zu kodieren und das erzeugte Antwort Dokument mittels Skript auszulesen. Zugriffe auf die Inhalte der fremden Origin sind nicht erlaubt.
Soweit nicht durch Sandboxing (Siehe 3.1.1) beschränkt, können innerhalb des IFrame, und damit dessen Kontext, Skripte ausgeführt werden. Aber es können Techniken wie das bereits vorgestellte Web Messaging in 3.1.1 verwendet werden.
Eine weitere Technik ist Fragment Identifier Messaging (FIM), auch „Hashtransport“ genannt. Der Fragment Identifier ist der hintere Teil einer Url, nach Schema, Host, und Pfad, der mit einem `#`, englisch hash, eingeleitet wird [BLM05, 3.5 Fragment]. Ändert sich nur der Fragment Identifier, so erfolgt kein neuer Abruf des Dokuments. Da die URL dem IFrame erstellenden Kontext bekannt ist, kann sie um einen Fragment Identifier erweitert werden. Dabei kann `iframeObject.src` oder `iframeObject.contentWindow.location.href` zum Festlegen der Url verwendet werden. Der Kontext im IFrame erhält daraufhin ein `hashchange` Event auf seinem `window` Objekt und kann die geänderte Url aus dem Event in `EventObject.newUrl` oder aus `window.location.hash` auslesen. Dabei kann der Absender der Nachricht nicht erkannt werden. Wegen Längenbeschränkungen der Url, ist die darüber ausgetauschte Nachrichtengröße begrenzt. Ist dem Kontext im IFrame der sendende Kontext und dessen Url bekannt, kann eine Antwort-Nachricht über den Hash Teil der Url gesendet werden. Ist der sendende Kontext, der dem IFrame überliegende, so kann `parent.location.href` mit einem entsprechendem Hash Argument belegt werden. Mit einem weiter verschachtelten IFrame Kontext, können auch Nachrichten gesendet werden.
Solange eingebettete Kontexte Skripte ausführen können, besteht trotz Trennung in eigene Browser-Kontexte, die durch die SOP geschützt sind, die Möglichkeit Schaden zuzufügen. Eine Möglichkeit wäre die URL der überliegenden Seite auf eine eigene, ähnlich aussehende zu ändern und den Benutzer auf der weitergeleiteten Seite dazu zu verleiten Daten preiszugeben.

Cross-Origin Resource Sharing (CORS) Trotz Same-Origin Policy (SOP) als Sicherheitsmechanismus, gibt es Situationen, in denen es erwünscht ist, Inhalte von einem vertrauenswürdigen Dritten einzubinden. CORS [Kes14b] ist eine Technik, die einen solchen Aufruf als Ausnahme zur SOP möglich macht. Dazu muss der Dritte bei Auslieferung seiner Inhalte den HTTP-Header `Access-Control-Allow-Origin` auf mindestens die anfragende Origin oder `*`, für eine generelle Erlaubnis, gesetzt haben. Werden über HTTP `GET`, `HEAD` und `POST` mit Standard Request-Header inklusive Standard `Content-Type` hinaus, zusätzliche Request-Header oder andere HTTP-Methoden verwendet, werden zusätzliche Response-Header und ein weiterer HTTP `OPTIONS` Request vor der eigentlichen Anfrage nötig.

Mit CORS sind Zugriffe mittels XMLHttpRequest auf fremde Origins, vorbehaltlich ihrer Mitarbeit, möglich. Damit sind auch Origin übergreifende WebSocket Verbindungen und Server-Sent Events verwendbar.
Ohne CORS wird beim Abruf von Ressourcen einer fremden Origin diese entweder als „tainted“ markiert, oder der Aufruf schlägt fehlt. Wird CORS verwendet und die Anfrage war erfolgreich, gelten die abgerufenen Ressourcen als Same-Origin und sind damit verwendbar, wie Ressourcen der abrufenden Origin. Für eine `data:URL` wird kein Verhalten spezifiziert und ein nicht weitergeleitetes `about:blank` fällt unter die gleiche Origin [HBF+14, 2.6.7 CORS-enabled fetch].

Web Messaging Web Messaging [Hic15], bereits als Kommunikations-Methode eines Webbrowsers in 3.1.1 eingeführt, dient der Kommunikation zwischen zwei Browser-Kontexten. Dabei funktioniert Web Messaging unabhängig von der Origin der Kontexte und ist deswegen auch für Cross-Origin Kommunikation geeignet.

Mit der Referenz auf ein `window` Objekt, können an dieses, mittels `.postMessage(message, targetOrigin [,transfer])`, Objekte gesendet werden.

- `message` ist ein Objekt, mit den zu sendenden Daten.
- `targetOrigin` ist die Origin, an die gesendet wird. Für eine beliebige Origin kann `*` verwendet werden.
- `transfer` ist optional und überträgt Objekte im materiellen Sinn, so dass diese danach nicht mehr an der Quelle nutzbar sind.

Das referenzierte `window` erhält daraufhin ein `message` Event mit dem Aufbau `{data, origin, lastEventId, source, ports}`. Das gleiche `MessageEvent` wird auch für Server-Sent Events und Websockets verwendet.

- `data` enthält das übertragene Daten-Objekt.
- `origin` enthält die Origin vom Sender-Dokument.
- `lastEventId` wird für Server-Sent Events genutzt und enthält die letzte gesehene Event ID.

- `source` enthält eine Referenz auf das `window` Objekt des Senders.
- `ports` mit einem optionalen Array von MessagePorts.

Der Sender sollte eine Empfänger-Origin angeben, weil sonst nicht sichergestellt werden kann, dass die Daten nicht doch ein Anderer erhält. Ein Empfänger sollte die Origin des Senders überprüfen um die Angriffsfläche klein zu halten. Eine zusätzliche Prüfung der erhaltenen Daten ist trotzdem ratsam.

Plugins Ein HTML-Dokument kann mit `embed` oder `object` Plugins einbinden, innerhalb derer nativer Programm-Code ausgeführt wird, der nicht mehr unter Kontrolle des Webbrowsers und seiner Sicherheitsmechanismen steht. Ein Plugin könnte zum Beispiel eine Netzwerk-Socket Verbindung zu einem beliebigen Dritten aufbauen.
Wegen der Weiterentwicklung des HTML Standards und den Implementierungen, sind viele Funktionen nativ im Webbrowser enthalten und es werden immer weniger Plugins benötigt.
Bekannte Plugins sind zum Beispiel Java oder Flash, die jeweils eigene Sicherheitsmechanismen mitbringen. Manche Webbrowser lassen ihre Plugins in einer geschützten Sandbox laufen und erhöhen so die Sicherheit etwas, zum Beispiel können Zugriffe aufs Dateisystem unterbunden werden.
Wegen der Anforderung aus der Problemstellung in 1.2, Standardsoftware zu verwenden, fallen Plugins, die zusätzlich installiert werden müssten, für eine Lösung aus.

WebRTC WebRTC [BBJ16][Alv16], bereits bekannt als Kommunikationstechnik des Webbrowsers aus 3.1.1, kann in einer `RTCPeerConnection`, über einen zu erstellenden `RTCDataChannel`, von Webbrowser zu Webbrowser, und damit Origin übergreifend, Datenpakete austauschen.

Um eine Verbindung zwischen zwei Webbrowsern mit WebRTC herzustellen, benötigen beide eine `RTCPeerConnection`. Um darüber eine gemeinsame Sitzung auszuhandeln, wird das Session Description Protocol (SDP) gemäß dem Offer/Answer Modell genutzt [RS02].

1. Dabei erstellt einer der beiden Webbrowser, im Folgenden Initiator genannt, per `createOffer` eine SDP Offer-Nachricht und sendet diese zur Gegenseite. Die Nachricht enthält die initiale Sitzungskonfiguration, die für den Initator mit `.setLocal Description` als lokale SDP Konfiguration festgelegt wird.
2. Über einen nicht definierten Kanal wird die Offer-Nachricht zur Gegenseite übertragen.
3. Die Gegenseite kann mit der Offer-Nachricht ihre entfernte SDP Konfiguration mit `.setRemoteDescription` festlegen, daraus über `createAnswer` eine SDP Answer-Nachricht erzeugen und ihre lokale SDP Konfiguration mit `.setLocalDescription` festlegen.

4. Über einen nicht definierten Kanal wird die SDP Answer-Nachricht zum Initiator gesendet.
5. Der Initiator legt die SDP Antwort als entfernte SDP Konfiguration mit `.setRemote Description` für sich fest.

[BBJ16]

WebRTC bietet mit der peer-to-peer Data API[BBJ16, 6.] die Möglichkeit, einen `RTCData Channel` in einer `RTCPeerConnection` nutzen zu können.
Dabei ist zu beachten, dass der erste `RTCDataChannel` in einer `RTCPeerConnection` ein aushandeln der `RTCPeerConnection`, mittels im Absatz zuvor beschriebenem Offer/Answer Modell erfordert. Wird der erste `RTCDataChannel` aber vom Initiator vor dem `createOffer` erstellt, ist kein extra Aushandeln mehr erforderlich.
Ein `RTCDataChannel` wird vom Initiator mit `createDataChannel` angelegt. Der Empfänger erhält dann ein `datachannel` Event mit Kanal `channel`, auf dem über das `MessageEvent message` eingehende Nachrichten ankommen und mit `send(data)` gesendet werden können. Das `MessageEvent` mit dem Aufbau `{data, origin, lastEventId, source, ports}`[12], hat das gleiche Format wie bei Web Messaging, Server-Sent Events oder Websocket.

Zusammenfassung Die verschiedenen vorgestellten Techniken und Ansätze decken unterschiedliche Ebenen und Rahmenbedingungen ab, unter denen aus einem Browser-Kontext heraus mit einem Dritten kommuniziert werden kann.
Neben der klassischen Proxy Lösung, die den Web-Server die Kommunikation vermitteln lässt, werden die anderen Techniken direkt vom Webbrowser verwendet. Angefangen von der Nutzung HTML-Element-spezifischer Kommunikation, die teilweise keinen vom Webbrowser auswertbaren Antwortkanal anbietet, folgen JSONP und Varianten auf Basis des `iframe` Elements. JSONP und IFrame Kommunikationsmechanismen sind nicht für eine Cross-Origin Kommunikation konzipiert worden und bergen Sicherheitsrisiken. Gleiches gilt für ein einbetten von fremdem Code über ein Plugin. Dagegen sind CORS und Web Messaging für Cross-Origin Kommunikation entwickelt worden. WebRTC ist zwar für Audio- und Video-Kommunikation von Webbrowser zu Webbrowser konzipiert, besitzt aber auch einen für Datenaustausch spezifizierten und geeigneten Kanal.

3.1.3 Unverknüpfbare Cross-Origin Kommunikation

Neben der Betrachtung der vorgestellten Techniken zur Cross-Origin Kommunikation hinsichtlich Nicht-Weitergabe der Dienst-Identität an den Dritten im Sinne der Problemstellung aus 1.2, sollen Techniken zur Wahrung dieser Unverknüpfbarkeit durch den Dritten vorgestellt werden. Ausgenommen sind Plugins, weil sie der Anforderung, aus der Problemstellung in 1.2, Standardsoftware zu verwenden, nicht nachkommen.

[12] Vgl. `MessageEvent` Web Messaging in 3.1.2.

Ursächlich für eine Verknüpfbarkeit von Benutzer zu Dienst durch den Dritten sind die HTTP-Header Origin und Referer [sic], die vom Browser des Benutzers an den Dritten gesendet werden.

Cross-Origin Techniken

Proxy Von den Cross-Origin Kommunikationstechniken aus 3.1.2, erfüllt eine Proxy Lösung nicht die nötige Unverknüpfbarkeit von Benutzer und Dienst durch den Identity Provider, weil der Web-Server des Service-Providers mit dem Identity-Provider kommuniziert und diesem damit seine Identität preisgibt. Ein zusätzlicher Proxy zwischen Service Provider und Identity Provider, würde eine Weitergabe der Identität verhindern, allerdings müsste dann sichergestellt werden, dass der dazwischen geschaltet Proxy selbst diese Daten nicht weiter geben kann.

HTTP GET und POST mit HTML-Elementen Mit HTML-Elementen lassen sich HTTP GET und POST Anfragen senden. Diese sind hauptsächlich als Steuerelemente oder anderweitige Beschreibung der Benutzeroberfläche gedacht und bieten keinen fertigen Kommunikationskanal. Die Same-Origin Policy sorgt dafür, dass Zugriffe auf die angefragten Inhalte der fremden Origin nicht ohne weiteres möglich sind.
Um trotzdem Nachrichten auszutauschen, kann eine Lösung darin bestehen, den Benutzer solche Inhalte übertragen zu lassen. Das kann durch Kopieren und Einfügen von Zeichenketten von einem Webbrowser Fenster in ein anderes passieren, oder durch herunterladen einer Datei aus einem Webbrowser Fenster und erneutes hochladen in einem anderen Webbrowser Fenster.
Eine andere Variante für eine Rückkommunikation besteht, wenn eine eingebettete Antwort Skripte verwenden kann. Dann könnte über andere skriptbasierte Techniken kommuniziert werden.

JSONP Mit JSONP, welches das `script` Element nutzt, kann ein automatisierter Anfrage- und Antwort Kanal etabliert werden. Das Problem ist, dass das Script ausgeführt wird. Damit werden alle darin enthaltenen Befehle ausgeführt. Es ist auch nicht möglich dies zu unterbinden und nur den Inhalt des Skripts auszulesen. Damit kann der Identity Provider beliebige Skript-Befehle einfügen und sich so auch die Url der geladenen Seite zum Beispiel per XMLHttpRequest (XHR) zusenden. Die Identität des Service Providers ist damit nicht geschützt.

IFrame Ein mit einem `iframe` Element eingebettetes Dokument fremder Origin agiert in seinem eigenen Kontext und ist durch die SOP vom einbettenden Kontext getrennt. Mit Sandboxing kann der eingebettete Kontext darüber hinaus in seinen Rechten eingeschränkt werden.
Wird dem IFrame zugehörigen Kontext mit Sandboxing das Recht genommen Skripte

auszuführen, kann dessen Kontext nicht mehr automatisiert mit dem einbettenden Kontext kommunizieren. Damit wäre analog zu „HTTP GET und POST mit HTML Elementen“ nur eine Kommunikation an den eingebetteten Kontext möglich.
Erhält der eingebettete Kontext das Recht Skripte ausführen zu können, bestehen weitere Kommunikationsmöglichkeiten über CORS, Web Messaging oder WebRTC. Diese werden in den folgenden Abschnitten behandelt.
Fragment Identifier Messaging (FIM) funktioniert ohne Preisgabe der Url des einbettenden Kontexts an den eingebetteten Kontext nur in Richtung des eingebetteten Kontexts.

CORS Mit der Zustimmung der fremden Origin über den `Access-Control-Allow-Origin` HTTP-Header, lockert der Webbrowser die SOP für diese fremde Origin. Damit kann zum Beispiel mittels XHR, WebSocket oder Server-Sent Events mit dieser Origin kommuniziert werden. In einer Anfrage an eine fremde Origin wird die effektive Skript Origin im `Origin` HTTP Header gesendet. Damit wird die aktuelle Origin an den Dritten weitergegeben und somit im Falle der Authentifizierung, die Identität des Service Provider an den IdentityProvider.

Web Messaging Die mit Web Messaging gesendeten Nachrichten eines Browser-Kontexts an einen anderen Browser-Kontext enthalten im `message` Event die Origin vom Absender-Kontext. Der Empfänger einer Web Messaging Nachricht ist bereits durch eine Referenz auf seinen Browser-Kontext adressierbar und muss nicht bekannt sein, solange als Ziel `*` angegeben wird. Beispielhaft sieht der Aufruf dafür folgendermaßen aus: `receiverContext.postMessage(message,*);`. Damit ist Web Messaging mit dem Kontext des ServiceProvider als Empfänger interessant. Denn die empfangene Origin des IdentityProvider ist dem ServiceProvider bereits bekannt.

WebRTC Wird zur Koordinierung der Verbindungen zwischen den WebRTC Clients ein zentraler Signaling Channel verwendet, so besteht dabei die Schwierigkeit einen Kanal aufzubauen, der die aktuelle Origin nicht weitergibt. Wird, wie im WebRTC Standard vorgeschlagen, XMLHttpRequest oder Web Sockets genutzt, wird die Origin an den Signaling Server weitergegeben. Damit verschiebt sich das Problem wie in der Proxy Lösung auf eine weitere Partei, den Signaling Server.
Das gilt auch, wenn ein ICE, STUN oder TURN [13] Server verwendet wird, der hinter einer NAT Firewall befindlichen WebRTC Clients eine Kommunikation mit anderen WebRTC Clients hinter einer NAT Firewall ermöglicht. Allerdings kann das in Zukunft kein Problem mehr darstellen, wenn mit IPv6 alle Geräte ihre eigene eindeutige Adresse besitzen und aus diesem Grund kein NAT mehr notwendig ist.

Wird das Signaling mit anderen Techniken realisiert, bei denen die Identität nicht an den Dritten weitergegeben wird, so kann mit WebRTC, wie in 3.1.2 beschrieben, ein bidirektionaler Datenkanal vom Benutzer, in seinem aktuellen Kontext, zum Dritten

[13] Siehe Fußnote 11 zu WebRTC in 3.1.1.

aufgebaut werden. Das beim Empfänger ausgelöste `MessageEvent`, mit dem Aufbau `{data, origin, lastEventId, source, ports}`[14], hat `origin = ""`, `lastEventId = ""` und `source = null`. Aus diesem Grund wird der Kontext nicht weitergegeben, in dem sich der `RTCDataChannel` Sender befindet. Beim Verbindungsaufbau mit SDP[RS02] wird im von der Gegenseite empfangenen `description.sdp` Objekt eine Origin mitgeliefert. Diese bezieht sich auf den Host des Webbrowsers und nicht den aktuellen Browser-Kontext [HJP06, 5.2 Origin].

Referrer und Origin anonymisieren

Um die geforderte Unverknüpfbarkeit von Benutzer und Dienst durch den Dritten zu gewährleisten, lassen sich die vorgestellten Kommunikations-Techniken mit anderen Techniken oder Verhaltensweisen eines Webbrowsers kombinieren. Sie werden in diesem Abschnitt vorgestellt.

Bereits aus 3.1.1 ist bekannt, dass die Origin im Falle von Sandboxing ohne `allow-same-origin` und im Falle von `about:blank` ohne erstellenden Kontext `null` ist. Nach „The Web Origin Concept“ [Bar11, 7.3] ist der Origin HTTP-Header für Anfragen aus Datenschutz sensiblen Kontexten `null`. Eine solcher Kontext wird aber nicht definiert. Für den leeren Referrer ist in 3.1.1 bereits der Fall vom Übergang von einem verschlüsselten in einen unverschlüsselten Kontext beschrieben. Ein Praktisches Beispiel dafür wäre der Wechsel von HTTPS auf HTTP. Eine weitere Möglichkeit für einen leeren Referrer ist die Verwendung von `rel="noreferrer"` auf `a` und `area` HTML-Elementen. Der Referrer ist auch leer, wenn die Origin nicht aus Protokoll, Host und Port besteht, wie zum Beispiel bei einer GUID, file:Uri oder data:Uri.

Meta Anweisungen Das `meta` Element dient der Festlegung von Metadaten, analog zu einem HTTP-Header.

Über das `http-equiv` Attribut, in Verbindung mit dem `content` Attribut als Parameter, erfolgt eine Aktualisierung auf das Aktuelle oder gegebene Dokument nach Ablauf der angegebenen Zeit in Sekunden. Dazu folgendes Beispiel zur Weiterleitung auf `page.html` nach 10 Sekunden: `<meta http-equiv="refresh" content="10; URL=page.html">`. Damit kann ohne Mitwirkung von Skript, aus dem Kontext des Dokumentes, eine Weiterleitung veranlasst werden.

Nicht im HTML5 Standard vom W3C [HBF+14] enthalten, und in der Entwurfsphase, aber bereits implementiert, ist die Referrer Policy[ES16], zur Festlegung vom Referrer Verhalten eines Dokuments. Auf dem `meta` Element kann das `name` Attribut Metadaten für das Dokument festlegen. Eine Referrer Policy kann über `name="referrer"` und der Referrer Policy im `content` Attribut festgelegt werden. Zur Wahl stehen folgende Policy Konfigurationen: `"no-referrer"`, `"origin"`, `"no-referrer-when-downgrade"`,

[14] Vgl. `MessageEvent` Web Messaging in 3.1.2.

"origin-when-crossorigin" und "unsafe-URL". Veraltet ist dagegen die Verwendung von "always", "default" und "never". Alternativ kann eine Referrer Policy mittels Referrer-Policy HTTP Header festgelegt werden.

Mittels Content Security Policy (CSP) [WBVS15] kann eine Policy über das meta Element festgelegt werden. Alternativ kann eine Definition im HTTP Response Header, bei Auslieferung des Dokuments vorgenommen werden. Eine Formulierung erfolgt mit http-equiv="Content-Security-Policy" und der Policy Beschreibung im content Attribut. Eine Policy beschreibt dabei eine Menge von erlaubtem oder beschränktem Verhalten und enthält eine Menge von Direktiven und eine Aktion. Eine Direktive stellt dabei eine Regelbeschreibung dar. Eine Aktion ist in der Regel die Durchsetzung der Policy, kann aber auch nur eine Meldung bedeuten. Allerdings unterstützt das meta Element nur den Modus Durchsetzung, der nicht angegeben werden muss.
Policies lassen sich in Anfrage Direktiven, Dokument Direktiven, Navigations Direktiven und Meldungs Direktiven unterscheiden.
Anfrage Direktiven können die Quelle von weiteren Anfragen beschränken, zum Beispiel für ein Nachladen von Skripten oder durch XMLHttpRequest.
Eine Dokument Direktive kann base-uri, plugin-types, sandbox und disown-opener sein. Mit base-uri kann die Uri des base Elements beschränkt werden. Plugin-types kann den erlaubten MIME Typ eines Plugins einschränken. Sandbox und disown-opener sind ausschließlich einer Verwendung im HTTP Response Header vorbehalten. Ersteres funktioniert wie Sandboxing, in 3.1.1, für den Browser-Kontext des Dokuments. Zweiteres entzieht dem Kontext der window.open(...) aufruft die damit einhergehenden Rechte.
Die Navigations Direktiven form-action und frame-ancestors beschränken den Aufruf bestimmter Ressourcen durch Navigation. Das betrifft einerseits mit form-action das Ziel von Formularen. Andererseits kann mit frame-ancestors eine Beschränkung der einbettenden Kontexte vom eigenen Dokument, zum Beispiel in einem IFrame, vorgenommen werden.
Die nicht mit dem meta Element nutzbaren Meldungs Direktiven legen fest, an wen die Meldung ausgeliefert werden soll.

Die Anfrage Direktive script-src kann unter anderem Scripte auf ihre Integrität und damit Veränderung prüfen. Wenn die Content-Security-Policy auf script-src gefolgt von einer Liste von "Hashverfahren-Hashwert" gesetzt ist, kann in einem script Element das Attribut integrity auf einen oder mehrere Werte von aufgelisteten "Hashverfahren-Hashwert" gesetzt werden.

Proxy zur Weiterleitung Die Variante, einen Proxy ausschließlich zum anonymisieren von Referrer und Origin zu verwenden, scheidet wegen dem Ziel der Arbeit in 1.3 aus, eine Lösung ausschließlich mit den drei Beteiligten aus Abbildung 1.1 zu finden.

3.1.4 Zusammenfassung

Die Kommunikationstechniken wurden erst im Hinblick auf einen Cross-Origin Einsatz betrachtet und danach im Spezielleren auf ihre zusätzliche Eignung zur Nachrichtenübermittlung für eine Lösung der Problemstellung in 1.2. Gemäß der Anforderung der Problemstellung soll der Identity Provider aus der stattfindenden Kommunikation nicht die Identität vom Service Provider lernen können. Dahingehend wurde auch das spezifizierte Verhalten noch einmal festgehalten und potenziell nützliche Techniken vorgestellt.

Neben dem beschriebenen, spezifizierten Verhalten eines Webbrowsers, kann es unspezifiziertes oder nicht genau spezifiziertes Verhalten geben. Außerdem können sich die Implementationen von verschiedenen Entwicklern unterscheiden. Für eine Implementation der Kommunikation, basierend auf den untersuchten Kommunikationstechniken, müsste aber eine Möglichkeit gefunden werden, die mit gängigen Browsern funktioniert, damit diese weder Referrer noch Origin weiter geben.
Vielversprechend dafür scheinen mittels IFrame eingebettete Browser Kontexte unter Verwendung von Sandboxing, `about:blank` und data:Uris.
Neben direkter Kommunikation, aus dem aktuellen Browser Kontext heraus, wäre ein Ansatz zum Beispiel ein Proxy-Kontext zu schaffen, der Origin und Referrer `null` hat und Nachrichten über Web Messaging entgegennimmt und mit `null` Origin weiterleitet.

Ob Referrer Policy und Content Security Policy (CSP) von aktuellen Webbrowsern unterstützt und genutzt werden können, müsste erst geprüft werden. Mit der CSP wären aber weitere Lösungsansätze denkbar. Eine Variante wäre die `sandbox` Direktive im HTTP Response Header beim Service Provider zu verwenden. Damit könnte ein Nachrichtenaustausch mit dem Dritten, dem Identity Provider, über Web Messaging mit Absender-Origin `null` erfolgen.
Wenn die Origin `null` ist kann auch mit CORS und XMLHttpRequest kommuniziert werden, ohne ein Einbetten des Browser-Kontexts vom Dritten. Dann wären auch WebSocket oder Server-Sent Events denkbar.

3.2 Implementation Kommunikation

Mit den aufgeworfenen Lösungsansätzen aus den Spezifikationen soll im Folgenden eine Implementierung erfolgen. Dazu werden im ersten Schritt vielversprechende Lösungstechniken auf ihre Eignung in gängigen Webbrowsern getestet. Für eine Variante wird dann die Kommunikation implementiert.

3.2.1 Versuchsaufbau

Der Einfachheit halber werden die Beteiligten als virtuelle Maschinen umgesetzt. Der Identity Provider und Service Provider verfügen über einen Anwendungsserver mit HTTP

Unterstützung, der vom Benutzer Browser angesprochen wird.

Die eingesetzten Softwarekomponenten sollen möglichst offene Lizenzen haben, frei verfügbar sein und sind so gewählt, dass eine Anwendung in einer bestehenden Umgebung mit Apache Tomcat als Anwendungsserver möglich ist. Deshalb wird als Anwendungsserver Apache Tomcat auf dem Linux Betriebssystem verwendet. Als Linux Distribution wurde Debian gewählt. Debian ist aktuell in seiner stabilen Fassung bei Version 8. Mit Apache Tomcat, aktuell in der Version 8, ergibt sich auch die Verwendung von Java, als serverseitige Programmiersprache und Entwicklungsplatform. Aufgrund der Verwendung der in Debian 8 mitgelieferten Laufzeitumgebung für Java, orientiert sich der verfügbare Sprachumfang an Java 1.7. Als Erweiterung für eine ansprechbare Rest-Schnittstelle wird Jersey eingesetzt, eine Referenzimplementation der Java API for RESTful Web Services (JAX-RS).
Für das Benutzersystem, von dem nur ein aktuell gängiger Webbrowser gefordert wird, kommt Chrome zum Einsatz. Die für das Darstellen der Inhalte und Bereitstellen der programmierbaren Softwareschnittstellen verantwortliche Browser Engine von Chrome (Google) ist Blink. Sie basiert auf der in Safari (Apple) verwendeten Webkit Engine, auf der auch Edge (Microsoft) basiert. Die Engine entscheidet über das Rendern der Inhalte. Ein Browser beinhaltet darüber hinaus oft Funktionen wie Tabs, Lesezeichen, Steuerelemente, Verlauf oder Erweiterungen. Weitere Tests sollen dann auch auf anderen aktuell gängigen Webbrowsern ausgeführt werden.

Die Statistik über die Top 5 meistgenutzten Browser zwischen April 2015 und April 2016 von StatCounter.com[15] enthält Chrome, Firefox, Internet Explorer und Safari. Dabei sind Chrome und Safari über ihre Browser Engine verwandt. Firefox verwendet die Gecko Engine und Internet Explorer die Trident Engine. Mit Tests von Chrome, Firefox und Internet Explorer, wären die Technologien der 5 meistgenutzten Browser abgedeckt und damit den Anforderungen aus der Problemstellung in 1.2, gängige Software zu verwenden, nachgekommen.

Zur Prüfung der übermittelten Daten können der Einfachheit halber Entwickler-Werkzeuge des Webbrowsers verwendet werden, die Anfrage, Antwort und entsprechende HTTP-Header anzeigen können. Denn es sind die HTTP-Header Origin und Referer [sic], die den aktuellen Kontext weitergeben können. Zur exakten Feststellung kann aber auch der Netzwerkverkehr auf dem Zielrechner belauscht werden. Für diesen Einsatz sind zum Beispiel tcpdump oder ssldump geeignet.

3.2.2 Webbrowser Spezifika

Als Webbrowser Auswahl werden die im Versuchsaufbau in 3.2.1 beschriebenen Webbrowser Chrome, Firefox und Internet Explorer in ihren aktuellen Fassungen genommen.

[15] Vgl. `http://gs.statcounter.com/#browser-ww-monthly-201504-201604`. Letzter Zugriff: 08.11.2016.

Es werden Chrome Version 52, Firefox Version 48 und Internet Explorer Version 11 verwendet.

Getestet werden sollen vielversprechend erscheinende Varianten, von in 3.1.3 besprochenen Techniken, die für eine Kommunikation geeignet sind, bei der die Identität des aktuellen Kontexts bei der Kommunikation nicht an den Dritten weitergegeben wird. Dazu gehört die Kommunikation über Web Messaging, mit einem in ein IFrame eingebetteten fremden Kontext, unter Zuhilfenahme von Sandboxing, `about:blank` und data:Uris.
Deswegen werden in einer ersten Teststufe ausschließlich diese drei auf Origin und Referrer Verhalten untersucht.

Browser-Tests: Origin und Referrer

In der Testumgebung wird durch den Service Provider ein HTML-Dokument ausgeliefert, welches das Standardgerüst `<html><head></head><body></body></html>` enthält. Der folgende Testcode ist dann in `head` eingefügt worden. Um dem HTML5 Standard zu entsprechen muss der `DOCTYPE` `<!DOCTYPE html>` vor dem `html` Tag angegeben werden [HBF+14, 8.1].

Sandboxing und data:Uri Der Programmcode für Sandboxing und data:Uri wurde für die Arbeit zusammengefasst, weil beide sich nur in drei Zeilen unterscheiden. Die entsprechenden Zeilen sind mit einem jeweils spezifischen Kommentar versehen. Der aufgeführte Fall ist für Sandboxing. Ein Wechsel zum data:Uri Test kann durch Umschalten der markierten Zeilen in den jeweils anderen Kommentarzustand erfolgen.

Beim Test wird per Skript ein IFrame eingefügt, welches zur Diagnose dessen Origin und Referrer anzeigt. Für den Internet Explorer 11 wurde eine Sonderbehandlung eingeführt, weil weder `iframe.srcdoc` noch `data:Uri` unterstützt werden. Der Internet Explorer 11 unterstützt zwar `data:Uri` für zum Beispiel Bilder, aber nicht als Ziel einer Navigation und eines IFrame [16]. Als Ersatz wurde als `iframe.src` eine `javascript:Uri` genutzt. Damit das Skript im IFrame die Diagnose-Informationen anzeigen kann, wurden eine entsprechende Ausnahme für Sandboxing hinzugefügt.

Das Ergebnis des Tests von Sandboxing und data:Uri wurde in Tabelle 3.2 zusammengefasst, weil die Ergebnisse identisch sind.

	Origin	Referrer
Chrome 52	`null`	parent Url
Firefox 48	`undefined`	`""`
Internet Explorer 11	`undefined`	`""`

Tabelle 3.2: Ergebnis Sandboxing und data:Uri Test hinsichtlich Origin und Referrer

[16] Vgl. `https://msdn.microsoft.com/en-us/library/cc848897.aspx`. Letzter Zugriff: 08.11.2016.

```
<script type="application/javascript">
window.addEventListener("load",function() {
  var iframeScript = "window.addEventListener(\"load\",function(){\n\
    document.body.innerHTML = \"iframe origin: \"+document.origin+\n\
    \"<br/>iframe referrer: \"+document.referrer;});";
  var iframeHTML = "<html><head><scr"+"ipt
↪   type=\"application/javascript\">"+
    iframeScript+
    "</scr"+"ipt></head><body></body></html>";

  var testIframe = document.createElement("iframe");
  document.body.appendChild(testIframe);

  var isIE11 = navigator.userAgent.match(/Trident|rv:11/g);
  if(isIE11) {
    testIframe.src = "javascript: '"+iframeHTML+"'";
  } else {
    testIframe.srcdoc = iframeHTML; //sandbox specific
    //testIframe.src = 'data:text/html;charset=utf-8,' +
↪   encodeURIComponent(iframeHTML); //data:Uri specific
  }
  testIframe.sandbox = "allow-scripts"; //sandbox specific
});
</script>
```

Quelltext 1: Test von Sandboxing und data:Uri auf Origin und Referrer.

Ergänzen ließe sich der Test mit einem `<meta http-equiv="refresh" content="0; url=data:text/html;charset=utf8,AStringHoldingIframeDataUri">`. Dann wird der Referrer in Chrome "". Alternativ und mit gleichem Ergebnis kann in Chrome auch die Referrer Policy folgendermaßen auf dem übergeordneten Dokument verwendet werden: `<meta name="referrer"content="no-referrer">`.
Zu beachten ist, dass sobald Sandboxing aktiv ist, nicht mehr auf diesen Kontext zugegriffen werden kann.

`about:blank` Es wird ein IFrame mit `src="about:blank"` erstellt und in dieses ein Diagnose Skript eingefügt. Das Skript für die Diagnose Informationen wird dynamisch als Script-Node eingefügt und nicht durch Setzen von `iframeObject.contentWindow.document.body.innerHTML = "...";`, weil sonst der Code ins IFrame eingefügt, aber nicht ausgeführt wird.

Das Ergebnis des Tests ist in Tabelle 3.3 zu finden.

```
<script type="application/javascript">
window.addEventListener("load",function() {
  var testIframe = document.createElement("iframe");
  testIframe.src="about:blank";
  testIframe.addEventListener("load",function(){
    var script = document.createElement("script");
    script.type = "application/javascript";
    script.innerHTML = "document.body.innerHTML = \n\
      \"iframe origin: \"+document.origin+\n\
      \"<br/>iframe referrer: \"+document.referrer";
    testIframe.contentWindow.document.head.appendChild(script);
  });
  document.body.appendChild(testIframe);
});
</script>
```

Quelltext 2: Test von `about:blank` auf Origin und Referrer.

	Origin	Referrer
Chrome 52	parent Url	parent Url
Firefox 48	`undefined`	`""`
Internet Explorer 11	`undefined`	`""`

Tabelle 3.3: Ergebnis `about:blank` Test hinsichtlich Origin und Referrer

Verbesserungen Verbesserungsmöglichkeiten bestehen für Chrome 52, da Firefox 48 und Internet Explorer 11 in den Tests bereits Origin und Referrer anonymisieren. Zum Einsatz dafür können in 3.1.3 vorgestellte Techniken zum Referrer und Origin anonymisieren kommen.
Für das Beispiel data:Uri und Sandboxing kann im Chrome 52 sowohl `<meta name="referrer" ... >` als auch `<meta http-equiv="refresh" ...>` zum erfolgreichen Anonymisieren von Origin und Referrer führen. Für Ersteres ist ein `<meta name="referrer" content="no-referrer">` im `head` Element des ausliefernden und einbettenden Dokuments erforderlich. Für Zweiteres wird das eingebettete Dokument im IFrame auf eine `data:Uri` weitergeleitet. Dazu wird folgende Anweisung in das `head` Element des eingebetteten Dokuments eingefügt: `<meta http-equiv="refresh" content="0; url=data:text/html,<html><head></head><body>test </body></html>">`. Die in diesem `meta` Element angegebene data:Uri kann dabei auch durch die Diagnose data:Uri aus dem data:Uri Beispiel ersetzt werden. Für den Internet Explorer 11, der für diesen Einsatz keine data:Uris unterstützt, braucht nicht auf eine Sonderlösung zurückgegriffen werden, weil in den Tests bereits erfolgreich Origin und Referrer anonymisiert sind.
Im `about:blank` Beispiel funktionierten `<meta name="referrer" ... >` und `<meta`

`http-equiv="refresh" ...>` nicht zum Anonymisieren von Origin und Referrer im Chrome 52. Folgendes Beispiel fügt stattdessen ein `form` Element mit Nutzdaten in ein `iframe` ein und sendet das Formular.

```
<script type="application/javascript">
window.addEventListener("load",function() {
  var testIframe = document.createElement("iframe");
  testIframe.src= "about:blank";
  testIframe.addEventListener("load",function(){
    var containerDiv = document.createElement("div");
    containerDiv.innerHTML = '<form id=\'testForm\''+
      'method=POST action=\''+ yourEndpointUrl + '\'>' +
      '<input type=hidden name=\'key\' value=\'value\' /></form>';
    testIframe.contentWindow.document.body.appendChild(containerDiv);

↪   testIframe.contentWindow.document.getElementById("testForm").submit();

    //alternative using a Element instead of form
    //var containerDiv = document.createElement("div");
    //containerDiv.innerHTML =
    //  "<a id=\"testLink\" href=\""+yourEndpointUrl+"\"></a>";

↪   //testIframe.contentWindow.document.body.appendChild(containerDiv);

↪   //testIframe.contentWindow.document.getElementById("testLink").click();
  });
  document.body.appendChild(testIframe);
});
</script>
```

Quelltext 3: Nachricht ohne Referrer und Origin in `about:blank`.

Statt dem `form` Element kann auch, wie im Kommentarteil enthalten, das `a` Element genutzt und die Daten in der Url übertragen werden.
Da zur Übermittlung HTML Steuerelemente verwendet werden und kein CORS, wird keine Origin mitgeliefert. Der Referer [sic] wird nicht gesendet, weil das Steuerelement sich in einem `about:blank` IFrame befindet. Für eine zusätzliche Absicherung kann nach dem Einfügen des `form` Elements und vor dem Absenden des Formulars, Sandboxing aktiviert werden.
Nachteil der Lösung ist der nicht enthaltene Rückkanal für Daten, die als Antwort von übermittelten Daten dienen. Eine mögliche Umsetzung eines bidirektionalen Kommunikationskanals für per `form` oder `a` gesendete Daten wird im folgenden Abschnitt, in 3.2.3, vorgestellt.

3.2.3 Kommunikation über IFrame mit HTTP POST und Web Messaging

Eine Möglichkeit der Kommunikation im Sinne der Problemstellung, in 1.2, besteht mittels der Kombination vom bereits vorgestellten HTTP POST in einem IFrame und Web Messaging. Das Ziel ist eine Kommunikation mit einem Dritten ohne dabei die Identität des Diensts, dem der aktuelle Browser-Kontext zugeordnet ist, weiterzugeben. Origin und Referrer werden dabei aus der Sende-Richtung über HTTP POST per Formular nicht gesendet. Diese Variante wurde gewählt um die bereits, in 3.2.2, aufgeworfene Frage des Rückwegs einer Nachricht beantworten zu können. Ergänzend zum Quellcode 3, der verbesserten `about:blank` Variante, erhält der Sender noch einen Web Messaging Empfänger Part.

```
window.addEventListener('message', function(event) {
  if(event.origin !== postMessageSenderOrigin) return;
  var sentData = event.data;
});
```

Quelltext 4: Web Messaging Empfänger des ursprünglichen Senders.

Der eingebettete Kontext, an den die Nachricht ohne Origin und Referrer gesendet wird, liefert auf die HTTP POST Anfrage ein Antwort-Dokument. In diesem sendet er mittels Web Messaging eine Nachricht an den übergeordneten, einbettenden Kontext, wie folgt.

```
window.parent.postMessage('response message','*');
```

Quelltext 5: Web Messaging Nachricht an den ursprünglichen Sender.

Weil der Sender nur in Form eines Absender-Kontexts bekannt ist, wird die Nachricht an die Origin `*` gesendet. Dem Empfänger der Web Messaging Nachricht wird die Origin im `message` Event mitgeliefert. Dies stellt aber für eine Lösung kein Problem dar, weil in der Problemstellung in 1.2 nur gefordert wurde, dass eine Seite, der Dritte, nicht die Identität der anderen, des Diensts, erfährt.

Der eingebettete Kontext kann die Url des übergeordneten, einbettenden Kontexts ändern. Zum Beispiel mit `window.parent.location = "https://maliciousSite.com";`. Zu beachten ist, dass `window.parent.location` vom eingebetteten Kontext nicht lesbar ist. Alternativ kann auch ein HTML Element mit `target = "_parent";` per Skript ausgelöst werden.
Als Schutz kann das IFrame mit Sandboxing versehen werden. Das sollte nach Einfügen und vor dem Absenden des Formulars passieren. Der Quellcode 3 wird dazu in der Zeile auf `appendChild(...);` folgend um `testIframe.sandbox = "allow-scripts";` ergänzt. Ohne `allow-scripts` könnte Web Messaging nicht ausgeführt werden.
Alternativ, aber weniger geeignet, kann gegen eine Weiterleitung das `beforeunload` und

`unload` Event des `window` Objekts vom übergeordneten, einbettenden Kontext verwendet werden. Mittels einer auf das Event gebundenen, temporären Weiterleitung auf die aktuelle Seite, mit einem anderen Hash-Suffix an der Url und erneutem laden des ursprünglichen Hash-Suffixes, wird das Laden einer anderen Webseite verhindert. Das dazugehörige Skript ist in Quellcode 6 enthalten.

```
var preventNavigation = function (event) {
        var originalHashValue = window.document.location.hash;
    window.setTimeout(function () {
            window.document.location.hash = 'preventNavigation' + new
↪   Date().getTime();
        window.document.location.hash = originalHashValue;
    }, 0);
};
window.addEventListener('beforeunload', preventNavigation, false);
window.addEventListener('unload', preventNavigation, false);
```

Quelltext 6: Verhindern einer Url Änderung.

Eine Nebenwirkung ist, dass auch ein Ändern der aktuellen Webseite mittels Eingabe einer anderen Url durch den Benutzer fehlschlägt. Der zugehörige Webbrowser Kontext kann aber geschlossen werden oder die preventNavigation Funktion mittels `window.removeEventListener('beforeunload', preventNavigation, false);`, beziehungsweise analog für das `unload` Event, wieder entfernt werden.

3.2.4 Weitere Kommunikations-Ansätze

Content Security Policy Die in 3.1.3 für das `meta` Element vorgestellte Content Security Policy (CSP) kann auch als HTTP-Header vom Web-Server gesendet werden. Einige Funktionen wie das Sandboxing können dabei nur per HTTP-Header definiert werden. Verwendet die Webseite vom besuchten Anbieter den `Content-Security-Policy` Header mit dem `sandbox` Wert, greift für das Dokument das in 3.1.1 beschriebene Sandboxing. Mit dem Header `Content-Security-Policy: sandbox allow-scripts` kann skriptbasiert kommuniziert werden. Weil die Origin des Dokuments eine GUID und damit für Skripte `null` ist, wird auch kein Referrer gesendet. Eine Einschränkung besteht darin, dass nicht mehr auf den Inhalt von IFrame Dokumenten zugegriffen werden kann, weil das aktuelle Dokument durch die CSP eine eigene GUID Origin hat. Der Inhalt des IFrames muss in dem Fall initial über das `src` oder `srcdoc` Attribut festgelegt werden. Mit dem IFrame kann dann über Web Messaging kommuniziert werden. Die Origin vom Absender ist dabei `null`. Auch ohne IFrame kann mittels Cross-Origin Resource Sharing und XMLHttpRequest oder Server-Sent Events mit einer fremden Origin, ohne Weitergabe von Origin oder Referrer kommuniziert werden. Für WebSocket ist die Origin

entsprechend `null`. WebRTC, vorgestellt in 3.1.3, kann auch benutzt werden, solange kein zentraler Signaling-Server oder NAT Traversierungsserver verwendet wird. Zum Verbindungsaufbau kann zum Beispiel XMLHttpRequest verwendet werden.
Anzumerken ist, dass Sandboxing per CSP von den drei getesteten Webbrowsern unterstützt wird, auch wenn der Implementierungsgrad der CSP sich unterscheidet[17]. So unterstützt der Internet Explorer 11 offiziell keine CSP, kann aber Sandboxing über den HTTP Header `X-Content-Security-Policy`, analog zur CSP aktivieren. Chrome 52 mit CSP Level 2 und Firefox 48 mit CSP Level 1 unterstützen entsprechend Sandboxing.

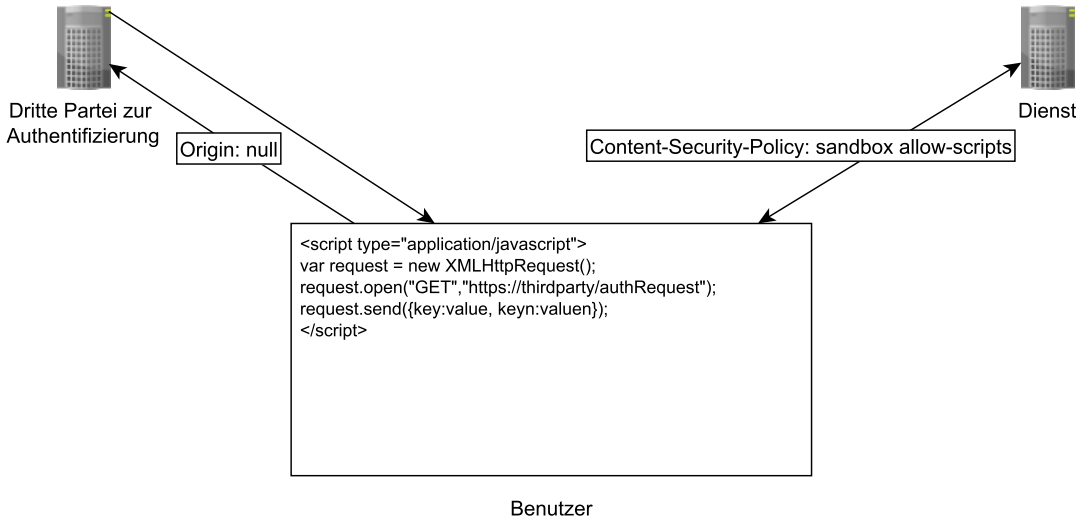

Abbildung 3.2: Kommunikation mit Content Security Policy Sandbox.

Proxy-Kontext Mit den bereits getesteten Techniken zur Schaffung eines Browser-Kontexts mit Origin `null` und leerem Referrer in 3.2.2, kann ein solcher eingebetter Browser-Kontext erstellt werden, der als Proxy Kontext fungiert. Der Proxy Kontext kommuniziert mit dem einbettenden Kontext über Web Messaging. Er steht dabei unter Kontrolle des einbettenden Kontexts und gibt die durch Web Messaging erhaltene Origin entsprechend nicht weiter. Innerhalb des Proxy-Kontexts kann mit Cross-Origin Resource Sharing und XMLHttpRequest oder Server-Sent Events mit einer fremden Origin kommuniziert werden, wobei Origin und Referrer entsprechend `null` beziehungsweise leer sind. WebSocket und WebRTC können auch genutzt werden, für zweiteres gilt aber, dass kein zentraler Server genutzt werden darf. Der Proxy Kontext kann auch in einem IFrame den Browser Kontext von einem Dritten einbetten und mit diesem per Web Messaging kommunizieren. Die Verwendung von Sandboxing wäre dabei angebracht, da sonst aus dem Kontext des Dritten die Url vom Proxy-Kontext oder seinem einbettenden Kontext geändert werden kann.
Eine abgewandelte Variante wäre ein übergeordneter Proxy Kontext, der an der Kommunikation beteiligte Browser Kontexte jeweils in einem IFrame einbettet und zwischen

[17]Vgl. `http://caniuse.com/#search=content%20security%20policy`. Letzter Zugriff: 08.11.2016.

ihnen per Web Messaging Nachrichten vermittelt. Absender der Nachrichten ist der Proxy Kontext mit Origin `null`.

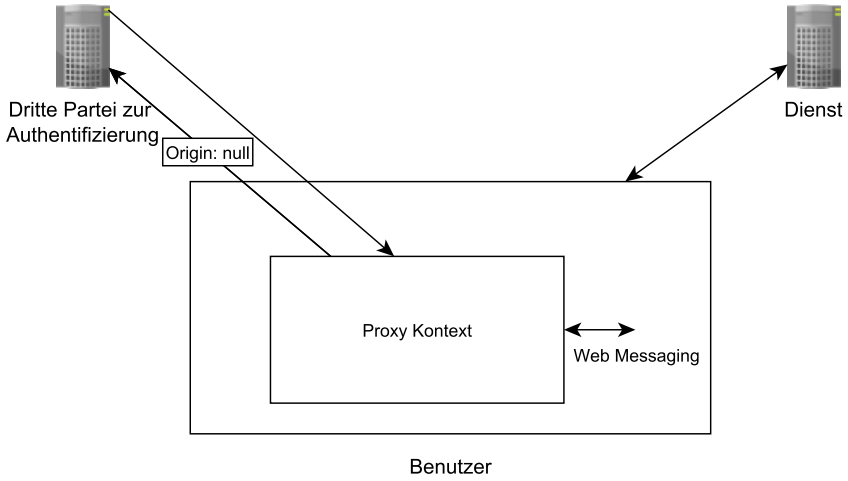

Abbildung 3.3: Kommunikation mit Proxy Kontext.

3.3 Authentifizierung

Mit den gefundenen Kommunikations-Möglichkeiten wird eine Authentifizierung im Sinne der Problemstellung in 1.2 möglich. Dabei soll eine Authentifizierung des Benutzers durch einen Dritten durchgeführt werden, ohne dass die Identität des Diensts an den Dritten weitergegeben wird, damit dieser Benutzer und Dienst nicht verknüpfen kann. Beispielhaft könnte ein Verfahren folgendermaßen aussehen.

1. Der Benutzer initiiert die Authentifizierung auf der Webseite des Service Providers, worauf dieser eine verschlüsselte Anfrage für den Identity Provider erstellt.
2. Der Webbrowser des Benutzers baut, zum Beispiel nach 3.2.3 oder 3.2.4, einen Kommunikationskanal zum Identity Provider auf. Die Identität vom Service Providers wird vertraulich behandelt und es erfolgt keine Weitergabe über Origin oder Referrer des Service Providers an den Identity Provider.
3. Über den etablierten Kommunikationskanal wird die Authenfizierungsanfrage an den Identity Provider gesendet und der Benutzer führt beim Identity Provider die weitere Authentifizierung durch.
4. Ist die Authentifizierung beim Identity Provider erfolgreich, sendet dieser über den etablierten Kommunikationskanal eine verschlüsselte Nachricht über den Erfolg für den Service Provider.
5. Der Webbrowser des Benutzers leitet die verschlüsselte Antwort an den Service Provider weiter.

6. Der Service Provider verifiziert anhand der erhaltenen Nachricht das Ergebnis der Authentifizierung und betrachtet den Benutzer bei Erfolg als authentifiziert.

Implementations-Vorschläge Schritt 1 kann zum Beispiel über XMLHttpRequest oder ein einfaches Formular realisiert werden. In der symmetrisch oder asymmetrisch für den Identity Provider verschlüsselten Nachricht, deren Schlüssel auf anderem Wege erlangt sein muss, sollte Schlüsselmaterial für eine Antwort vom Identity Provider enthalten sein.

Schritt 2 beschreibt den Aufbau der Verbindung ohne Weitergabe der Identität des Diensts. Das können die beschriebenen Varianten sein. Also zum Beispiel wie in 3.2.3 mit dem Erstellen eines eingebetteten Kontexts über ein per HTTP POST gesendetes Formular, welches per Web Messaging antworten kann. Oder wie in 3.2.4 vorgestellt, ein XMLHttpRequest aus einer Content Security Policy Sandbox mit `allow-scripts`. Eine weitere in 3.2.4 vorgestellte Möglichkeit besteht in der Erstellung eines eingebetteten Kontexts über Sandboxing oder data:Uri.

In Schritt 3 kann dann zum Beispiel aus dem anonymisierten Kontext heraus ein neues Browser Fenster mit `var windowObjectReference = window.open(Url,WindowName, [OptionalWindowFeatures]);` geöffnet werden. Darin können dann die weiteren Authentifizierungsschritte beim Identity Provider ablaufen.

Die Antwort der Authentifizierung in Schritt 4 kann je nach Kommunikationskanal per Push oder Poll an den Webbrowser des Benutzers übergeben werden. Im Fall von `window.open(..)` besteht die Möglichkeit vom geöffneten Kontext per Web Messaging an den `window.opener` Nachrichten zu senden. Erhält ein Proxy Kontext die Nachricht, muss diese an den Kontext des Service Providers per Web Messaging weiter gegeben werden.
Die Antwort wird dabei mit dem auf dem Hinweg erhaltenen Schlüsselmaterial verschlüsselt.

Schritt 5 kann dann wieder mit klassischen Methoden wie einem XMLHttpRequest realisiert werden. Das gleiche gilt für Schritt 5, welcher im Kontext des Service Providers statt findet.

3.4 Zusammenfassung

Es wurden gemäß den Zielen aus 1.3 die lösungsrelevanten Techniken eines standardkonformen Browsers untersucht. Für die ausgemachte Teilproblemstellung, im Browser mit einem Dritten zu kommunizieren ohne die Identität des aktuellen Browser-Kontexts vom besuchten Dienst weiterzugeben, wurden Lösungsansätze entwickelt, implementiert und in einem Versuchsaufbau getestet. Die darauf aufbauende Authentifizierung wurde aufgrund des Umfangs der Arbeit skizziert, aber nicht implementiert.

Aufgefallen ist, dass die Browser in ihren Implementierungen hinsichtlich der Origin des eingebetteten Dokuments den HTML5 Standard [HBF+14, 5.2] unterschiedlich interpretieren. Laut Standard übernimmt ein `about:blank` Dokument die Origin vom erstellenden Browser-Kontext, falls dieser existiert. Für aus anderen Dokumenten per data:Url oder javascript:Url erzeugte Dokumente wird die effektive Skript Origin vom navigierenden Dokument übernommen. Deswegen sollte das in den Tests per data:Url oder javascript:Url erzeugte IFrame die Origin vom navigierenden Dokument erhalten, was nicht der Fall ist. Es ist anzunehmen, dass die Browser ein setzen des IFrame `src` Attributs als manuelle Eingabe der data:Url oder javascript:url annehmen. Aus der isolierten, eingebetteten Sicht des IFrames ist dessen Url auch manuell definiert.

4 Ergebnis

Untersucht wurden die Kommunikations-Techniken eines standardkonformen Webbrowsers hinsichtlich ihrer Eignung, für die in der Problemstellung in 1.2 aufgeworfene Authentifizierung. Dabei sollte eine dritte, authentifizierende Partei nicht erfahren, welcher Dienst die Authentifizierung veranlasst hat. Hinsichtlich dessen wurden folgende Kommunikations-Techniken untersucht.

- XMLHttpRequest
- Web Messaging
- WebSocket
- Server-Sent Events
- WebRTC
- diverse HTML-Elemente

Keine der untersuchten Kommunikations-Techniken war für sich genommen, für diesen speziellen Anwendungsfall geeignet. Die Identität des vom Benutzer besuchten Diensts wurde entweder über den Referer- [sic] oder Origin HTTP Header, oder über die `origin` Eigenschaft eines `message` Events an den Dritten weitergegeben.
Mit dem Wissen über die Festlegung von Origin und Referrer und in welchen Fällen diese nicht gesetzt werden, entstanden vielversprechende Lösungsideen für die gewünschte Kommunikation. Diese Ansätze wurden implementiert und in ausgewählten Webbrowsern auf Erfolg getestet. Dazu zählten:

- IFrame mit data:Uri
- IFrame mit aktivem Sandboxing
- IFrame mit `about:blank`

Mittels ergänzender Techniken konnten alle drei Ansätze hinsichtlich der geforderten Grundvoraussetzungen erfolgreich implementiert werden. Die nur für unidirektionale Nachrichten, zum Dritten, geeignete Variante `about:blank` wurde so erweitert und als Kommunikations-Lösung implementiert, dass eine bidirektionale Kommunikation möglich war.
Zwei weitere Varianten, die darauf abzielen einen Kontext zu schaffen, dessen Origin `null` ist, wurden beschrieben. Damit wurden die Grundvoraussetzung geschaffen, um über den Webbrowser des Benutzers mit einem Dritten zu kommunizieren, ohne die Identität des

besuchten Diensts weiterzugeben. Auf dieser Grundlage wurde eine Authentifizierung beschrieben, die den speziellen Anforderungen der aufgeworfenen Problemstellung in 1.2 gerecht wird.

Als standardkonform wird ein Browser angesehen, der die in 3.1.1 „Konzepte Browser“ aufgeführten Open Web Platform Standards implementiert, mindestens aber HTML5[HBF+14] und abhängige Standards.
So sind data:Uris in [HBF+14, 2.1.1 Resources] beschrieben und Sandboxing in [HBF+14, 5.4 Sandboxing]. JavaScript, gemeint ist ECMAScript 5.1 [Ecm11], Cross-Origin Resource Sharing, XMLHttpRequest, Server-Sent Events und WebSocket sind auch als Abhängigkeiten im HTML5 Standard [HBF+14, 2.2.2 Dependencies] enthalten. WebRTC ist nicht im HTML5 Standard enthalten und befindet sich derzeit noch im Status eines „Working Draft“.
Trotzdem unterscheiden sich Browser in ihrer Implementation. So wurde in 3.2.2 „Browser Tests“ bereits eine Sonderbehandlung des Internet Explorer 11 erläutert. Einen Hinweis darauf, inwiefern bestimmte Funktionen in Browsern enthalten sind liefern Web Dienste, wie zum Beispiel `http://caniuse.com`. So besitzen Content Security Policy[1] und Sandboxing[2] eine relativ breite Unterstützung. Infolge ist mit einer Kombination aus den vorgestellten Kommunikationsansätzen eine breite Unterstützung möglich.

Die skizzierte Authentifizierung durch einen Dritten wird in den Kontext der verwandten Dritt-Partei Authentifizierungs-Techniken Kerberos, OpenID und Auth²(nPA) gesetzt. Darauf folgend sollen mögliche Kritikpunkte oder Schwächen beleuchtet werden und ein zukunftsweisender Ausblick gegeben werden.

4.1 Vergleich mit verwandten Lösungen

4.1.1 Kerberos

Kerberos setzt mit seinen Tickets, die dem Benutzer gegeben werden, damit dieser sie beim Dienst für eine Authentifizierung einsetzen kann, auch auf ein dezentrales Prinzip. Allerdings existiert eine Zuordnung von Benutzer und Dienst, damit Kerberos entscheiden kann, ob ein Benutzer berechtigt ist einen Dienst zu benutzen, worauf diesem ein entsprechendes Ticket ausgestellt wird. Der Dienst selbst ist auch bei Kerberos angemeldet und hat einen gemeinsamen Schlüssel mit Kerberos ausgehandelt, mit dem beide ihre Nachrichten für den jeweils anderen verschlüsseln. In der Kurzvorstellung von Kerberos, in 2.3.1, wurde bereits festgestellt, dass Kerberos nicht für eine Lösung geeignet ist.

[1] Vgl. `http://caniuse.com/#feat=contentsecuritypolicy`. Stand Oktober 2016, unterstützen 9 von 10 aktuellen Browsern die CSP.

[2] Vgl. `http://caniuse.com/#feat=iframe-sandbox`. Stand Oktober 2016, unterstützen 9 von 10 aktuellen Browsern Sandboxing.

Auch unterscheidet sich das Szenario, weil es in dieser Arbeit nicht um die Wiederverwendbarkeit der Authentifikation bei verschiedenen Diensten geht, sondern darum dem Dritten nicht zu ermöglichen eine Anfrage einem Dienst zuzuordnen.

Im Gegensatz zu Kerberos muss der Netzwerkverkehr nicht gegen den Einblick von dritten geschützt werden, weil auf das verschlüsselte HTTPS Protokoll zurückgegriffen werden kann und sollte. Allerdings lassen sich Ideen zum Verhindern von Replay-Angriffen, bei denen die gleiche Nachricht erneut gesendet wird, übernehmen. So könnte zum Beispiel der Service Provider eine begrenzte Gültigkeit einer zu erbringenden Authentifikation erzwingen oder die Adresse des Benutzers überprüfen.

4.1.2 OpenID

OpenID ist für den Einsatz mit einem Webbrowser konzipiert und nutzt entsprechend das HTTP Protokoll. Der Fokus von OpenID liegt darauf, dass verschiedene OpenID (Identity) Provider für eine Authentifizierung genutzt werden können. Im Gegensatz dazu entscheidet in der vorgeschlagenen Authentifikation, in 3.3, der Service Provider über die Wahl des Identity Providers. OpenID setzt auf direkte Kommunikation zwischen Identity Provider und Service Provider, Relying Party genannt. Zum einen, um Informationen zu einer vom Benutzer gegeben Identität, in Form einer Url abzurufen und zum anderen, um ein gemeinsames Geheimnis zum Signieren untereinander versendeter Nachrichten zu etablieren. Gleichermaßen wird bei der Weiterleitung des Benutzers von der Relying Party zum Identity Provider über den Origin oder Referrer HTTP-Header die Identität der Relying Party weitergegeben werden. Eine Gemeinsamkeit der vorgestellten Authentifizierung mit OpenID ist die freie Wahl der Art des Identitätsnachweises beim Identity Provider. OpenID leitet den Benutzer von der Relying Party, wie in der vorgestellten Authentifizierung, zum Identity Provider und danach wieder zurück und hat damit einen ähnlichen Umgang für den Benutzer.

OpenID verwendet zum Schutz vor Replay-Angriffen eine Nonce, welche einen zufälligen Wert darstellt, die die Relying Party sendet und in einer Antwort erwartet. Die gleiche Nonce wird dabei nur einmal akzeptiert. Die Antwort des Identity Providers enthält eine Signatur, anhand derer die Relying Party feststellt, dass die Nachricht vom Identity Provider stammt.

4.1.3 Auth^2(nPA)

Auth^2(nPA) stellt analog zur vorgeschlagenen Authentifizierung, in 3.3, eine Lösung der in 1.2 vorgestellten Problemstellung dar. Im Vergleich zu der in 3.3 vorgestellten Authentifizierung, wird ein anonymisierender Proxy verwendet, statt die Kommunikation vollständig über den Benutzer abzuwickeln. Die Nachricht von der Einrichtung, vergleichbar mit dem Service Provider, zum Dienstanbieter, vergleichbar mit dem Identity Provider, wird aber genauso über den Benutzer weitervermittelt. Dabei muss, zum

Schutz vor Profilbildung, gleichermaßen sichergestellt werden, dass der Dienstanbieter nicht die Identität der Einrichtung erfährt. Die Nachrichten zwischen Einrichtung und Dienstanbieter sind dabei auch verschlüsselt, wobei auf Standardverfahren verwiesen wird.

4.2 Schwachstellen und Angriffsvektoren

Wird das Authentifizierungsverfahren, wie in 3.3 vorgeschlagen, umgesetzt, so sollten folgende Voraussetzungen gesichert sein.

- Anonymität vom Service Provider gegenüber dem Identity Provider[3]
- Vertraulichkeit und Authentizität der Kommunikation
  - zwischen Service Provider und Identity Provider
  - zwischen Benutzer und Service Provider
  - zwischen Benutzer und Identity Provider
- Funktionssicherheit bei allen Beteiligten

Für eine grundsätzliche Absicherung der Verbindung zwischen dem Benutzer zum Service Provider und vom Benutzer zum Identity Provider wird vorausgesetzt. Dabei bietet sich das HTTPS Protokoll mit SSL/TLS und Public Key Infrastructure (PKI) an. Die öffentlichen Schlüssel werden dabei von den Webbrowser Herstellern mitgeliefert.

Eine Möglichkeit die geforderte Anonymität vom Service Provider gegenüber dem Identity Provider sicher zu stellen, wäre das Verhalten des Webbrowsers vom Benutzer hinsichtlich Origin und Referrer mit einem eigenen Test Service zu prüfen. Weil der Origin HTTP-Header nur gesendet wird, wenn eine Nachricht an eine fremde Origin gerichtet wird, muss der Test Service für eine unterschiedliche Origin mindestens einen anderen Port als der eigentliche Dienst benutzen. Zum Test Service könnte der anonymisierende Kanal aufgebaut und eine Nachricht gesendet werden. Wird vom Test Service festgestellt, dass Origin `null` und Referrer leer sind, kann der Kanal zum Identity Provider aufgebaut werden.

Ändern sich die Web-Standards und Implementierungen in den Webbrowsern, besteht durchaus die Eventualität, dass Origin und Referrer nicht mehr wie angedacht anonymisiert werden können oder weitere HTTP-Header gesendet werden, die eine Anonymisierung des Service Providers untergraben.
Zum Beispiel enthält das `location` Objekt[4] im Chrome 52 eine Eigenschaft `ancestor Origins`, über die ein eingebetter Kontext die Origin von allen übergeordneten Kontexten auslesen kann. Für den Lösungsvorschlag zur Kommunikation per IFrame mit HTTP

[3] Wie in der Problemstellung in 1.2 gefordert.
[4] Erreichbar über `window.location` oder `window.document.location`.

POST und Web Messaging in 3.2.3, kann die Anonymität des Service Providers gegenüber dem Identity Provider von zweiterem aufgehoben werden. Als Gegenmaßnahme kann der Kontext des Identity Providers per Content Security Policy Sandboxing verwenden, womit seine Origin in `location.ancestorOrigins` `null` wird.

Für eingebettete Kontexte gilt trotz Sandboxing keine absolute Sicherheit. Im Zweifel kann der Benutzer dazu verleitet werden Informationen preiszugeben. Ebenfalls verhindert Sandboxing nicht, dass der Kontext weiterhin durch andere Kontexte adressier- und ansprechbar ist. Allerdings kann mit diesen nur sehr begrenzt interagiert werden, siehe 3.1.1.
Der Service Provider sollte sich davor absichern, selbst eingebettet zu werden. Die Bedingung `window===window.top` ist dann erfüllt, wenn der Kontext nicht eingebettet ist und kann vom Service Provider vor Einsatz entsprechend geprüft werden.

Werden Nachrichten mit Absender Origin `null` akzeptiert, wie zum Beispiel vom Identity Provider, so kann nicht unterschieden werden ob die Nachricht von einem Unbeteiligten stammt. Das kann im Falle von CORS oder Web Messaging auftreten. Deswegen sollten die empfangenen Nachrichten, soweit möglich, auf korrekten Inhalt überprüft werden.
Verwendet der Service Provider den auf anderem Wege beschafften, nicht-öffentlichen Public Key des Identity Providers und verschlüsselt seine Nachrichten damit, kann der Empfänger sich bei erfolgreichem Entschlüsseln mit seinem privaten Schlüssel sicher sein, dass der Absender jemand ist, dem er diesen nicht-öffentlichen Public Key anvertraut hat. Das gilt zumindest solange, wie der nicht-öffentliche Public Key geheim gehalten wird. Ist das der Fall, sollte ein Schutz vor einer Nachrichtenüberflutung ausreichen.

Um die Anonymität des Service Provider vor dem Identity Provider zu untergraben, könnte ersterer prüfen ob sich Ressourcen von zweiterem im Webbrowser Cache des Benutzers befinden. Dazu könnte eine Liste mit potenziellen Kandidaten dienen, von denen jeweils Ressourcen per HTTP geladen werden und dabei die Zeit gemessen wird. Als Gegenmaßnahme könnte der Webbrowser angewiesen werden, keinen Cache zu verwenden. Dazu wird, HTTP 1.1 angenommen, der HTTP-Header `Cache-Control: no-cache, no-store, must-revalidate` [FR14, 5.2] mitgesendet.

Der Benutzer oder ein Angreifer, der die Kommunikation belauscht, sollen keine erfolgreichen Antworten des Identity Providers erneut für eine Authentifikation beim Service Provider verwenden können. Das entspricht einem Replay Angriff. Zusätzlich zur unterliegenden Verschlüsselung der Verbindung vom Benutzer zum Identity Provider und Service Provider, verschlüsseln diese beiden ihre Nachrichten untereinander. Damit kann der Benutzer die ausgetauschten Nachrichten nicht lesen, unbemerkt verändern oder unbemerkt selbst erstellen.
Ein erneutes Senden kann bemerkt werden, indem eine einmal gültige Nachricht mitgesendet wird. Dass kann das vom Service Provider mitgelieferte Schlüsselmaterial sein, welches neben einer begrenzten Lebens- und Einlösedauer nur einmal gültig ist. Zusätzlich könnte noch die IP-Adresse vom Benutzer oder eine Session-ID des Service Providers zur Prüfung der Gültigkeit verwendet werden.

4.3 Datenschutz

Der Benutzer gibt Identity Provider und Service Provider jeweils Informationen, analog zum Besuch von deren Webseite, weiter. Dazu gehören Netzwerk- und Softwareparameter, unter anderem IP-Adresse, Parameter zum Browser und im Browser gespeicherte Sitzungsinformationen. Darüber hinaus fallen die jeweils erforderlichen, anzugebenden Informationen an. Das ist beim Service Provider die behauptete Identität, die durch die Authentifizierung verifiziert werden soll. Beim Identity Provider fallen die erforderlichen Informationen zur dort behaupteten Identität und deren Nachweis an.
Als Vermittler dient der Benutzer im Falle des zufälligen Schlüsselmaterials vom Service Provider an den Identity Provider. Auch die damit verschlüsselte Antwort des Identity Providers an den Service Provider wird von ihm weitergeleitet. Um dem Service Provider zu ermöglichen, eine Zuordnung von der dort behaupteten Identität und der vom Identity Provider verifizierten Identität zu treffen, sollte in der Nachricht ein beiden Seiten bekannter, pseudonymer Identifikator übermittelt werden.
Mit der stattfindenden Kommunikation über den Benutzer lernt der Identity Provider, dass eine Anfrage aus einer Menge von autorisierten Service Providern stattfindet. Eine Zuordnung oder Profilbildung von Benutzer zu Dienst ist ihm nicht möglich. Der Service Provider kennt bereits den Identity Provider und erlangt kein weiteres Wissen. Der Benutzer erlangt, durch die für ihn verschlüsselten Nachrichten kein Wissen kein zusätzliches Wissen.

Wird die Datenübermittlung dem Benutzer transparent dargestellt und seine Zustimmung eingeholt, ermöglicht es diesem seiner informationellen Selbstbestimmung nachzukommen. Des Weiteren wurde versucht geringstmöglich Daten zu übermitteln.

5 Fazit

Anhand der gefundenen Kommunikations-Techniken des Webbrowsers, die die Identität des aktuellen Kontexts nicht weitergeben, kann die skizzierte Authentifizierung durch einen Dritten vorgenommen werden, ohne dass dieser dabei die Identität des genutzten Diensts erfährt. Dabei agiert der Webbrowser des Benutzers als Vermittler zwischen dem Dienst und dem authentifizierenden Dritten. Vom Benutzer wird dabei nicht mehr als ein gängiger Webbrowser gefordert. Weil im Webbrowser enthaltene Standard-Technologien verwendet werden, ist keine Installation von Komponenten auf Benutzer-Seite erforderlich. Der teilweise unterschiedlichen Umsetzung der Standards durch die Browser-Entwickler wurde Rechnung getragen. Die Ziele in 1.3 sind damit erfüllt.

Damit kann datenschutzrechtlichen Anforderungen nachgekommen werden, bei denen bestimmte Informationen nicht erlangt werden dürfen. Im konkreten Fall ist das, zusätzlich zu den, bei der Authentifikation beim Dritten anfallenden Daten, nicht das Wissen erlangen zu können, bei welchem Dienst die Authentifikation genutzt wird. Eine Profilbildung von Authentisierendem zu Dienst, durch den Identity Provider, kann damit nicht von diesem vorgenommen werden. Aus Sicht des Benutzers sollte das entwickelte Verfahren daher zu begrüßen sein, schließlich fallen weniger Daten an.
Wegen der gestiegenen Komplexität gegenüber einer kompletten Abwicklung beim Service Provider, der wo nötig Daten transparent mit dem Identity Provider austauscht, sollte das vorgeschlagene Verfahren dort eingesetzt werden, wo entsprechende Anforderungen gestellt werden. Das kann aufgrund sensibler Sachverhalte oder rechtlichen Rahmenbedingungen sein.

Die Same-Origin Policy bleibt als Sicherheitsmechanismus intakt, denn es können nach wie vor nur unter Mitarbeit des Dritten, Informationen von ihm über den Webbrowser des Benutzers ausgewertet werden.

5.1 Schlussfolgerungen

Wenn möglich, sollten sinngemäß dafür gedachte Techniken eingesetzt werden, da sich das intendierte Verhalten so auch in Zukunft erwarten lässt. Wenn sich Webbrowser in ihrem Verhalten bezüglich Referrer und Origin unterscheiden, kann das an der Implementation liegen. Es kann aber auch ein Indiz dafür sein, dass der Standard Interpretationsspielraum lässt, der zu dem unterschiedlich implementierten Verhalten führt. Für eine zukunftsfähige Technik ist eine unsichere Basis, auf die aufgesetzt wird, ungünstig.

Deswegen bietet sich besonders der Einsatz der Content Security Policy (CSP) oder Sandboxing auf einem eingebetteten Kontext an. In beiden Fällen ist eine `null` Origin explizit beabsichtigt. CSP und Sandboxing sind explizite Sicherheitsfeatures, wenngleich sie nicht direkt auf ein Anonymisieren von Nachrichten ausgelegt sind.

Der gewählte Ansatz ist benutzerzentrisch, weil Kommunikation und Benutzerinteraktion über den Webbrowser des Benutzers geschieht. Damit ist der Ansatz stark vom Webbrowser abhängig, der nicht unter Kontrolle des Diensts steht. Umso wichtiger für solche Ansätze sind einheitlich umgesetzte Standards in den Browsern.

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
BDSG	Bundesdatenschutzgesetz
CORS	Cross-Origin Resource Sharing
CSP	Content Security Policy
CSRF	Cross-Site-Request-Forgery
CSS	Cascading Style Sheets
DOM	Document Object Model
EU	Europäische Union
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
JSONP	JavaScript Object Notation with Padding
NAT	Network adress translation
nPA	neuer Personalausweis der Bundesrepulik Deutschland
OECD	Organisation für wirtschaftliche Zusammenarbeit und Entwicklung
rID	Restricted Identification
SDP	Session Description Protocol
SOP	Same-Origin Policy
TKG	Telekommunikationsgesetz
TMG	Telemediengesetz
UNGA	UN-Generalversammlung
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
XHR	XMLHttpRequest
XSS	Cross-Site-Scripting

Abbildungsverzeichnis

1.1 Beteiligte Parteien der Problemstellung. 6

3.1 Nachrichtenaustausch mit dem Dienst als Proxy. 26
3.2 Kommunikation mit Content Security Policy Sandbox. 45
3.3 Kommunikation mit Proxy Kontext. 46

Tabellenverzeichnis

3.1 HTML Elemente mit HTTP `GET`/`POST` Verhalten. [HBF+14, 4.] 27
3.2 Ergebnis Sandboxing und data:Uri Test hinsichtlich Origin und Referrer . 39
3.3 Ergebnis `about:blank` Test hinsichtlich Origin und Referrer 41

Quellcodeverzeichnis

1 Test von Sandboxing und data:Uri auf Origin und Referrer. 40
2 Test von `about:blank` auf Origin und Referrer. 41
3 Nachricht ohne Referrer und Origin in `about:blank`. 42
4 Web Messaging Empfänger des ursprünglichen Senders. 43
5 Web Messaging Nachricht an den ursprünglichen Sender. 43
6 Verhindern einer Url Änderung. 44

Literaturverzeichnis

[Alv16] Alvestrand, H.: *Overview: Real Time Protocols for Browser-based Applications*, Januar 2016. https://datatracker.ietf.org/doc/draft-ietf-rtcweb-overview/, work in progress.

[Bar11] Barth, Adam: *RFC 6454 The Web Origin Concept*, Dezember 2011. http://tools.ietf.org/html/rfc6454.

[BBJ16] Bergkvist, Adam, Daniel C. Burnett und Cullen Jennings: *WebRTC*, Mai 2016. https://www.w3.org/TR/2016/WD-webrtc-20160531/.

[BDG90] *Bundesdatenschutzgesetz*. In: *Bundesgesetzblatt Teil I, Seite 1*, Band 73. Bundesanzeiger-Verlag, 29. Dezember 1990. http://www.gesetze-im-internet.de/bdsg_1990/, Zuletzt geändert am 25.02.2015, BGBl. I S. 162.

[Bis02] Bishop, Matthew A.: *The Art and Science of Computer Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002, ISBN 0201440997.

[BLM05] Berners-Lee, T. und R. FieR. Fielding. Masinter: *RFC 3986 Uniform Resource Identifier (URI): Generic Syntax*, Januar 2005. https://tools.ietf.org/html/rfc3986.

[BSH+84] Benda, Ernst, Helmut Simon, Konrad Hesse, Dietrich Katzenstein, Gisela Niemeyer, Hermann Heußner und Johann F Henschel: *BVerfGE 65, 1*. Entscheidungen des Bundesverfassungsgerichts, 65:1–71, 1984. https://dejure.org/dienste/vernetzung/rechtsprechung?Text=BVerfGE%2065,%201.

[CoE81] CoE, Council of Europe: *Explanatory Report to the Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data*, Januar 1981. https://rm.coe.int/CoERMPublicCommonSearchServices/DisplayDCTMContent?documentId=09000016800ca434.

[CoE14] CoE, Council of Europe: *Handbook on European data protection law*. Publications Office of the European Union, 2014, 2nd Auflage, 2014, ISBN 978-92-871-9934-8. http://fra.europa.eu/en/publication/2014/handbook-european-data-protection-law.

[Eck12] Eckert, Claudia: *IT-Sicherheit : Konzepte, Verfahren, Protokolle*. Oldenbourg, München [u.a.], 7., überarb. aufl. Auflage, 2012, ISBN 978-3-486-70687-1.

[Ecm11] Ecma International: *ECMAScript Language Specification*, Juni 2011. http://www.ecma-international.org/ecma-262/5.1/index.html.

[EP95] EU-Parlament: *Richtlinie 95/46/EG*. Amtsblatt der Europäischen Gemeinschaften, (L 281):31–50, Oktober 1995. http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31995L0046, Richtlinie 95/46/EG des Europäischen Parlaments und des Rates vom 24. Oktober 1995 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr.

[EP02] EU-Parlament: *Richtlinie 2002/58/EG*. Amtsblatt der Europäischen Gemeinschaften, (L 201):37–47, Juli 2002. http://eur-lex.europa.eu/legal-content/DE/ALL/?uri=CELEX%3A32002L0058, Richtlinie 2002/58/EG des Europäischen Parlaments und des Rates vom 12. Juli 2002 über die Verarbeitung personenbezogener Daten und den Schutz der Privatsphäre in der elektronischen Kommunikation (Datenschutzrichtlinie für elektronische Kommunikation).

[ES16] Eisinger, Jochen und Emily Stark: *Referrer Policy*, Juni 2016. https://www.w3.org/TR/2016/WD-referrer-policy-20160601/.

[Eur81] Europarat: *Übereinkommen zum Schutz des Menschen bei der automatischen Verarbeitung personenbezogener Daten*, Januar 1981. http://www.coe.int/en/web/conventions/full-list/-/conventions/treaty/108.

[FEL+16] Faulkner, Steve, Arron Eicholz, Travis Leithead, Alex Danilo, Erika Doyle Navara, Edward O'Connor und Robin Berjon: *HTML 5.1*, Juni 2016. https://www.w3.org/TR/html51/.

[FM11] Fette, I. und A. Melnikov: *RFC 6455 The WebSocket Protocol*, Dezember 2011. https://tools.ietf.org/html/rfc6455.

[FR14] Fielding, R. und J. Reschke: *RFC 7231 Hypertext Transfer Protocol – HTTP/1.1*, Juni 2014. https://tools.ietf.org/html/rfc7231.

[Gar05] Garrett, Jesse James: *Ajax: A New Approach to Web Applications*. Februar 2005. http://adaptivepath.org/ideas/ajax-new-approach-web-applications/.

[GG49] *Grundgesetz für die Bundesrepublik Deutschland*. In: *Bundesgesetzblatt Teil I, Seite 1*. Bundesanzeiger-Verlag, 23. Mai 1949. http://www.gesetze-im-internet.de/gg/, Zuletzt geändert am 23.12.2014, BGBl. I S. 2438.

[Hü08] Hühnlein, Detlef: *Identitätsmanagement - Eine visualisierte Begriffsbestimmung*. Datenschutz und Datensicherheit - DuD, (3):161–163, Februar 2008.

[HBF+14] Hickson, Ian, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor und Silvia Pfeiffer: *HTML5*, Oktober 2014. https://www.w3.org/TR/2014/REC-html5-20141028/.

[Hic12] Hickson, Ian: *The WebSocket API*, September 2012. https://www.w3.org/TR/websockets/.

[Hic14] Hickson, Ian: *Server-Sent Events*, Dezember 2014. https://www.w3.org/TR/2014/PR-eventsource-20141209/.

[Hic15] Hickson, Ian: *HTML5 Web Messaging*, Mai 2015. https://www.w3.org/TR/2015/REC-webmessaging-20150519/.

[HJP06] Handley, M., V. Jacobson und C. Perkins: *RFC 4566 Session Description Protocol (SDP)*, Juli 2006. https://tools.ietf.org/html/rfc4566.

[Jaf14] Jaffe, Jeff: *Application Foundations For The Open Web Platform*, Oktober 2014. https://www.w3.org/blog/2014/10/application-foundations-for-the-open-web-platform/.

[Kes14a] Kesteren, Anne: *XMLHttpRequest Level 1*, Januar 2014. http://www.w3.org/TR/2014/WD-XMLHttpRequest-20140130/.

[Kes14b] Kesteren, Anne van: *Cross-Origin Resource Sharing*, Januar 2014. https://www.w3.org/TR/2014/REC-cors-20140116/.

[Kes16] Kesteren, Anne van: *Fetch*, August 2016. https://fetch.spec.whatwg.org/.

[LB15] Lemke, Claudia und Walter Brenner: *Mensch und Gesellschaft im digitalen Zeitalter*. In: *Einführung in die Wirtschaftsinformatik*, Seiten 53–87. Springer Berlin Heidelberg, 2015, ISBN 978-3-662-44064-3. http://dx.doi.org/10.1007/978-3-662-44065-0_3.

[Mas86] Mason, Richard O.: *Four Ethical Issues of the Information Age*. MIS Quarterly, 10(1):pp. 5–12, 1986, ISSN 02767783. http://www.jstor.org/stable/248873.

[MRJ11] Müller, Wolf, Jens Peter Redlich und Mathias Jeschke: *Auth*^{2}(nPA). Datenschutz und Datensicherheit - DuD, 35(7):465–470, 2011, ISSN 1614-0702. http://dx.doi.org/10.1007/s11623-011-0116-9.

[NS78] Needham, Roger M. und Michael D. Schroeder: *Using Encryption for Authentication in Large Networks of Computers*. Commun. ACM, 21(12):993–999, Dezember 1978, ISSN 0001-0782. http://doi.acm.org/10.1145/359657.359659.

[NS87] Needham, R M und M D Schroeder: *Authentication Revisited*. SIGOPS Oper. Syst. Rev., 21(1):7–7, Januar 1987, ISSN 0163-5980. http://doi.acm.org/10.1145/24592.24593.

[Oec02] Oecd: *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. Organisation for Economic Co-operation and Development, 2002. http://www.oecd.org/internet/ieconomy/privacy-guidelines.htm.

[OEC13] OECD: *The OECD Privacy Framework*, 2013. http://www.oecd.org/internet/ieconomy/privacy-guidelines.htm.

[PG07] Peter Gola, 1940: *Bundesdatenschutzgesetz : BDSG ; Kommentar*. München : Beck, München, 9., überarb. und erg. aufl. Auflage, 2007.

[RR06] Recordon, David und Drummond Reed: *OpenID 2.0: A Platform for User-centric Identity Management*. In: *Proceedings of the Second ACM Workshop on Digital Identity Management*, DIM '06, Seiten 11–16, New York, NY, USA, 2006. ACM, ISBN 1-59593-547-9. http://doi.acm.org/10.1145/1179529.1179532.

[RS02] Rosenberg, J. und H. Schulzrinne: *RFC 3264 An Offer/Answer Model with the Session Description Protocol (SDP)*, Juni 2002. https://tools.ietf.org/html/rfc3264.

[SNS88] Steiner, Jennifer G, B Clifford Neuman und Jeffrey I Schiller: *Kerberos: An Authentication Service for Open Network Systems.* In: *USENIX Winter*, Seiten 191–202, 1988.

[TMG07] *Telemediengesetz (TMG).* In: *Bundesgesetzblatt Teil I, Seite 179*, Band 6. Bundesanzeiger-Verlag, 28. Februar 2007. http://www.gesetze-im-internet.de/tmg/, Zuletzt geändert am 17.07.2015, BGBl. I S. 1324.

[UNG13] UNGA: *The right to privacy in the digital age*, Dezember 2013. http://www.un.org/ga/search/view_doc.asp?symbol=A/RES/68/167.

[W3C] W3C: *Open Web Platform*. https://www.w3.org/standards/.

[WBVS15] West, Mike, Adam Barth, Dan Veditz und Brandon Sterne: *Content Security Policy Level 2*, Juli 2015. http://www.w3.org/TR/2015/CR-CSP2-20150721/.

[Wes16] West, Mike: *Content Security Policy Level 3*, August 2016. https://www.w3.org/TR/2016/WD-CSP3-20160818/.

Alle Urls sind wurden zuletzt am 08.11.2016 geprüft.