Network Architectures
and Services
NET 2015-09-1

Proceedings

Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)

Summer Semester 2015

Munich, Germany

Editors

Georg Carle, Daniel Raumer, Lukas Schwaighofer

Publisher

Chair for Network Architectures and Services

Department of Informatics

Technische Universität München

Network Architectures
and Services
NET 2015-09-1

FI & IITM
SS 15

Proceedings zu den Seminaren Future Internet (FI) und Innovative Internet Technologien und Mobilkommunikation (IITM)

Sommersemester 2015

München, 19. 2. 2015 – 17. 07. 2015

Editoren: Georg Carle, Daniel Raumer, Lukas Schwaighofer

Organisiert durch den Lehrstuhl Netzarchitekturen und Netzdienste (I8),
Fakultät für Informatik, Technische Universität München

Technische Universität München

Proceedings of the Seminars
Future Internet (FI), and Innovative Internet Technologies and Mobile Communication Networks (IITM)
Summer Semester 2015

Editors:

Georg Carle
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
Technische Universität München
85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: http://www.net.in.tum.de/~carle/

Daniel Raumer
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
E-mail: raumer@net.in.tum.de
Internet: http://www.net.in.tum.de/~raumer/

Lukas Schwaighofer
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
E-mail: schwaighofer@net.in.tum.de
Internet: http://www.net.in.tum.de/~schwaighofer/

Cataloging-in-Publication Data

Seminars FI & IITM SS 15
Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internettechnologien und Mobilkkommunikation“ (IITM)
München, Germany, 19. 2. 2015 – 17. 07. 2015
ISBN: 978-3-937201-50-4

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2015-09-1
Lehrstuhl Netzarchitekturen und Netzdienste (I8) NET 2015-09-1
Series Editor: Georg Carle, Technische Universität München, Germany

Vorwort

Vor Ihnen liegen die Proceedings des Seminares „Future Internet“ (FI) und des Seminares „Innovative Internettechnologien und Mobilkommunikation“ (IITM). Wir sind stolz, Ihnen Ausarbeitungen zu aktuellen Themen, die im Rahmen unserer Seminare im Sommersemester 2015 an der Fakultät für Informatik der Technischen Universität München verfasst wurden, präsentieren zu dürfen. Den Teilnehmerinnen und Teilnehmern stand es wie in der Vergangenheit frei, das Paper und den Vortrag in englischer oder in deutscher Sprache zu verfassen. Dementsprechend finden sich sowohl englische als auch deutsche Paper in diesen Proceedings.

Unter allen Themen, die sich mit Aspekten der Computernetze von morgen befassen, verliehen wir wieder einen Best Paper Award. Dieser ging an Herrn Roman Trapickin, der in seiner Ausarbeitung „Who Is Scanning the Internet?“ der Fragestellung nachging, wer, mit welchem Interesse und welchen Werkzeugen das Internet scannt.

Einige der Vorträge wurden aufgezeichnet und sind auf unserem Medienportal unter `http://media.net.in.tum.de` abrufbar.

Im Seminar FI wurden Beiträge zu aktuellen Themen der Netzwerkforschung vorgestellt. Die folgenden Themen wurden abgedeckt:

- Vergängliche Kommunikation
- HTTP/2
- Dark Internet Mail Enviornment
- Privatsphäre im Netz
- Konzepte zur QoS-Verbesserung durch SDN
- Cloud-Gaming: ein Überblick
- Routing Caches in software-basierten Paketverarbeitungssystemen

Auf `http://media.net.in.tum.de/#%23Future%20Internet%23SS15` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Im Seminar IITM wurden Vorträge aus dem Themenbereich der Netzwerktechnologien inklusive Mobilkommunikationsnetze vorgestellt. Die folgenden Themen wurden abgedeckt:

- SDN zur DDoS Verteidigung
- Was ist Privatsphäre – Informationstheoretische Ansätze
- Maßnahmen zum Schutz der Privatsphäre
- Wer scannt das Internet?

Auf `http://media.net.in.tum.de/#%23IITM%23SS15` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Wir hoffen, dass Sie den Beiträgen dieser Seminare wertvolle Anregungen entnehmen können. Falls Sie weiteres Interesse an unseren Arbeiten habe, so finden Sie weitere Informationen auf unserer Homepage `http://www.net.in.tum.de`.

München, September 2015

Georg Carle

Daniel Raumer

Lukas Schwaighofer

Preface

We are pleased to present you the interesting program of our seminars on "Future Internet" (FI) and "Innovative Internet Technologies and Mobil Communication" (IITM) which took place in the summer semester 2015. In both seminar courses the authors were free to write their paper and give their talk in English or German.

We honored the best paper from both seminars with an award. This semester it was given to Mr Roman Trapickin who presented an overview to entities scanning the Internet, their motivation and the used tools in his paper "Who Is Scanning the Internet?".

Some of the talks were recorded and published on our media portal `http://media.net.in.tum.de`.

In the seminar FI we dealt with issues and innovations in network research. The following topics were covered:

- Ephemeral Communication
- HTTP/2
- Dark Internet Mail Environment
- Web Privacy
- Survey of Concepts for QoS improvements via SDN
- Gaming in the Cloud: a Survey
- Routing Caches in Software Packet Forwarding Devices

Recordings can be accessed on `http://media.net.in.tum.de/#%23Future%20Internet%23SS15`.

In the seminar IITM we dealt with different topics in the area of network technologies, including mobile communication networks. The following topics were covered:

- Leveraging SDN for DDoS defenses
- What is Privacy? – Information theory
- Methods of privacy preservation
- Who Is Scanning the Internet?

Recordings can be accessed on `http://media.net.in.tum.de/#%23IITM%23SS15`.

We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `http://www.net.in.tum.de`.

Munich, September 2015

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany (I8)

Seminarleitung

Daniel Raumer, Technische Universität München, Germany
Lukas Schwaighofer, Technische Universität München, Germany

Betreuer

Paul Emmerich (emmericp@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Oliver Gasser (gasser@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Benjamin Hof (hof@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Marcel von Maltitz (maltitz@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Johannes Naab (naab@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Daniel Raumer (raumer@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Quirin Scheitle (scheitle@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Lukas Schwaighofer (schwaigh@in.tum.de)
Technische Universität München, Mitarbeiter I8

Seminarhomepage

`http://www.net.in.tum.de/de/lehre/ss15/seminare/`

Inhaltsverzeichnis

Seminar Future Internet

Seminar Innovative Internet Technologien und Mobilkommunikation

Ephemeral Communication

Alexander Biro
Betreuer: Marcel von Maltitz
Seminar Future Internet SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: alexander.biro@tum.de

KURZFASSUNG

Im Zusammenhang mit digitaler Kommunikation und Datenverarbeitung ist der Schutz von Daten und privaten Informationen von allergrößter Wichtigkeit. Daten stellen ein wichtiges wirtschaftliches Gut dar: Nutzerdaten können für personalisierte Werbung genutzt werden und detaillierte Informationen über die Produkte anderer Unternehmen, um die Qualität eigener Produkte zu steigern. In dieser Arbeit wird der relativ neue Ansatz selbstlöschender Nachrichten zum Schutz von Daten anhand von bestehenden Diensten vorgestellt und kritisch bewertet. Ziel dieses neuen Ansatzes ist die Nachricht neben bekannten Angreifermodellen (externe Angreifer, Dienstanbieter) nun auch gegen den legitimen Empfänger der Nachricht zu schützen.

Ephemeral Communication stellt durch kurze Lebensdauern der Nachrichten einen sehr interessanten Ansatz zum Schutz der Nachrichten vor dem Empfänger dar. Jedoch ist dieser Ansatz praktisch nur sehr begrenzt umsetzbar, da die sensiblen Inhalte der Nachrichten mit Hilfe von Aufnahmen des Bildschirms, entweder durch Screenshots oder externe Kameras, permanent gespeichert werden können.

Schlüsselworte

Ephemeral Communication, Vergängliche Kommunikation, Selbstzerstörende Nachrichten, Datenschutz, Security, Privacy

1. MOTIVATION

Seit den Enthüllungen von Edward Snowden im Juni 2013 über die Spionage-Aktionen der amerikanischen National Security Agency (NSA) wurde in vielen Staaten vermehrt über Datenschutz diskutiert. [1] Der Schutz von Informationen und Daten beschränkt sich nicht nur auf Ärzte oder Banken, sondern ist für jede Organisation und auch besonders für jede Privatperson von allergrößter Bedeutung.

Beispielsweise ist Datenschutz für ein Softwareunternehmen zum Schutz der eigenen Produkte sehr wichtig. Falls z.B. der Quellcode dieser Produkte frei zur Verfügung stehen würde, wäre die gewinnbringende Vermarktung dieser Produkte schwierig.

Für Automobilhersteller ist auch der Schutz ihrer Konzepte und Ideen wichtig, anderenfalls könnte ein Konkurrent die Autos günstiger nachbauen und das ursprüngliche Unternehmen würde einen großen wirtschaftlichen Schaden davontragen.

Privatpersonen werden auf sozialen Internetplattformen wie Facebook, dazu angehalten möglichst viele persönliche Informationen in das private Profil einzupflegen. Ein Nutzerprofil wird dabei erst als vollständig betrachtet, sobald Informationen zum Wohnort, Beruf, Bildung und weiteres vorhanden sind. Auf Facebook werden täglich mehrere Milliarden Inhalte geteilt [2]. Wenn ein Nutzer hier ein ungünstiges Foto von sich selbst teilt, können sehr viele verschiedene Personen dies sehen. Auch dritte Personen, wie z.B. Arbeitgeber, können unter Umständen die persönlichen Bilder sehen und dadurch negativ beeinflusst werden. Es kann viele Situationen geben, bei denen geteilte Inhalte einer Person schaden können. Aus diesem Grund ist es ratsam nicht nur auf sozialen Plattformen, sondern generell sehr vorsichtig mit seinen privaten Daten und Informationen umgehen.

Dienste wie verschlüsselte E-Mails [3] oder Off-The-Record Messenger [5] bieten Funktionen an, um möglichst sicher miteinander zu kommunizieren. Allerdings können diese beiden Ansätze nicht vor dem Missbrauch der Nachricht durch den Empfänger schützen. Diese Arbeit beschreibt den neuen Ansatz “Ephemeral Communication” zum Schutz von privaten Daten und gibt Beispiele für bestehende Dienste. Anschließend werden die Grenzen und Schwierigkeiten dieser Technik genauer betrachtet sowie eine mögliche Umsetzung am Beispiel von Vanish [30] vorgestellt.

2. ANGREIFERMODELLE

Die möglichen Bedrohungen lassen sich in zwei Kategorien einteilen:

- Anbieter des Ephemeral Messaging Dienstes und externe Angreifer

  Versendete Nachrichten werden über die Server des Dienstes zum Empfänger transportiert. Oft erstellen die Server ein Protokoll und archivieren die Nachrichten. Die Anbieter des Dienstes können aus wirtschaftlichen Gründen ein hohes Interesse an den Inhalten versendeter Nachrichten besitzen. Durch die Inhalte der Nachrichten können diese auf die Vorlieben und Gewohnheiten ihrer Nutzer schließen und ihnen personalisierte Werbung und Angebote bieten. Die Anbieter haben oft viele Mitarbeiter aber kein Interesse daran, einem Nutzer direkt zu schaden.

  Externe Angreifer sind Personen, welche sich für die Inhalte der Nachrichten aus persönlichen oder krim-

doi: 10.2313/NET-2015-09-1_01

inellen Gründen interessieren. Diese Angreifer können beispielsweise den Netzwerkverkehr ihrer Opfer mitschneiden (wenn beide sich im selben Netzwerk befinden) um die Nachrichten abzufangen. Diese Personen nutzen die erhaltenen Informationen um ihrem Opfer aktiv zu schaden.

- Empfänger der Nachricht

  Selbst der legitime Empfänger der Nachricht könnte zu einer potentiellen Bedrohung werden. Ein Beispiel-Szenario dazu: Nutzer A schickt eine Nachricht mit einer Aussage, welche ihm zu einem späteren Zeitpunkt schaden könnte, an einen Nutzer B. Somit hat nun B einen Beweis für die Aussage von A und könnte diese später nutzen um A zu schaden. Hätte A seine Aussage in einem natürlichen Gespräch zu B geäußert, hätte B keinen Beweis gegen A.

  Zu dem Zeitpunkt, an welchem eine Nachricht verschickt wird, hat der Empfänger vielleicht noch kein Interesse daran dem Sender damit zu schaden. Der Empfänger hat allerdings die Möglichkeit die Nachricht solange zu speichern und aufzubewahren bis die Nachricht dem Sender schaden kann. Der Sender hat keinen Einfluss darauf und kann die dauerhafte Speicherung der Nachricht nicht verhindern.

  Im folgenden wird nun hauptsächlich der zweite Typus Angreifer (Empfänger) behandelt. Der Schutz vor dem ersten Typus Angreifer lässt sich bereits mit etablierten Methoden (z.B. PGP und Off-The-Record Messaging) bereitstellen.

3. WARUM WIRD EPHEMERAL MESSAGING BENÖTIGT?

In diesem Kapitel werden, anhand des im Kapitel 2 bestimmten Angreifermodells, die Grenzen von verschlüsselten E-Mails und Off-The-Record (OTR) Messengern zum Schutz der Privatsphähre dargestellt. Anschließend wird der neue Ansatz “Ephemeral Communication” zum besseren Schutz persönlicher Daten vor dem Empfänger vorgestellt.

Verschlüsselte E-Mails werden oft mit dem freien Pretty-Good-Privacy (OpenPGP) [4] Standard verschlüsselt. PGP verwendet eine asymmetrische Verlüsselung, deswegen muss der Sender dem Empfänger kein Passwort zum Öffnen der Nachricht mitteilen. Die Nachrichten werden mit der digitalen Signatur des Senders authentifiziert und mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Die Nachricht wird direkt am Gerät des Absenders verschlüsselt und in diesem Zustand über die Server des Mail-Anbieters zum Empfänger transportiert. Nur der Besitzer des privaten Schlüssels des Empfängers ist in der Lage die Nachricht zu entschlüsseln. Durch die digitale Signatur kann sich der Empfänger sicher sein, dass die Nachricht nicht gefälscht wurde und tatsächlich vom richtigen Sender stammt. [3]

Niemand bis auf den legitimen Empfänger, kann den Inhalt Nachricht der Nachricht lesen. Der Inhalt der Nachricht ist vor den Anbietern des E-Mail-Dienstes und weiteren Angreifern geschützt. Einer Nachricht werden aus technischen Gründen zum Versand unverschlüsselte Meta-Daten angehängt. In den Meta-Daten einer Nachricht stehen beispielsweise der Absender, der Empfänger, die Zeit des Versands und der Betreff der Nachricht. Anbieter und Angreifer können sehen, dass es zu einer Kommunikation gekommen ist, wissen wer die beiden beteiligten Kommunikationspartner sind, aber nicht über was sie schreiben.

Dieser Ansatz schützt gut gegen Anbieter und Angreifer, aber nicht vor dem Empfänger. Der Empfänger könnte die Nachricht, bewusst oder unbewusst durch einen Mail-Client in entschlüsselter Form auf seinem Computer speichern. Die Nachricht könnte anschließend in falsche Hände gelangen, wenn jemand Zugriff auf den Computer besitzt. Mit Hilfe von Schadsoftware (z.B. Computerviren) könnte diese Nachricht auch automatisch an einen Angreifer weitergeleitet werden. Da der Absender einer verschlüsselten Nachricht durch seine digitale Signatur bekannt ist, kann die Information direkt einer Person zugeordnet werden und er kann den Versand der Nachricht nicht abstreiten. Allerdings besteht auch die Möglichkeit, dass der Empfänger direkt zum Angreifer wird. Er kann die Informationen aus der Nachricht speichern und zu einem späteren Zeitpunkt gegen den Sender einsetzen.

OTR-Messaging wurde entwickelt um vertrauliche und verschlüsselte Kommunikation bei Instant Messaging bereitzustellen. Zusätzlich kann nach einem Gespräch nicht bewiesen werden, dass ein Nutzer eine gewisse Nachricht gesendet hat oder nicht. Das heißt, dass der Versand einer Nachricht bei OTR-Messaging im Nachhinein immer abgestritten werden kann. OTR baut dabei auf die folgenden vier großen Prinzipien: [5]

- Verschlüsselung: Während eines Gespräch kann niemand den Inhalt der Nachrichten mitlesen, denn die Nachrichten werden Ende-zu-Ende verschlüsselt.
- Authentifizierung: Zu Beginn von jedem Gespräch wird mit Hilfe des Digital-Signature-Algorithm (DSA) sichergestellt, dass die Person mit der kommuniziert wird, tatsächlich die ist, für die sie gehalten wird. Die Authentifizierung findet nur am Beginn des Gesprächs mit digitalen Signaturen statt und verwendet den Diffie-Hellman (DH) Schlüssel-Aus-tausch zur Berechnung eines gemeinsamen Schlüssels.
- Abstreitbarkeit: Die beiden Kommunikationspartner besitzen jeweils einen langlebigen, öffentlichen DSA-Schlüssel zur gegenseitigen Verifizierung. Nach der anfänglichen Authentifizierung durch digitale Signaturen, werden folgende Nachrichten, aufgrund der Abstreitbarkeit, mit Message Authentication Codes (MAC's) authentifiziert. Gesendeten Nachrichten wird immer die MAC des Senders beigefügt. Nach dem Versand einer Nachricht berechnet der Sender einen neuen privaten Schlüssel, so dass beide Kommunikationspartner sich gegenseitig immer nur ”alte” Schlüssel veröffentlichen. Der Empfänger der Nachricht kann sicher sein, dass die Nachricht vom richtigen Sender stammt, da für die Berechnung der MAC-Schlüssel der gemeinsame Schlüssel aus dem DH-Schlüssel-Austausch benötigt wird. Gegenüber Fremden kann nicht bewiesen werden, ob eine Nachricht von einem bestimmten Absender stammt, da beide Kommunikationspartner in der Lage waren die MAC's mit Hilfe des gemeinsamen Schlüssels zu errechnen.

doi: 10.2313/NET-2015-09-1_01

- Folgenlosigkeit: Falls es einem Angreifer gelingen sollte den privaten Schlüssel eines Nutzers herauszufinden, kann dieser damit nicht die Nachrichten vergangener Gespräche entschlüsseln. Versendete Nachrichten werden bei OTR nicht mit dem privaten Schlüssel eines Nutzers authentifiziert oder verschlüsselt. Allerdings ist der Angreifer nun in der Lage gefälschte Nachrichten unter dem Namen des Nutzers zu verbreiten. Sobald der Nutzer den Verlust seines privaten Schlüssels mitteilt, verliert die digitale Signatur ihre Gültigkeit und der Angreifer kann keine Nachrichten mehr fälschen.

Durch die Funktionen von OTR lässt es sich bereits sehr sicher im Internet kommunizieren, jedoch gibt es auch Informationen, welche auch ohne die Zuordnung zu einer Person oder einer Sache wertvoll sind. Ein Beispiel dazu wäre folgendes Szenario: Eine Person versteckt sehr viel Geld an einem geheimen Ort für eine zweite Person und verschickt diese Information mit einem OTR-Messenger. Wenn jemand drittes den Inhalt der Nachricht herausfindet, kann dieser die Information auch ohne Bezug zu den Gesprächspartnern nutzen.

Um versendete Nachrichten besser gegen den Empfänger zu schützen wurde der Ansatz "Ephemeral Communication" entwickelt. Mit diesem Ansatz versendete Nachrichten besitzen eine begrenzte, vom Sender bestimmte Lebensdauer. Mit Ablauf dieser Lebensdauer wird die Nachricht dauerhaft gelöscht. Selbst der Empfänger der Nachricht, die Anbieter des Dienstes oder dritte Angreifer haben nach dieser Zeit keine Möglichkeit mehr die Nachricht zu lesen.

Ephemeral Messaging ist das digitale Pendant zu natürlichen, analogen Gesprächen. Alle Sicherheitseigenschaften natürlicher Gespräche sollen bei dem digitalen Gegenstück auch umgesetzt werden. Zu diesen Eigenschaften zählt, dass kein Gesprächsprotokoll erstellt wird. Weiterhin soll es nicht möglich sein im Nachhinein zu beweisen, dass eine Person eine bestimmte Aussage gemacht hat. Darüber hinaus kann, im Gegensatz zu natürlichen Gesprächen, bei Ephemeral Messaging mit Hilfe einer sicheren Ende-zu-Ende Verschlüsselung sichergestellt werden, dass niemand fremdes mitlesen kann. [6]

Ziel von Ephemeral Communication ist der Schutz privater Daten und Nachrichten durch Inexistenz nach einer gewissen Zeit. "Private Daten und Informationen können am besten geschützt werden, wenn keine zu schützenden Daten existieren" [6].

Das folgende Beispiel soll eine mögliche Anwendung von Ephemeral Messaging darstellen. Die Ehefrau A möchte sich von ihrem Ehemann B scheiden lassen und kommuniziert via E-Mail deswegen mit ihrem Scheidungsanwalt C. A und B nutzen Zuhause gemeinsam einen Computer. B könnte sich auf verschiedenen Wegen (z.B. A meldet sich im Postfach an und verlässt später eventuell das Zimmer) Zugang zu dem E-Mail Postfach von A verschaffen. A möchte sicher sein, dass ihr Nachrichtenverlauf mit C sicher ist und B auf keinen Fall mitlesen kann. Mit verschlüsselten E-Mails könnten die Nachrichten nicht geschützt werden, denn diese sind nach der Anmeldung am Postfach, frei zugänglich. Mit Hilfe von OTR-Messengern kann der Versand einer Nachricht abgestritten werden, jedoch aber nicht, dass eine Kommunikation stattgefunden hat. B könnte hier die Nachrichten lesen, aber nicht beweisen, dass A eine gewisse Nachricht gesendet hat. Durch Ephemeral Messaging könnte die Lebensdauer der Nachrichten auf beispielsweise 15 Minuten nach Abruf festgelegt werden. Diese Zeitspanne reicht aus, damit A ihre Nachrichten lesen und antworten kann. Nach dem Ablauf dieser Zeit vernichten sich alle Spuren dieser Kommunikation von selbst und B hat keine Möglichkeit die Nachrichten mitzulesen und etwas über das Vorhaben von A in Erfahrung zu bringen.

4. ÜBERBLICK ÜBER BESTEHENDE EPHEMERAL MESSAGING DIENSTE

In diesem Kapitel werden bekannte Ephemeral Messaging Dienste mit ihren wichtigsten Funktionen vorgestellt und kritisch beleuchet:

4.1 Snapchat

2011 wurde mit Snapchat die erste Anwendung für mobile Geräte veröffentlicht, welche es dem Nutzer erlaubt, selbstzerstörende Nachrichten zu versenden. Die Anwendung wurde von zwei Informatikstudenten (Evan Spiegel und Bobby Murphy) der Universität Stanford aus den USA programmiert. [9] Vor dem Versenden einer Nachricht entscheidet der Sender, wie lange sie auf dem Gerät des Empfängers angezeigt werden darf. Als Lebensdauer sind nur Werte von einer bis zehn Sekunden möglich. Nach Ablauf der festgelegten Zeit werden die Nachrichten selbstständig von beiden Geräten gelöscht und sind von nun an nicht mehr abrufbar. Snapchat hat dabei den Fokus nicht auf das Versenden von Textnachrichten, sondern von Bildern und kurzen Videos ("Snaps") gelegt. Die App ist für die beiden beliebtesten Betriebssysteme mobiler Geräte (Android und iOS) verfügbar. Heute ist Snapchat mit insgesamt über 100 Millionen Nutzern [10] und 350 Millionen versendeten Snaps pro Tag [11] die beliebteste Anwendung im Ephemeral Messaging Bereich. Bei Snapchat werden alle Nachrichten unverschlüsselt versendet und die Meta-Daten bleiben auch nach dem Ablauf der Lebensdauer erhalten und werden auf den Servern des Anbieters gespeichert. Des Weiteren gibt es die sogenannte "Story"-Funktion, welche es den Nutzern erlaubt persönliche Informationen wie Bilder für 24 Stunden im eigenen Profil zu veröffentlichen, die "Replay"-Funktion um einen Snap pro Tag nach Timeout nochmal sehen zu können und "Discover" um aktuelle Nachrichten zu lesen oder neue Nutzer zu finden. Wenn ein Nutzer einen Screenshot von einer empfangenen Nachricht erstellt, wird dies dem Sender umgehend mitgeteilt. [7] Bei der Anmeldung muss die E-Mail Adresse sowie die Telefonnummer angegeben werden. Beide Informationen werden genutzt um weitere Kommunikationspartner zu finden. [8]

Die fehlende Ende-zu-Ende Verschlüsselung von Snapchat lässt jeden Angreifer und den Anbieter ungehindert bei jeder Nachricht mitlesen und bietet deswegen keinen Schutz gegenüber diesem Angreifermodell. Durch die Replay- und Story-Funktion verliert Snapchat größtenteils die Vergänglichkeit der Nachrichten. Aufgrund der Verletzung der Vergänglichkeit liefert Snapchat nur einen sehr geringen Schutz gegen den Empfänger. Wegen diesen Tatsachen ist Snapchat nicht zum Versand sensibler Botschaften geeignet.

doi: 10.2313/NET-2015-09-1_01

4.2 Ansa

Ansa bietet im Gegensatz zu Snapchat eine vollständige Ende-zu-Ende Verschlüsselung, so dass selbst die Anbieter des Dienstes nicht den Inhalt der Nachrichten sehen können. Bei Ansa können Nachrichten entweder im normalen Modus oder im "Off-The-Record"-Modus mit Selbstzerstörung versendet werden. [14] Das Timeout kann hier nicht frei eingestellt werden und liegt immer bei 60 Sekunden. Versehentlich versendete Nachrichten können im Nachhinein durch "Remote Deletion" wieder vom Gerät des Empfängers gelöscht werden. Diese Funktion heißt bei Ansa "Synced Deletion" [12]. Diese Funktion ermöglicht es Nutzern versendete Nachrichten direkt nach dem Versand zu löschen, so dass der Empfänger keine Chance hat diese zu lesen. Nachrichten können aber auch erst nach einigen Minuten oder Stunden gelöscht werden, solange die Nachricht noch nicht durch den Empfänger geöffnet wurde und wegen ihrer begrenzten Lebensdauer bereits gelöscht wurde. Weiterhin wird der Sender einer Nachricht auch bei Ansa benachrichtigt, wenn der Empfänger davon einen Screenshot erstellt hat. Ansa ist bisher, nur für Android und iOS verfügbar. Der Ansa Account wird, wie bei Snapchat, an die E-Mail Adresse sowie an die Telefonnummer gekoppelt. [13]

Ansa bietet seinen Nutzern durch eine sichere Ende-zu-Ende Verschlüsselung und Synced Deletion ein hohes Maß an Sicherheit an. Die Anwendung besitzt alle wichtigen Funktionen um die Privatsphähre möglichst gut zu wahren. Screenshots und Bildschirmaufnahmen sind allerdings auch bei Ansa ein großes Sicherheitsproblem. Darüber hinaus können Nutzer auch Nachrichten verschicken, welche sich nicht automatisch löschen.

4.3 Wickr

Wie auch Ansa, bietet Wickr seinen Nutzern eine verlässliche Ende-zu-Ende Verschlüsselung an. [17] Bei Wickr können Nach-richten nur mit einer begrenzten Lebensdauer versendet werden, aber diese ist frei wählbar und lässt sich maximal auf sechs Tage nach dem ersten Öffnen beim Empfänger einstellen. [16] Die Anbieter des Dienstes versprechen, dass keine Meta-Daten der versendeten Nachrichten gespeichert werden. Dieses Versprechen kann jedoch nicht überprüft werden und stellt keine Garantie für die Löschung der Meta-Daten dar. [15] Wickr speichert auch keine Passwörter auf ihren Servern, deswegen kann dieses bei Verlust auch nicht wieder zurückgesetzt werden und das Konto ist nicht mehr zugänglich. Zum Nutzen der Anwendung muss der Nutzer ein Passwort vergeben, dieses wird allerdings nur zum lokalen Starten von Wickr benötigt. [15] Eine weitere Funktion ist, die Remote Deletion wie bei Ansa. Weiterhin besitzt auch Wickr wie Ansa und Snapchat, die Benachrichtigungsfunktion bei Screenshots. Die Messaging Anwendung ist bereits für fast alle Plattformen verfügbar: Windows Desktop, OS X, Linux (32 und 64 Bit), Android und iOS. Bei der Anmeldung fÃijr Wickr, wird weder die E-Mail Adresse noch die Telefonnummer mit dem Account gekoppelt. Diese Einstellung lässt sich später optional vornehmen. [18]

Wickr verspricht seinen Nutzern keine persönlichen Informationen wie echte Namen, Passwörter der Meta-Daten zu speichern. Ob dieses Versprechen wirklich eingehalten wird, lässt sich nicht überprüfen. Nutzer können standardmäßig nur durch ihren Wickr-Benutzernamen gefunden werden und sind so relativ sicher. Da Wickr auf so vielen Plattformen verfügbar ist, besteht das Risiko, dass Nachrichten am Computer geöffnet werden und dort gespeichert werden. Am Computer gibt es einfachere Möglichkeiten den Inhalt des Bildschirms zu speichern, so kann z.B. ein Video der Bildschirmausgabe gespeichert werden.

4.4 Frankly Messenger

Der Frankly Messenger bietet keine Ende-zu-Ende Verschlüsselung an, besitzt aber eine Vielzahl an sozialen Funktionen. Zu diesen sozialen Funktionen zählen die Gruppen-Chat-Funktion, Kopplung der App mit Facebook und die Möglichkeit an Kontakte, welche die Anwendung nicht besitzen zu schreiben. Wenn ein Nutzer von Frankly Messenger eine Nachricht mit einer begrenzten Lebensdauer an einen Kontakt, außerhalb von Frankly Messenger (z.B. via E-Mail) schickt, soll sich diese auch automatisch löschen. [21] Wie die Löschung der Nachricht bei dem Versand einer Nachricht zwischen mehreren Diensten funktioniert, wird allerdings nicht von Seiten der Anbieter von Frankly Messenger beschrieben. Aus diesem Grund wird diese Funktion nicht weiter behandelt.

Versendete Nachrichten löschen sich selbst innerhalb von 10 Sekunden nach Abruf, diese Zeitspanne kann nicht verkürzt oder verlängert werden. Zudem ist es auch möglich einzelne, versendete Nachrichten vor der Löschung zu bewahren, indem man den Button mit der Stecknadel beim Erstellen einer Nachricht antippt. Frankly Messenger besitzt zusätzlich auch die Remote Deletion um nachträglich die Nachrichten zu löschen, welche zuvor von der Löschung ausgenommen wurden. Der Gruppenchat innerhalb von Frankly Messenger bietet zusätzlich die Funktion nur die Inhalte der Nachrichten und nicht die Sender anzuzeigen. Das heißt, dass kein Teilnehmer des Gruppenchats genau sagen kann welcher Teilnehmer welche Nachricht gesendet hat. [20] Die App ist für Android und IOS verfügbar. Beim Registrierungsprozess wird der Account mit der Telefonnummer verbunden und jeder Nutzer erhält eine eigene und einzigartige PIN. Die Kopplung mit der E-Mail-Adresse ist optional. Neue Kontakte werden mit Hilfe ihrer PIN hinzugefügt. [19]

Die fehlende Ende-zu-Ende Verschlüsselung schadet der Privatssphähre der Nutzern sehr und ermöglicht Nachrichten im Rahmen des ersten Angreifermodells mitzulesen. In Gruppen-Chats geteilte Informationen sind stärker gefährdet als normale Chats, da hier mehrere Personen theoretisch Screenshots erstellen könnten. Durch die Anonyme Gruppenchat-Funktion kann der Versand einer Nachricht nicht einem gewissen Nutzer zugeschrieben werden. Diese Funktion ähnelt der Funktion von OTR-Messengern.

4.5 Privatext

Privatext ist eine minimalistische Ephermeral Messaging Anwendung mit Ende-zu-Ende Verschlüsselung für Android und IOS. Die App kann mit einem Passwort versehen werden, so dass niemand sonst Zugriff auf die privaten Nachrichten erhält. [23] Die Lebensdauer der Nachrichten kann zwischen 30 Sekunden und 24 Stunden frei eingestellt werden. Weiterhin kann eingestellt werden, ob die Zeit ab dem Versand der Nachricht anfängt zu laufen oder erst sobald der Empfänger

diese geöffnet hat. Bei den anderen vorgestellten Diensten beginnt die Zeit erst ab dem öffnen der Nachricht zu laufen. Eine weitere Funktion von Privatext ist "Confirmation Texting", vor dem Absenden einer Nachricht fragt die Anwendung den Nutzer, ob dieser auch sicher den richtigen Kontakt für die Nachricht gewählt hat. Damit soll verhindert werden, dass private Nachrichten versehentlich an die falsche Person gesendet werden. [22, 24] Zudem besitzt auch Privatext eine Remote Deletion Funktion und die Benachrichtigung des Senders bei Screenshots seiner Nachrichten. Der Privatext Account wird nur mit der E-Mail Adresse und einem beliebigen Nickname gekoppelt.

Privatext bietet die Möglichkeit, die Lebenszeit der Nachricht gleich nach dem Absenden zu starten. Außerdem kann die kleine Funktion "Confirmation Texting" verhindern, dass private Informationen aus versehen an falsche Personen gesendet werden. Weiterhin können Nutzer auch den Start der Anwendung mit einem Passwort schützen, sodass wirklich nur der legitime Empfänger in der Lage ist die Nachrichten zu lesen.

Alle vorgestellten Apps sind kostenlos erhältlich und bieten im Kern dieselbe Funktionalität an. Große Unterschiede gibt es hinsichtlich der Frage, ob Verschlüsselung verwendet wird und bei der Länge der Lebensdauer der Nachrichten. Im Augenblick gibt es lediglich für Snapchat aufgrund der großen Beliebtheit Fake-Clients. Wahrscheinlich ist es theoretisch möglich für viele Dienste Fake-Clients zu programmieren um damit die Selbstzerstörung der Nachrichten zu umgehen. Die genaue Funktionsweise der vorgestellten Dienste ist nicht bekannt und aus diesem Grund ist der Nutzer der Anwendungen gezwungen dem Anbieter zu vertrauen.

5. GRENZEN UND SCHWIERIGKEITEN

Ephemeral-Messaging-Dienste unterscheiden sich durch die begrenzte Lebensdauer der Nachrichten am deutlichsten von herkömmlichen Messaging-Anwendungen. Hauptziel des Ephemeral Communication Ansatzes ist, dass die Nachrichten unwiederbringlich nach dem Timeout gelöscht werden und die Informationen nicht gespeichert und anschließend weitergegeben oder missbraucht werden können. Dieses Ziel zu erreichen ist überaus kompliziert, denn es existieren mehrere Sicherheits-Probleme welche eine permanente Speicherung ermöglichen.

Beispielsweise können Nachrichten von Snapchat auf verschiedene Wege dauerhaft gespeichert werden. Mit Hilfe einer veränderten Snapchat App (z.B. SaveMySnaps), einem Fake Client können noch ungelesene Nachrichten aus der originalen Snapchat App, ohne Zeitlimit betrachtet und anschließend gespeichert werden. Das funktioniert wie folgt: Der "Angreifer" meldet sich im Fake Client mit den gleichen Log-In Daten, wie in der originalen Anwendung an und findet dort ein ähnliches Interface. Der Sender der Nachricht wird bei einer Speicherung durch SaveMySnaps [25] standardmäßig nicht benachrichtigt und erhält nicht einmal eine Lesebestätigung. SaveMySnaps bietet aber den Versand von Lesebestätigungen an, damit der Absender keinen Verdacht schöpft. [26, 27] Alternativ lassen sich auch bereits geöffnete und durch Snapchat gelöschte Snaps durch Datenwiederherstellungstools sichern. Eine App mit dieser Funktion unter Android heißt Dumpster. [28]

Da es bei Snapchat so viele Möglichkeiten gibt die automatische Selbstzerstörung der Nachrichten zu umgehen, sollten sicherheitsbewusste User zu alternativen Anwendungen greifen. In anderen Anwendungen wie z.B. Wickr, Ansa oder Frankly Messenger wurden noch keine Sicherheitslücken gefunden, welche eine ähnlich einfache Speicherung erlauben. Die versendeten Nachrichten sind allerdings auch hier nicht vollkommen sicher, da der Empfänger sie jederzeit durch Screenshots aufzeichnen kann. Screenshots werden bei vielen Herstellen durch das Drücken einer bestimmten Tastenkombination (z.B. Home-Taste + Ein/Ausschalter bei Samsung) erstellt. Um die Nutzer daran zu hindern Screenshots zu erstellen, müssen diese beim Betrachten von Bildern, mit einem Finger auf den Bildschirm tippen. Das Bild verschwindet entweder nach Ablauf der Zeit oder sobald der Bildschirm nicht mehr berührt wird.

Diese Methode zum verhindern von Screenshots wird "tap-to-view" genannt. Tap-to-view schützt selbstverständlich nicht gänzlich vor Screenshots, aber dadurch werden eventuell weniger Screenshots gemacht. Auf "gerooteten" Android Geräten ist es mit der App "Screenshot HD" [29] möglich Screenshots zeitgesteuert oder durch das Schütteln des Geräts zu erzeugen. "Gerootet" bedeutet, dass der Nutzer Admin-Rechte auf seinem System besitzt. Im Auslieferungszustand haben Android Nutzer keinen administrativen Zugriff auf das System.

Noch einfacher als durch Screenshots, kann man sich durch das Anfertigen einer Fotografie oder eines Films vom Bildschirminhalt des Geräts, während die Nachricht geöffnet ist, gegen über die Selbstlöschung hinwegsetzen. Bei dieser Art des Angriffs wird das sogenannte "analog hole" ausgenutzt. Vor der Ausnutzung des analog hole können sich die Anbieter der Dienste nur sehr schwer schützen. Da der Bildschirm angetippt werden muss, ist wenigstens nicht der gesamte Bildschirm auf dem Foto oder Film erkennbar.

Schwierigkeiten gibt es zusätzlich, wenn die Anwendung auf einer Vielzahl an unterschiedlichen Plattformen funktionieren soll. Die Anwendung Wickr kann beispielsweise auf Windows Phone, Windows Desktop, Mac OS X, Linux, Android und IOS genutzt werden. Auf Android kann die Screenshot-Funktion für die Anwendung deaktiviert werden, bei allen anderen Plattformen aber (noch) nicht. Bei Desktop Systemen gibt es sehr viele Methoden die Bildschirmausgabe zu speichern. Jede Plattform besitzt eigene Besonderheiten und Funktionen und macht es den Entwicklern sehr schwer Probleme einheitlich zu lösen.

6. FUNKTIONSWEISE SELBSTLÖSCHENDER NACHRICHTEN AM BEISPIEL VON VANISH

Die meisten Anwendungen veröffentlichen aus sicherheitstechnischen oder wirtschaftlichen Gründen ihren Programmcode nicht. Aus diesem Grund ist es für Außenstehende nicht einfach herauszufinden wie eine Applikation funktioniert. Die genaue Funktionsweise von Closed-Source Software bleibt so geheim.

Im folgenden wird Vanish [30], eine quelloffene Software der University of Washington, vorgestellt. Da bei Vanish der Quellcode der Anwendung frei zugänglich ist, kann die genaue Funktionsweise der Anwendung nachvollzogen werden. Vanisch erlaubt mit Hilfe von verteilten Hash-Tabellen, Nachrichten mit einer festgelegten Lebensdauer zu verschicken. Verteilte Hash-Tabellen (Englisch: Distributed Hash-Table, DHT) werden genutzt, um den Ablageknoten einer Datei in einem Peer-to-Peer Netzwerk zu speichern. Nach Ablauf der festgelegten Lebensdauer der Nachrichten wird die Hash-Tabelle die entsprechenden Einträge entfernen und die Knoten werden die Daten unwiderruflich löschen.

Bei Vanish besteht ein anderes Angreifermodell. Hier halten sich die Empfänger der Nachrichten an das Protokoll und haben kein Interesse an der dauerhaften Speicherung von Nachrichten. Bei Vanish ist das Ziel sich gegen einen externen Angreifer zu verteidigen, welcher sich Zugriff zum Endgerät des legitimen Empfängers verschafft um Nachrichten mitzulesen und zu speichern. (Vergleiche Beispiel zur Scheidung des Ehepaares am Ende des 3. Abschnitts)

Im Folgenden wird das Vorgehen bei der Ver- und Entschlüsselung von Vanish erklärt:

Als erstes wird ein zufälliger Schlüssel K erzeugt, um die zu versendenden Daten D zu verschlüsseln. Hierfür wird eine symmetrische Verschlüsselung genutzt, damit das Verschlüsselte Datenobjekt (Ciphertext) C mit dem gleichen Schlüssel K wieder entschlüsselt werden kann. Dem Nutzer wird dieser Schlüssel nicht mitgeteilt und bleibt geheim.

Dieser geheime Schlüssel K wird nun durch die Anwendung des Secret Sharing Algorithmus von Adi Shamir [32] in N Teile bzw. Shares $(K_1, K_2, \ldots, K_N)$ geteilt. Bei Vanish liegt der Schwellwert standardmäßig bei 90%, das heißt, dass mindestens 90% aller Shares benötigt werden um den ursprünglichen Schlüssel K zu errechnen. Falls mehr als 90% der Shares verloren gehen, kann der Schlüssel nicht mehr errechnet werden und die Daten können nicht entschlüsselt werden. Der Schwellwert kann auch frei eingestellt werden. Je höher dieser Wert eingestellt ist, umso schneller kann der ursprüngliche Schlüßel nicht mehr berechnet werden und die Information ist dauerhaft verloren.

Die errechneten N Shares sollen nun auf die Knoten eines weltweiten Peer-to-Peer (P2P) Netzwerkes verteilt werden. Zur Bestimmung welcher Share des Schlüssels K auf welchen Knoten des P2P-Netzwerks gespeichert wird, generiert Vanish einen zufälligen "Zugriffsschlüssel" L. Aus dem Zugriffsschlüssel werden die Adressen der Knoten, auf denen die Shares gespeichert werden, bestimmt. Die Schlüsselteile (Shares) $(K_1, \ldots, K_N)$ werden an den Positionen, welche durch die Zugriffsschlüssel $(L_1, \ldots, L_N)$ bestimmt werden, in eine verteilte Hash-Tabelle geschrieben. Die Zugriffsschlüssel L werden anschließend alle im Vanishing Data Object (VDO) gespeichert, um später dem Empfänger mitzuteilen wo die Shares gefunden werden können. Beispielsweise wird das Schlüsselteil K_1 an der Adresse L_1 bzw. K_N an der Stelle L_N des P2P-Netzwerks gespeichert.

Abbildung 1 [30] veranschaulicht wie die Schlüsselteile $K_{1,\ldots,N}$ ent-sprechend dem Zugriffsschlüssel L innerhalb der DHT

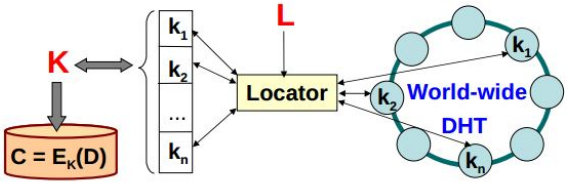

Abbildung 1: Schematischer Ablauf

abgespeichert werden. Die Schlüsselteile $K_{1,\ldots,N}$ werden benötigt um den symmetrischen Schlüssel K zur Entschlüsselung der verschlüsselten Daten C zu errechnen.

Die verschlüsselten Daten C werden mit der Anzahl der Schlüsselteile N, den Zugriffsschlüsseln $(L_1, \ldots, L_N)$ und dem Schwellwert zu einem VDO-Object zusammengefasst. Dieses VDO kann anschließend via Mail oder Messenger verschickt werden und ist bis zu einem festgelegten Timeout (beginnend ab Versand der Nachricht) lesbar. Das VDO besteht folglich aus (L, C, N, Schwellwert). Nach dem Time-out wird die DHT die Shares verwerfen und die Datei kann nicht mehr entschlüsselt werden. In die DHT werden immer Tupel bestehend aus den Shares des Schlüssels und einem Wert für ihr Time-out eingefügt. Die DHT durchsucht permanent alle ihre Einträge auf abgelaufene Time-outs und entfernt diese.

Um die Daten D eines VDOs vor dem Ablauf der Lebensdauer lesen zu können müssen als erstes die Zugriffsschlüssel $(L_1, \ldots, L_N)$ aus dem VDO extrahiert werden. Anschließend werden die Shares $(K_1, \ldots, K_N)$ an den Stellen, welche durch die Zugriffsschlüssel angegeben sind, zu dem Schlüssel K zusammengesetzt. Der Schwellwert gibt hier genau an wie viele Shares benötigt werden um K zu errechnen. Mit diesem Schlüssel können nun die verschlüsselten Daten C entschlüsselt und gelesen werden. [30]

Vanish nutzt OpenDHT als P2P-Netzwerk. OpenDHT ist eine öffentliche, verteilte Hashtabelle, das heißt, dass jeder dem Netzwerk beitreten kann. Wenn nun ein Angreifer sehr viele Knoten in dieses Netzwerk einbringt, besteht die Gefahr, dass die Shares beim Verteilen auch auf diese Knoten gespeichert werden. Der Angreifer kann so eventuell alle Shares sammeln und die Daten längerfristig speichern. Durch ein Erhöhen des Schwellwerts zur Rekonstruktion des Schlüssels K kann man dieser Art des Angriffs entgegen wirken. Zusätzlich kann auch die Anzahl N der Shares erhöht werden um die Kosten dieses Angriffs weiter zu steigern. Es besteht jedoch weiter die Gefahr, dass zufällig die benötigte Anzahl an Shares auf den Knoten des Angreifers gespeichert werden. Die genannten Maßnahmen können die dauerhafte Selbstzerstörung der Daten zwar nicht garantieren, aber sie steigern drastisch den Aufwand und die Kosten für den Angreifer. [31]

7. SCHLUSSFOLGERUNGEN

Von allen vorgestellten Anwendungen ist Wickr die einzige, welche die Aufnahme des Bildschirminhalts mittels Screenshots auf Android blockiert. Weiterhin gibt der Anbieter des Dienstes das Versprechen, alle Meta-Daten der versendeten

Nachrichten zu löschen und das alle gelöschten Informationen nicht mehr wiederhergestellt werden können. Es kann allerdings nicht überprüft werden, ob tatsächlich alle Spuren des Nachrichtenaustauschs gelöscht werden. Aufgrund der Ende-zu-Ende-Verschlüsselung aller Nachrichten ist Wickr trotzdem mein persönlicher Favorit unter den vorgestellten Anwendungen. Die Anwendung Privatext kann ich ebenso empfehlen, da sie viele sinnvolle Funktionen zum Schutz der Nachrichten bietet, wie z.B. Confirmation Texting und die Passwortsperre für das Starten der Applikation. Die Nutzung von Snapchat kann ich, aufgrund der vielen Möglichkeiten (z.B. Fake-Clients) empfangene Nachrichten zu speichern, nicht empfehlen. Vermutlich kann bei vielen Diensten beispielsweise durch Reverse Engineering ein Fake-Client implementiert werden, welcher eine permanente Speicherung der Nach-richten ermöglicht. Für Snapchat gibt es bereits mehrere Fake-Clients, dies liegt wahrscheinlich an der Tatsache, dass Snapchat der Ephemeral Messaging Dienst mit der größten Nutzerzahl ist.

Wenn in Zukunft auf allen Plattformen ein Weg gefunden wird, die Nutzung von Fake-Clients zu unterbinden, würde dies die Sicherheit der Anwendungen enorm steigern. Zusätzlich müssten möglichst viele Möglichkeiten der Bildschirmaufnahme innerhalb der Ephemeral Messaging Anwendung blockiert werden, um dem Empfänger die Speicherung der Nachricht möglichst schwer zu machen. Darüber hinaus wären Bildschirme, welche aufgrund ihrer Konstruktion oder Oberfläche das Fotografieren der Anzeige erschweren, in diesem Kontext sehr wünschenswert.

Ephemeral Communication ist meiner Meinung nach ein sehr interessanter Ansatz um persönliche Nachrichten und Daten im Internet zu schützen. Diese Lösung schränkt die Nutzung nur wenig ein und liefert kann ein hohes Maß an Sicherheit liefern. Aufgrund von Fake-Clients und der Möglichkeit von Bildschirmaufnahmen aufgrund des “analog hole” bleibt allerdings ein wesentliches Risiko der Datenspeicherung bestehen. Das Konzept wirkt im ersten Moment sehr ansprechend, ist aber aus diesem Grund praktisch nicht umsetzbar.

Besonders schützenswerte und private Nachrichten und Daten sollten trotz selbstlöschender Nachrichten und Ende-zu-Ende-Verschlüsselung nicht über das freie Internet geschickt werden. Es gibt keinen Weg die Sicherheit einer Nachricht in jedem Fall zu garantieren. Aus diesem Grund sollten geheime Informationen am besten von Angesicht zu Angesicht an einem geschützten Ort ausgetauscht werden.

8. LITERATUR

[1] *The Guardian: Edward Snowden and the NSA files* http://www.theguardian.com/world/2013/jun/23/edward-snowden-nsa-files-timeline, letzter Aufruf: 02.04.2015

[2] *FutureBiz: Facebook Statistiken* http://www.futurebiz.de/artikel/facebook-statistiken-475-mrd-inhalte-werden-taeglich-auf-facebook-geteilt, letzter Aufruf: 05.05.2015

[3] *eMail-Verschlüsselung mit PGP* http://www.elektronikinfo.de/pc/pgp.htm, letzter Aufruf: 05.05.2015

[4] *PGP - Pretty Good Privacy* http://tools.ietf.org/html/rfc2440, letzter Aufruf: 02.04.2015

[5] T. Engel, R. Weis, C. Kordecki *Sichere Nachrichtenübermittlung mit Off-the-Record-Messaging*, Doktorarbeit, 2007, Technische Fachhochschule Berlin

[6] *Apptimate* https://apptimate.io/2014/01/ephemeral-messaging-for-privacy-protection/, letzter Aufruf: 02.04.2015

[7] *Snapchat* https://www.snapchat.com/, letzter Aufruf: 02.04.2015

[8] *Snapchat bei Google Play* https://play.google.com/store/apps/details?id=com.snapchat.android&hl=de, letzter Aufruf: 05.05.2015

[9] *Snapchat Review* http://www.1mtb.com/snapchat-review-and-features-of-the-ephemeral-messaging-app/, letzter Aufruf: 02.04.2015

[10] *Snapchat Nutzerzahlen* http://www.golem.de/news/sexting-dienst-snapchat-hat-100-millionen-nutzer-1408-108835.html, letzter Aufruf: 05.05.2015

[11] *Snapchat Anzahl versendeter Snaps pro Tag* http://www.netzpiloten.de/snapchat-geplatzter-traum-von-loschbaren-daten/, letzter Aufruf: 05.05.2015

[12] *Ansa* http://www.ansa.com, letzter Aufruf: 02.04.2015

[13] *Ansa bei Google Play* https://play.google.com/store/apps/details?id=com.ansa.messenger, letzter Aufruf: 02.04.2015

[14] *Ansa bei TechCrunch* http://techcrunch.com/2013/09/09/ansa-is-a-messaging-app-that-let-you-talk-off-the-record/, letzter Aufruf: 02.04.2015

[15] *Wickr* https://www.wickr.com/, letzter Aufruf: 02.04.2015

[16] *Wickr Review* http://www.1mtb.com/wickr-review-and-features-of-the-ephemeral-messaging-app/, letzter Aufruf: 02.04.2015

[17] *Wickr bei Mashable* http://mashable.com/2013/03/04/wickr/, letzter Aufruf: 02.04.2015

[18] *Wickr bei Google Play* https://play.google.com/store/apps/details?id=com.mywickr.wickr2&hl=de, letzter Aufruf: 02.04.2015

[19] *Frankly Messenger* http://franklyinc.com/, letzter Aufruf: 02.04.2015

[20] *Frankly Messenger Review* http://www.1mtb.com/frankly-messenger-review-and-features-of-the-ephemeral-messaging-app/, letzter Aufruf: 02.04.2015

[21] *Frankly Messenger bei Google Play* https://play.google.com/store/apps/details?id=com.chatfrankly.android&hl=de, letzter Aufruf:

02.04.2015

[22] *Privatext* http://www.privatext.co/, letzter Aufruf: 02.04.2015

[23] *Privatext bei Google Play* https://play.google.com/store/apps/details?id=com.privatext.droid&hl=de, letzter Aufruf: 02.04.2015

[24] *Privatext bei Gigaom* https://gigaom.com/2013/06/24/burn-after-reading-privatext-messaging-app-allows-secure-texts-pictures-to-self-destruct/, letzter Aufruf: 02.04.2015

[25] *SaveMySnaps* http://www.savemysnaps.com/, letzter Aufruf: 02.04.2015

[26] *5 Ways To Save Snapchat Snaps Permanently Without The Senders Knowledge* http://www.1mtb.com/5-ways-to-save-snapchat-snaps-permanently-without-the-senders-knowledge/, letzter Aufruf: 02.04.2015

[27] *Top 5 Apps To Save Snapchat Photos/Videos/Stories On iOS And Android* http://www.1mtb.com/top-5-best-apps-to-save-download-snapchat-photos-videos-stories-ios-android/, letzter Aufruf: 02.04.2015

[28] *Dumpster* http://dumpsterapp.mobi/index.html, letzter Aufruf: 02.04.2015

[29] *ScreenShot HD bei Google Play* https://play.google.com/store/apps/details?id=com.acr.screenshothd, letzter Aufruf: 02.04.2015

[30] R. Geambasu, T. Kohno, A. Levy, H. M. Levy *Vanish: Increasing Data Privacy with Self-Destructing Data*, Proc. of the 18th USENIX Security Symposium, 2009, University of Washington

[31] L. Zeng, Z. Shi, S. Xu, D. Feng *SafeVanish: An Improved Data Self-Destruction for Protecting Data Privacy*, Cloud Computing, Second International Conference, 2010, Huazhong University of Science and Technology

[32] A. Shamir *How to Share a Secret*, Communications of the ACM, Volume 22 Issue 11, 1979, Masssachusetts Institute of Technology

doi: 10.2313/NET-2015-09-1_01

HTTP/2

Michael Conrads
Betreuer: Benjamin Hof
Seminar Future Internet SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: conrads@in.tum.de

ABSTRACT

The new version of HTTP/1.1, HTTP/2, has been approved by the IETF's steering group for publication as an RFC. The number of assets per web page has increased over the last ten years. Optimization techniques like sharding, CSS data: inlining or image sprites are used to address speed issues inherent to the HTTP/1.X protocols. HTTP/2 removes the need for these optimizations by introducing binary frames, transmitted over streams on a single TCP/IP connection. It enables the client to indicate priorities for streams to the server. Servers are able to push data to clients anticipating the clients need for resources like JavaScript or CSS files referenced in requested HTML. HTTP/2 transmits headers using the HPACK format, using redundancy in the header key-value pairs to reduce the transmitted header size. The HTTPbis working group reached its goal in creating an performance optimized HTTP protocol, but it is questionable if advanced features of HTTP/2 will be fully supported.

Keywords

SPDY, HTTP/2, HPACK

1. INTRODUCTION

HTTP is one of the most commonly used protocols on the web. It was first defined in 1991 [6] and was published as a Request For Comment (RFC) in 1996 (RFC 1954, Hypertext Transfer Protocol – HTTP/1.0) [10]. It was widely adopted and the next version HTTP/1.1 followed in 1999 [9]. In addition to functionality updates, speed issues were addressed by adding *pipelining* [9, section 8.1.2.2], which is explained in section 3. Furthermore, HTTP/1.1 made persistent TCP connections the default setting for HTTP [9, section 8.1.2]. The increasing number of images, CSS and JavaScript files per page require an increasing number of HTTP request, which made new drawbacks of HTTP/1.1 became apparent. Figure 1 shows the average web page transfer size as well as number of requests, based on the Alexa top 1 million pages between Jan 2011 and Mar 2015 [12]. As can be seen, the number of requests per page increased from 78 to 96 and the total transmission size increased from 724kB to 1977kB in the last 4 years.

HTTP/2 tries to address these problems and adapt HTTP to the requirements of the modern web.

2. A SHORT INTRODUCTION TO HTTP

HTTP/1.X (HTTP/1.0 and HTTP/1.1) are text-based protocols used for transmitting content between a client and a server in request-response form. A well known use case is the transmission of web pages. HTTP is built on top of the TCP/IP stack, guaranteeing reliable transfer between two parties. HTTP/1.X messages consists of a header and a body, separated by two carriage-return linefeeds (`CR LF CR LF`, `CR` and `LF` are control characters denoting a new line).

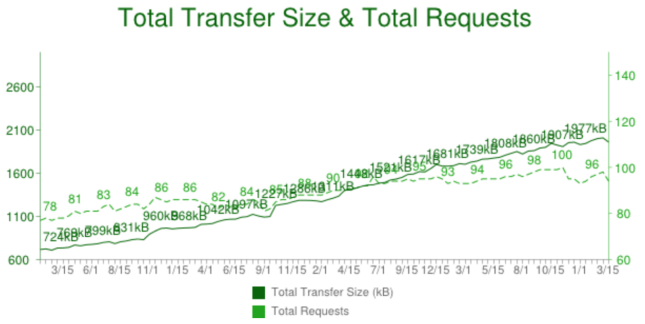

Figure 1: Average transfer size and number of requests required to load a web page

Listing 1 shows an exemplary HTTP/1.1 response from a server to a request from the client. Every header value is separated by one `CR LF`. The last header value '`Server: Apache`' is separated from the content of the response '`<!DOCTYPE html`' by `CR LF CR LF`, visible as the empty line after the header. The header of a HTTP message contains meta-information regarding the request or response, e.g. the method used for the request (GET, POST etc.) or the status code for the response as well as additional information like the type of the content or information about the server. The body of the HTTP response contains the requested content, in this example a page containing HTML.

Listing 1: Example HTTP Response from a Server

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Apache

<!DOCTYPE html ...>
<html>
content of the web page here
</html>
```


doi: 10.2313/NET-2015-09-1_02

3. PROBLEMS WITH HTTP/1.X

When requesting resources over HTTP/1.0, a new TCP connection is established for every resource transmitted. After the transmission of the resource, the TCP connection is closed again. With the number of resources per web page growing, clients need an increasing number of requests to load them [12]. The TCP *slow-start* mechanism was designed to decrease network load and find the optimal transmission rate for an established connection [21]. *Slow-start* gradually increases the number of transmitted bytes, based on the amount of transmission acknowledgments received. As a result, short lived TCP connections never reach their optimal transmission capacity.

HTTP/1.1 addresses these issues by allowing established TCP connections to be reused by default (`keep-alive`) [9, Section 8.1]. Nevertheless, the popular Apache webserver closes a connection if no request is sent over a period of time, which defaults to 5 seconds [22]. Even if the timeout on the server side is increased, clients maintain their own timeout setting e.g. in Firefox version 36 persistent connections timeouts after 115 seconds. To reuse an established TCP connection, the client has to wait for a response to a request before issuing a new one. HTTP/1.1 circumvents this problem by introducing Pipelining.

Pipelining enables the client to issue multiple requests to a server without having to wait for a response to the first request to issue the second one. The order of the requests is significant, as it is required that the server responds to these requests in the order they were issued. As a result, if the response to the first request is delayed (e.g. because of large file size), all other requests are delayed as well. This problem is called *head of line blocking*. As can be seen, by fixing one problem, HTTP/1.1 creates a new one.

4. HTTP/1.1 WORKAROUNDS

Several workarounds have been found to mitigate the issues described in section 3 and increase web page load speed:

CSS data: inline By including base64 encoded images in CSS stylesheets, multiple images can be loaded with one CSS request, reducing the number of requests to the server.

Spriting Multiple images are combined into one larger image, called a sprite. The individual images are displayed by specifying the offset X and Y coordinates in combination with the target image size [15]. This workaround aims at reducing the number of requests.

Sharding HTTP/1.1 states, a maximum of 2 parallel HTTP connections should be open simultaneously between a client and a server. [9, 8.1.4] Modern browsers don't conform to this standard. Firefox version 36.0.1 allows 6 persistent connection. Sharding refers to the technique of distributing assets of a page on multiple subdomains. These are considered seperate servers, thus resetting the *number-of-connections* limit. Figure 2 show a screenshot of a HTTP-request trace when visiting `www.gmx.de`. GMX distributes images via multiple domains to increase the number of connections between client and server.

Concatenation Another way of decreasing the number of HTTP requests is the concatenation of text based assets. Similar to the spriting technique for images, multiple

Domain	Type	Method	Scheme	Status
i0.gmx.net	Image	GET	HTTP	200
i0.gmx.net	Image	GET	HTTP	200
i2.gmx.net	Image	GET	HTTP	200
i0.gmx.net	Image	GET	HTTP	200
i0.gmx.net	Image	GET	HTTP	200
i2.gmx.net	Image	GET	HTTP	200
i1.gmx.net	Image	GET	HTTP	200
i1.gmx.net	Image	GET	HTTP	200
i2.gmx.net	Image	GET	HTTP	200
i0.gmx.net	Image	GET	HTTP	200

Figure 2: Sharding used by www.gmx.de

JavaScript or CSS files are concatenated into larger files, reducing the number of requests sent to the server.

The existence of these workarounds points to a shift in usage of the HTTP/1.1 protocol in regard to its original purpose when it was created.

5. THE HTTP/2 WORKING GROUP

The Working Group (WG) charged with maintaining the HTTP specifications is part of the Internet Engineering Task Force (IETF) [13]. Its charter states the goals they want to achieve with the HTTP/2 standard [14]:

- "[...] us[e] draft-mbelshe-httpbis-spy-00 as starting point."
- "Substantially and measurably improve end-user perceived latency in most cases, over HTTP/1.1 using TCP."
- "Address the 'head of line blocking' problem in HTTP."
- "Not require multiple connections to a server to enable parallelism, thus improving its use of TCP, especially regarding congestion control."
- "Retain the semantics of HTTP/1.1, [...] including [...] HTTP methods, status codes, URIs, and [...] header fields." This goal aims at compability with websites accessing HTTP/1.1 header fields.
- "Clearly define how HTTP/2.0 interacts with HTTP/1.x" As HTTP/2 will be gradually rolled, dynamic upgrade mechanism from HTTP/1.1 connections to HTTP/2 are important to ensure a smooth transition which is not noticed by the user (e.g. no popup asking the user if he wants to use the HTTP/2 protocol).
- "Clearly identify any new extensibility points and policy for their appropriate use."

The fulfillment of these goals will be evaluated in section 9.

5.1 The SPDY Protocol

As stated in the WG charter, `draft-mbelshe-httpbis-spy-00` [4] of the SPDY protocol was used in 2012 as the starting point for HTTP/2 [16]. The SPDY protocol was developed by Google, which made the existence of the project known to the public in 2009. It was meant as a replacement for HTTP and "optimize the way browsers and servers communicate" [3]. SPDY features a binary format, stream multiplexing as well as header compression. These features were

retained in the HTTP/2 protocol and are described in the following sections. The SPDY Protocol is supported on the client side by Google Chrome since version 6 (2010), Firefox since version 13 (2012) and Safari since version 8 (2014) [8]. On the server side, SPDY support was added as a mod to Apache version 2.2 (2012) [20] and nginx since version 1.3 (2013) [1].

Google will retire the SPDY protocol in "early 2016", as "[...] HTTP/2 is well on the road to standardization." [2].

6. THE HTTP/2 PROTOCOL

This section gives a summary of the core features of the HTTP/2 protocol. Contrary to the HTTP/1.1 text based protocol, HTTP/2 is a binary protocol. The main difference between text based and binary protocols is the way in which they represent values. Text based protocols use ASCII key-value pairs like 'Content-Length: 42' (which is at least 16 bytes long) whereas binary protocols transmit values in designated bit positions (e.g. the first 3 bytes in an HTTP/2 message contain the length of the message). This space-efficient value representation reduces the size of transmitted messages.

HTTP/1.1 defines `CR LF` as the separator between header fields [9, section 4.1]. The header parsing is not clearly defined in some points, e.g. HTTP/1.1 defines four different ways to specify message length based on the context of the message or the existence of header fields [9, section 4.4]. The RFC even states an example of wrong implementations producing extra `CR LF` sequences [9, section 4.1]. HTTP/2 transmits data in frames, which are logical grouped bytes. The concept of frames and their different types are explained in section 6.2.

As opposed to HTTP/1.1, HTTP/2 defines a clear separation between header information and content. Header fields are transmitted using `HEADERS` frames, content is transmitted using `DATA` frames. In HTTP/1.1 header and content are transmitted together, separated by `CR LF CR LF`.

Another advantage of binary protocols is the reduction of protocol overhead in regard to number of bytes transmitted, as described at the top of this section. Common criticism of binary protocols is their illegibility when displayed during text based debugging. While this is true, TLS encrypted HTTP/1.1 messages are not human readable as well. In addition, the popular network packet analyzer Wireshark features HTTP/2 support[1].

Frames and Streams

Frames and streams are the core concepts of HTTP/2. Frames are the basic unit of data transmission between clients and servers. Similar to TCP or IP frames, they start with meta information about their purpose and payload and carry their content in the payload field. Frames are described in section 6.2. Frames are transmitted via streams. Streams are theoretical groups of frames which are transmitted over one TCP connection. The TCP connection is not aware of the existence of streams. Every frame carries a `stream identifier`, marking their assigned stream. Streams can be prioritized in regard to each other and allow multiplexing of frames on one TCP connection. Streams are described in section 6.3.

The following sections shows how an HTTP/2 connection is established and data transmitted between clients and servers.

6.1 Initiating a HTTP/2 connection

There are multiple ways to initialize a HTTP/2 connection, depending on the usage of TLS or cleartext transmission.

HTTPS over TLS

To establish a secure connection to a HTTPS URI, the client uses the application layer protocol negotiation (ALPN, [11]), an extension to the TLS protocol. As part of the TLS handshake, the client sends a list of supported application layer protocols (represented by protocol identifiers) to the server in the *ClientHello* message. The server responds with a selected protocol in the *ServerHello* message [11, section 3.1]. The protocol identifier used for HTTP/2 is `h2` [5, section 3.3].

After the completion of the TLS handshake, client and server have to send a connection preface [5, Section 3.4]. The *Connection Preface* is composed of the string '`PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n`' base 64 encoded and a `Settings` frame. `PRI` is a new HTTP method, used exclusively for the HTTP/2 connection establishment [5, section 11.6].

Cleartext HTTP

If the client is not aware of server HTTP/2 capabilities, it uses the HTTP/1.1 upgrade mechanism, issuing a HTTP/1.1 request to the server. This initial request contains a HTTP `Upgrade` header field with the value `h2c` (HTTP/2 cleartext) as well as a base64 encoded HTTP/2 `SETTINGS` frame as the value of the new header field `HTTP2-Settings` [5, section 3.2].

The server acknowledges the protocol change by sending a `101 switching protocols` HTTP/1.1 message. If the server does not support HTTP/2, it ignores the `Upgrade` and `HTTP2-Settings` header fields and respond with a HTTP/1.1 `200 OK` response.

If the client is aware of HTTP/2 capabilities of the server (e.g. from the `Alt-Svc`, see section 8 or previous connections) and wants to establish a cleartext TCP connection, it can send the *Connection Preface* and immediately start sending HTTP/2 frames.

Note the `\r\n\r\n` (`CR LF CR LF`) in the *Connection Preface*. When receiving the *Connection Preface*, the server starts scanning for a HTTP/1.1 header, which is ended by `CR LF CR LF`. In cases where the client assumes the server supports HTTP/2 from prior knowledge, but the server does not (e.g. server has been replaced), the server does not recognize the *Connection Preface* and does not attempt to scan the following HTTP/2 frames.

[1] `https://wiki.wireshark.org/HTTP2`

6.2 Frames

Frames are the smallest unit of communication between client and server. Figure 3 shows the format of a HTTP/2 frame. The number in braces specifies the number of bits per field. Every frame starts with a `Length` field, containing the number of bytes in the `Payload` field. `Type` and `Flags` are discussed in the following section. `R` is a reserved bit. The `Stream Identifier` (stream ID) indicates the stream this frame is assigned to, see section 6.3. The `Frame Payload` field contains the actual payload of the frame.

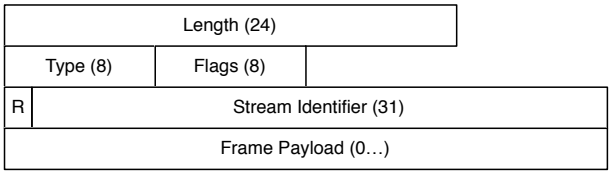

Figure 3: HTTP/2 Frame format

Frame Types and Flags

There exist several frame types, the most important ones are listed here, other frame types are mentioned where appropriate. The data transmitted by these frames is carried in the payload field. Unknown frame types must be discarded [5, section 4.1].

DATA A frame with type `DATA` transmits request or response payloads like HTML, CSS, images or other resources.

HEADERS The `HEADERS` frame is used to open a new stream and transmit *header fragments*, compressed via the compression format HPACK. If the HTTP header in its compressed form is too large the fit in one `HEADERS` frame, it is split up into *header fragments* where the first one is transmitted in the `HEADERS` frame and the following fragments are transmitted in `CONTINUATION` frames. If the `END_HEADERS` flag is set in one frame, it signals to the receiver the completed transmission of all *header fragments* and the receiver is able to assemble them to their decompressed HTTP header form. The HPACK compression and *header fragments* are explained in section 7.

An optional use of the `HEADERS` frame is to specify an initial stream dependency, explained in section 6.3.

SETTINGS The `SETTINGS` frame is used to control the entire connection, not just a single stream. The `Stream Identifier` must therefore be set to `0x00`. The `SETTINGS` frame must be sent from both endpoints to negotiate the connection at startup.
The `SETTINGS` frame is used to control the size of the dynamic header compression table (see section 7). It allows the clients to enable/disable `PUSH_PROMISE` frames and specify a flow control window size (`SETTINGS_INITIAL_WINDOW_SIZE`, see section 6.4) as well as a maximum frame size (`SETTINGS_MAX_FRAME_SIZE`).

PRIORITY The `PRIORITY` frame is used to indicate a dependency between two streams. It contains the identifier of the stream the current stream depends on, as well as a weight which is assigned to the siblings of the current stream. Stream dependencies and weights are explained in section 6.3.

PUSH_PROMISE The `PUSH_PROMISE` frame payload contains a promised stream ID and *header fragments*. The `PUSH_PROMISE` must only be sent from the server and indicates to the client the intent to push resources which are not (yet) requested. An example would be the request to a web page with an embedded image A. The server knows, that the client will request the image as soon as it receives the HTML with the embedded link. The server can send a `PUSH_PROMISE` frame *before* it sends the HTML in a `DATA` frame. The `PUSH_PROMISE` has to be sent *beforehand* to avoid race conditions (e.g. the client requests resources while the server is already sending them). The client receives the `PUSH_PROMISE` frame which contains a request header for the not yet requested image A. The client reserves a stream with the transmitted `Stream Identifier` and knows (from the header) that the server will push the image A on this stream. When the client parses the HTML, it can match the link to image A to the request header for image A sent in the `PUSH_PROMISE` from the server, thereby saving the request to the server for the image. In this regard, `PUSH_PROMISE` works similar to the HTTP/1.1 *Data: inline* technique with the possibility of client opt-out: If the client does not want to receive server pushes, it can disable them with the `SETTINGS_ENABLE_PUSH` flag in the `SETTINGS` frame. If the client wants to receive server pushes in general, but wants to cancel a specific `PUSH_PROMISE`, it can send a `RST_STREAM` frame.

RST_STREAM The `RST_STREAM` frame (Reset Stream) is used to cancel a stream. In HTTP/1.1, peers had no way of aborting an ongoing transmission of data without having to close the TCP connection. The `RST_STREAM` frame can be sent at any time to tell the peer that all transmissions on a specific stream should be canceled.

WINDOW_UPDATE The `WINDOW_UPDATE` frame is used to *increase* the flow control window. Flow control can apply to individual streams as well as the whole connection. If a peer wants to reduce the flow control window, it has to send a new `SETTINGS` frame, containing the new value for the setting `SETTINGS_INITIAL_WINDOW_SIZE`. Flow control is described in section 6.4.

6.3 Streams

A stream is a concept for bi-directional transmission of frames between client and host, not a physical existing connection. One TCP connection can contain multiple streams. Every frame is assigned to a stream. As streams are logical concepts, there is no overhead in establishing new streams, as opposed to establishing a new TCP connection. Streams are referenced by IDs. To circumvent a possible ID collision, streams opened by the server are assigned even numbered IDs, streams opened by the client odd numbered IDs [5, section 5.1.1].

Streams are opened using a `HEADERS` frame, identifying the resource that will be transmitted over the stream. If the `PRIORITY` flag is set, the `HEADERS` frame includes an initial stream priority and dependency. As opposed to HTTP/1.1 requests/responses, HTTP/2 stream communication is stateful. Both `HEADERS` and `DATA` frames are needed to transmit resources and their order is significant for their reassembly on the receiver side.

doi: 10.2313/NET-2015-09-1_02

Priority and Dependency

Streams can be dynamically (re)prioritized using a `Priority` frame, which assigns priorities as weights. Weights are values between 1 and up to 255. The priority of a stream tells the server which requests it should answer first. If a server receives requests from a high-priority stream and a low priority stream, it should assign more resources (e.g. bandwidth) to answering the high-priority stream than the low-priority stream.

Streams can be dependent on each other. This creates a dependency tree on the server. The stream with ID 0 is not used and is inserted on the server to form the root of the tree. The HTTP/2 specification does not enforce strict compliance to the priority and dependency model [5, section 5.3.1 and 5.3.2].

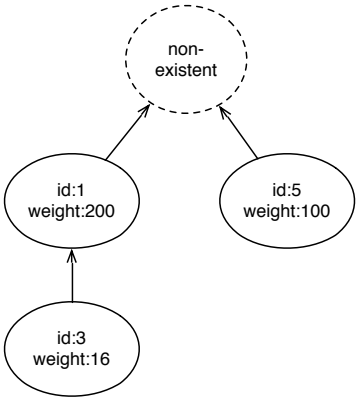

Figure 4: Stream Dependency example

Figure 4 shows an example for stream dependency. Streams 1 and 5 do not have a parent stream (the `Stream Dependency` field contained `0x00`) but different weights. This prompts the server to assign more resources to stream 1 than to stream 5. Stream 3 is depending on stream 1, which means requests on stream 1 should be handled before requests on stream 3. A weight of 16 is assigned as the default value, if no priority is specified.

Consider this exemplary transmission of a web page using the dependency in figure 4:
A client requests a web page, which contains an image and a JavaScript file. As the HTML contains the links to all needed resources, it should be transmitted first and will be requested over stream 1. The JavaScript file is transmitted over stream 3, as the JavaScript has to wait for the browser to build the DOM-tree before it is able to apply DOM manipulations. The image can be requested over stream 5, as soon as the HTML `DATA` frame containing the image link is parsed.

6.4 Data Transmission and Flow Control

Persistent connections introduced in HTTP/1.1 in combination with *pipelining* allow multiple HTTP requests in parallel. The order in which these requests are answered is used to match the responses to the requests and is therefore significant. Consider two exemplary *pipelined* HTTP/1.1 requests to a server, requesting a database entry and a static html page. *Pipelining* requires these requests to be answered in the same order, e.g. the response from the database followed by the static html page. As the response from the database takes longer than the static html page, the first response blocks the seconds one. Streams allow HTTP/2 frames to be multiplexed over one TCP connection. The order of frames is significant per stream, not per TCP connection. This ensures requests to slower resources (e.g. images or database entries) do not block faster responses, regardless of the order in which they are sent. Figure 5 visualizes a simplified HTTP/1.1 transmission without the usage of persistent connections to demonstrate the TCP and HTTP handshake overhead. The *index.html* response carries the payload as well as the HTTP headers response.

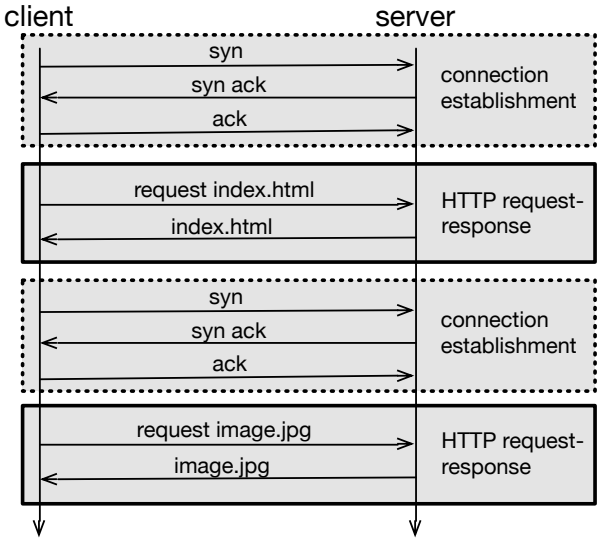

Figure 5: HTTP/1.1 transmission without keep-alive, simplified

Figure 6 shows the request and transmission of HTML as well as several images: `HEADERS` frames from the client open new streams and request resources. No TCP overhead is generated if a new stream is opened. The server responds by sending a `HEADERS` frame with the specified stream ID to transmit the HTTP response headers, followed by `DATA` frames carrying the resources. If the server transmitted a resource, the last frame on the stream carries the `END_STREAM` flag (`ES` in figure 6), signaling the end of transmission on the stream.

HTTP/2 provides a flow control mechanism for each TCP connection. Flow control applies only to `DATA` frames, all other frames are not affected. TCP flow control is meant to protect the client from receiving data faster than it can process. As multiple streams are transmitted over the same TCP connection, TCP flow control cannot apply to single streams. HTTP/2 flow control can restrict or increase the throughput on individual streams by transmitting `SETTINGS` or `WINDOW_UPDATE` frames. Every peer maintains a flow control window, which is initially set to 65,535 Bytes [5, section

doi: 10.2313/NET-2015-09-1_02

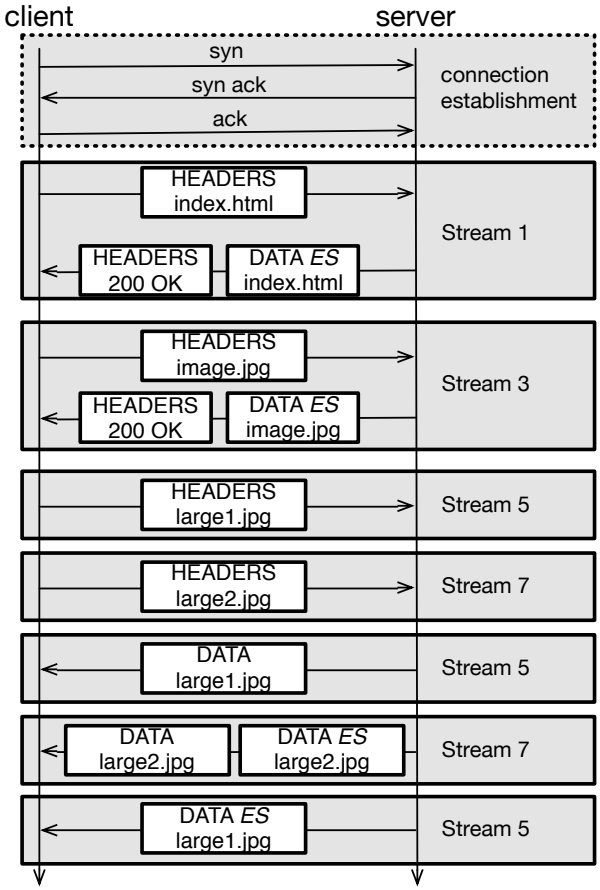

Figure 6: HTTP/2 transmission, simplified. ES denotes the END_STREAM flag

5.2.1]. While stream priorities are relative values regarding the processing of requests on the server side, flow control can set hard limits on individual streams.

7. HPACK HEADER COMPRESSION

One of the HTTP/2 goals is to "Retain the semantics of HTTP/1.1 [...] including [...] header fields." [14]. HTTP/1.1 header fields are very repetitive, with minimal changes between requests for resources. The header compression format `HPACK` [19] uses this repetition to substantially reduce transmitted header size.

`HPACK` allows header fields to be split into multiple *header fragments* if they are too large to fit into one frame. `HEADERS` and `CONTINUATION` frames are used to transmit *header fragments*. The client collects the fragments and assembles the complete header [19, section 1.1]. The format of the *header fragments* in the `HEADERS` frame is identical to the one in the `CONTINUATION` frame. For convenience, only the `HEADERS` frame is mentioned.

The `HPACK` compression uses two tables for decoding, a static table and a dynamic table.

Static Table

The static table contains fixed values for the most common header fields. The index 2 for example contains the HTTP method `GET`. Index 5 contains path `/index.html`. A transmission of the index 2 and the index 5 results in `GET /index.html`. The content of the static table can be found in [19, Appendix A.].

Dynamic Table

The dynamic table is empty at the start of the connection. Both client and server maintain the same dynamic table. `HEADERS` frames transmit headers in multiple key-value tuples inside *header fragments*, in combination with bit flags if the entry in the key or value field in each tuple is an index or a literal.

If a peer wants to send a header, it iterates over each field in the original HTTP header (consisting of key and value), looking for a representation in the static table. If an entry is found, the appropriate field in the `HEADERS` frame is set. Combinations of an indexed key and a literal value or vice versa are indicated by the bit flags for each tuple. If a representation cannot be found in the static table, the dynamic table is searched. If no entry can be found as well, the key or value is added to the dynamic table and added as a literal to the `HEADERS` frame. When decoding the `HEADERS` payload, the receiver adds the transmitted literals to its dynamic table as well, recreating the same dynamic table as the sender. If the same key or value is sent again in another request or response, its index from the dynamic table is used.

Hence, `HPACK` compression is not stateless, requiring every peer to maintain an identical encoding context in form of the dynamic table. To further reduce size, the `header fragments` can be *Huffman* encoded, which is indicated by a bit in the transmitting frame.

The `HPACK` specification describes only the decoding of headers. The encoder is required to transmit the header fragments in a way, such that the client is able to decode them [19, Section 2.]. Individual implementations (on client and server side) must come up on their own with efficient algorithms for table traversal and storage. The specification therefore does not provide an implementation proposal.

8. EXTENSIONS

HTTP/2 is a very strict protocol. The specification states, that all unknown frames have to be ignored by the receiver [5, Section 5.5]. Extension points exist to increase the flexibility of the protocol. HTTP/2 extensions are not part of the main specification, which states only how extensions must be registered.

Alternative Services

The *Alternative Services Extension* (Alt-Svc) [17] is an additional header field that can be transmitted using the `Alt-Svc` frame in a HTTP/2 connection or as a regular header field in HTTP/1.1. Servers can use the *Alt-Svc* frame to signal the client a resource is available in another location, too. It can be used to switch protocols, e.g. the server tells the client during a HTTP/1.1 connection the resource is also available via HTTP/2 on another port: `Alt-Svc:`

`h2c=":8080"`. It can reroute the client to another endpoint as well: client visits `www.domain.com/endpoint` and receives `Alt-Svc: http="www2.domain.com/endpoint"`. The client is now able to transition to the `www2` server without having to notify the user. This is not to be confused with a HTTP redirect response (a 3XX status code). `Alt-Svc` allows the client to decide, if he wants to use the alternative service [17, 2.4]. In addition, `Alt-Svc` can offer multiple alternative locations for a resource; the client is free to choose any. The HTTP redirect status does not allow such fine control as it forces a hard redirect on the client. The client can response with a header field `Alt-Used` to notify the server of a successful transition. Using `Alt-Svc`, the server can transition a client slowly to another location until all requests are sent to the new location (in case a full transition is desired).

Opportunistic TLS

Opportunistic TLS [18] is an experimental extension based on `Alt-Svc`. It aims at mitigating pervasive monitoring attacks (RFC 7258). Servers advertise using `Alt-Svc` if their content is available over TLS as well. Clients are now able to choose between staying on the current cleartext connection, risking e.g. http traffic monitoring, or switch to a new TLS connection.

9. FULFILLMENT OF WG GOALS

The WG aimed at improving end-user perceived latency. Akamai, a large cloud service provider, released a HTTP/2 test page (https://http2.akamai.com/demo), demonstrating the speed improvements gained by using HTTP/2 over HTTP/1.1.

Another goal was to address the TCP *slow-start* performance problem. HTTP/1.1 tried to improve performance by introducing persistent connections, with led to *head of line blocking* problems. HTTP/2 mitigates the slow-start by reusing the same TCP connection. As opposed to HTTP/1.1, HTTP/2 stream multiplexing and stream priorities circumvent the *head of line blocking* issues.

HTTP/2 does not change HTTP/1.1 semantics. As soon as the client has reassembled all *header fragments*, the HTTP/2 header is identical to the HTTP/1.1 header. Therefore, migration to HTTP/2 does not break working HTTP/1.1 websites which access header fields (e.g. for cookies).

As the upgrade to HTTP/2 is not mandatory, the HTTP/2 specification provides methods for discovering server HTTP/2 capabilities during an existing HTTP/1.1 cleartext connection or a TLS handshake using `ALPN` without breaking existing HTTP/1.1 implementations.

The strength of HTTP/2 is the reduction of number of round trips, as less TCP connections need to be established and the transmitted header size is reduced. If the RTT is already low because of good network conditions, the impact of HTTP/2 for the end user will not be felt as strongly.

10. CONCLUSION

The success of HTTP/2 can be measured on its rate of adoption. All major browsers have added HTTP/2 and/or SPDY support. Due to the HTTP/1.1 header compatibility and the HTTP/2 support in Apache and nginx, gradual roll out of the protocol should be straightforward for most servers, eliminating the need for the HTTP/1.1 workarounds mentioned section 4. According to Daniel Stenberg, a member of the HTTP/2 WG, Google reported 5% of their global traffic uses HTTP/2 by the end of January 2015 [7]. While this early adoption rate sounds promising, it is difficult to validate this number and therefore should be treated with caution. In the opinion of the author, HTTP/2 coverage will never reach 100%, as many small or "abandoned" private webservers will never be updated (following the principle: 'if it ain't broken, don't fix it').

Several speed comparisons between HTTP/1.1 and HTTP/2 exist, claiming varying speed improvements using HTTP/2 over HTTP/1.1[2]. These comparisons are often lacking in information regarding the test setup and are dubious in their expressiveness. HTTP/2 core features like stream multiplexing or header compression aim at increasing responsiveness for web pages using a large number of assets. To objectively measure HTTP/2 performance, a web page operator has to measure HTTP/1.1 web page load times with current HTTP/1.1 workarounds in place and compare the results to the same page using HTTP/2 without using the mentioned workarounds. This test requires a full HTTP/2 implementation (using features like promise and stream priorities) on server and client side as well as a web page with actual content. While HTTP/2 test pages show promising results, the use case of requesting 200 images at once seems questionable. Therefore it is very difficult to provide actual HTTP/2 test results using real web page content and would go beyond the scope of this paper.

Nevertheless, the author is certain that the impact of HTTP/2 for the end user will be felt on mobile devices due to the efficient usage of TCP. HTTP/2 offers advantages for web pages with large numbers of assets as well, removing the need for many HTTP/1.1 optimizations. The webserver support for advanced features of HTTP/2 like stream dependency, priorities or push promises seems questionable though, as the HTTP/2 specification does not strictly require the server to adhere to them or use them at all.

[2] `http2.golang.org/gophertiles` or `blog.httpwatch.com/2015/01/16/a-simple-performance-comparison-of-https-spdy-and-http2`

doi: 10.2313/NET-2015-09-1_02

11. REFERENCES

[1] Bartenev, V. Announcing SPDY draft 2 implementation in nginx. http://mailman.nginx.org/pipermail/nginx-devel/2012-June/002343.html, June 2012.

[2] Beky, B. and Bentzel, C. Hello HTTP/2, Goodbye SPDY. http://blog.chromium.org/2015/02/hello-http2-goodbye-spdy-http-is_9.html, Feb. 2015.

[3] Belshe, M. and Peon, R. A 2x Faster Web. http://googleresearch.blogspot.de/2009/11/2x-faster-web.html, Nov. 2009.

[4] Belshe, M. and Peon, R. SPDY Protocol. http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00, Feb. 2012. Internet-Draft.

[5] Belshe, M. and Peon, R. and Thomson, M. Hypertext Transfer Protocol version 2, Draft 17. https://tools.ietf.org/html/draft-ietf-httpbis-http2-17, Feb. 2015. Internet-Draft.

[6] Tim Berners-Lee. The Original HTTP as defined in 1991. http://www.w3.org/Protocols/HTTP/AsImplemented.html.

[7] Daniel Stenberg. HTTP/2 is at 5%. http://daniel.haxx.se/blog/2015/02/10/http2-is-at-5/, Feb. 2015.

[8] Deveria, A. HTTP/2 protocol / SPDY. http://caniuse.com/#feat=spdy, Feb. 2015. dynamically generated.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol-HTTP/1.1, June 1999. RFC 2616.

[10] R Fielding, U C Irvine, and H Frystyk. HTTP1.0, May 1996. RFC 1945.

[11] Friedl, S. and Popov, A. and Langley, A. and Stephan, E. Transport Layer Security (TLS), Application-Layer Protocol Negotiation Extension, July 2014. RFC 7301.

[12] httparchive.org. Trends. http://httparchive.org/trends.php?s=All&minlabel=Jan+20+2011&maxlabel=Mar+15+2015, Mar. 2015. dynamically generated.

[13] IETF HTTP Working Group. About Us. http://httpwg.github.io/about.

[14] IETF HTTP Working Group. Charter for Working Group. http://datatracker.ietf.org/wg/httpbis/charter/, Oct. 2012.

[15] Mozilla Developer Network. CSS Image Sprites. https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/CSS_Image_Sprites, 2015.

[16] Nottingham, M. Rechartering HTTPbis. https://lists.w3.org/Archives/Public/ietf-http-wg/2012JanMar/0098.html, Feb. 2012. HTTPbis Mailing List.

[17] Nottingham, M. and McManus, P. and Reschke, J. HTTP Alternative Services, Draft 6. http://tools.ietf.org/html/draft-ietf-httpbis-alt-svc-06, Feb. 2015. Internet-Draft.

[18] Nottingham, M. and Thomson, M. Opportunistic Security for HTTP. https://tools.ietf.org/html/draft-ietf-httpbis-http2-encryption-01, Dec. 2014.

[19] R Peon and H. Ruellan. HPACK - Header Compression for HTTP/2. http://tools.ietf.org/html/draft-ietf-httpbis-header-compression-12, Feb. 2015. Internet-Draft.

[20] Steele, M. mod_spdy is now an Apache project. http://googledevelopers.blogspot.de/2014/06/modspdy-is-now-apache-project.html, June 2014.

[21] Stevens, W. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, Jan. 1997. RFC 2001.

[22] The Apache Software Foundation. Apache-Kernfunktionen. https://httpd.apache.org/docs/2.4/mod/core.html#keepalivetimeout, 2015.

All online sources were last accessed on 28. April 2015.

Dark Internet Mail Environment

Linus Lotz
Tutor: Oliver Gasser
Seminar Future Internet SS2015
Chair for Network Architectures and Services
Department of Computer Science, Technische Universität München
E-mail: lotz@in.tum.de

ABSTRACT
The current mail system is still very important for communcation today, but does not integrate End-to-End encryption and a lot of metadata is leaked during transit. This paper will have a look at Dark Internet Mail Environment (DIME) developed by Ladar Levison.

DIME tries to address these issues by using a new encrypted mail format called D/MIME to provide End-to-End encryption and removing the identifiying information from the protocol conversations in order to reduce the visible metadata. We close with a comparision of DIME with other systems.

Keywords
Dark Mail, DIME, DNSSEC, E-mail, Privacy, Metadata

1. INTRODUCTION
The Dark Mail initiative was founded by Ladar Levison. He was the founder of Lavabit, an e-mail service, that stored e-mails encrypted on the server. After Edward Snowden revealed himself, Lavabit was forced to reveal it's TLS private keys.

In order to protect his users Ladar Levison chose to shut down his service. Shortly after he announced that he had teamed up with Silent Circle, a company founded by Phil Zimmerman (author of PGP and ZRTP). He started a very successful kickstarter campaign for Dark Mail and in December 2014 the first public documentation was released. This documentation describes the Dark Internet Mail Environment (DIME), which we present in this paper.

We start with an detailed overview of DIME in Section 2. In Section 3 we will have a look at the attacker model and the security promised by DIME. Related work will be presented in Section 4 and we will conclude in Section 5.

2. ARCHITECTURE OF DARK MAIL
The Dark Internet Mail Environment[13] consists of several parts. It uses a new Public Key Infrastructure that will be described in section 2.1. The new DIME message format called D/MIME is described in Section 2.2. For transporting and retrieving messages DMTP and DMAP are used, which are described in Sections 2.3 and 2.4.

2.1 Public Key Infrastructure
The DIME specification defines a new certificate called signet, which is used for users and domains. These signets are verified by a Primary Organization Key (POK) or Secondary Organization Key. The last two keys are stored in a DNS record.

2.1.1 Primary Organization Key
The Primary Organization Key (POK) can sign the organizational signets, outgoing mails and user signets for this organization.

2.1.2 Secondary Organization Key
The Secondary Organization Key (SOK) is used to sign the outgoing mails and user signets. When using a Secondary Organization Key, it is possible to keep the POK offline, so it cannot be compromised so easily.

The offline POK would then be used to sign new organizational signets. The SOK would be on the live system to sign outgoing mails. If the SOK is compromised, it cannot be used to create another organizational signet.

2.1.3 Signets
The DIME specification also specifies a simple certificate format for user and domain keys. They are designed so they fit their needs and are very simple. There are two kinds of signets: user signets and organization signets. The user signet is signed with the key from the current organization signet. Every user signet needs to be signed by the POK or an authorized SOK and an organization signet needs to be signed by the POK.

2.1.4 Organization Signets
The organization signets are required to contain the POK and a secp256k1 key which is used to encrypt informations that should be visible to the mailserver. It may also include the hosts for web and mail access and their respective TLS public key fingerprints.

It is also possible to add an onion access host, for access over TOR. The signet can also contain contact addresses for 'abuse', 'admin' or 'support'.

2.1.5 User Signets
Every time a user generates a new key, he signs it with his previous key. This creates a chain of keys, where each key

doi: 10.2313/NET-2015-09-1_03

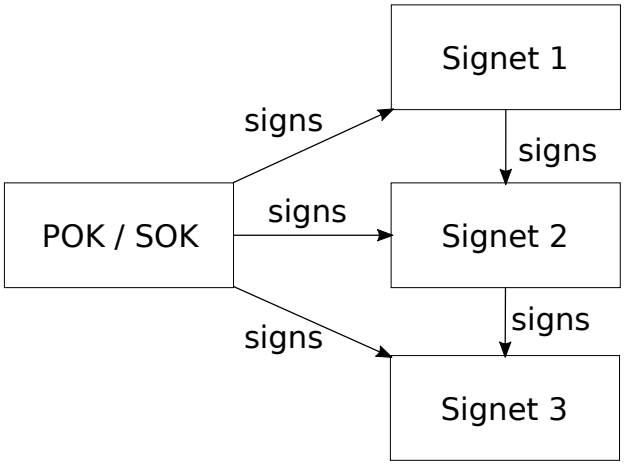

Figure 1: The signing of new user signets

is signed by it's predecessor. The key is then sent to the mailserver, which then signs it with the POK or SOK. The resulting signatures can be seen in Figure 1.

After verifying a key in this chain manually it is possible to verify future keys automatically. All public keys are stored on the mailserver, which can be asked for the keys. This makes it easy to replace the current key regularily. This is a very simple form of Forward Secrecy.

The specification does not describe what happens, if the user loses his key at one point. We assume, that the user has to create a new chain of keys, which can be verified by the server. How other users would switch to this new chain is unclear.

2.1.6 DNS record

DIME uses DNS to get a special DIME record or a `_dime` TXT record [13, chap. 4]. This record contains a Primary Organization Key and optionally a Secondary Organization Key. It is also used to specify the delivery host, which is responsible for accepting messages and providing the signets for this domain.

The record also allows to specify a "policy" setting. This policy setting determines if the domain either accepts Dark Mail or legacy mail or both.

Another special host that can be specified in the record is a so called "syndicate". This host can be used to retrieve signet information for validation.

A "sub" field is used to set a policy for subdomains, which can either allow different DIME records or disallow them. This setting also determines whether subdomains should use the same information as a parent or should be considered as not DIME capable, if no record exists.

An expiery field is used to tell how long a domain is considered DIME enabled even after the record was removed. The TTL is used to set when the record should be refreshed.

An example record could look like this[13, p. 32]:

```
ver=1 pok=MmprdmRjaWtjOTYyZDllMGhrOGNhMTRsZmoyam
t2ZGM pol=mixed
syn=mirror.example.tld
tls=QUYxRjA0MkZDMjQ0OUEzOUJENEE5QkU2MTdENDM3OUV
EQTI1QjQ1REYwODEwODE2ODlGMUE2Q0U1MjQ3M0Y2Mw
dx=dmtp.domain.tld ttl=1776 exp=30 sub=strict
```

Using DNSSEC is encouraged to secure these records to prevent an attacker from performing a downgrade attack. An attacker, who is able to change a DNS response for a DIME record to an NXDOMAIN response, could force a fallback to normal SMTP when DNSSEC is not used. [13, p. 27]

2.1.7 TLS Public Key

The TLS public key fingerprint can be added to the DIME record, which is used in conjunction with DNSSEC to authenticate the certificate. If the record is DNSSEC secured the TLS certificate can be self-signed or signed by an unknown or untrusted CA[13, p. 29].

As a fallback, a valid certificate that matches the hostname and is from a trusted CA is accepted too. This makes it possible to deploy DIME, even if the domain is not secured by DNSSEC.

2.1.8 User Key Management

Dark Mail differentiates between different account modes, which differ in how much the user keys are exposed to the service provider.

In the lowest level "Trustful", the keys are stored on the server and might be encrypted with the user password. The advantage of this procedure is, that any legacy client can be used to access the mails and send mails and the service provider handles all of the Dark Mail encryption. In this case the e-mail content is visible to the service provider.

In the stronger "Cautious" level, the service provider has copies of the encrypted user keys. In this mode a legacy client can not be used, but the message content is not known to the service provider unless he can decrypt the key or sends wrong signets as a Man-in-the-Middle attack.

The strongest level is called "Paranoid" in this case, the provider never has access to the private keys, not even in their encrypted form. If the user regularily rotates his keys and deletes the old ones, he can attain a very simple form of perfect forward secrecy, since he was the only one who had access to the private keys.

2.2 Message Data Format (D/MIME)

D/MIME[13, chap. 6] is a new binary format for storing mails. It is the DIME alternative to MIME[6], the mail format used by e-mails today. D/MIME divides the message in chunks (Figure 3), which get encrypted seperately (Figure 2).

Each encrypted payload (Figure 4) is encrypted with a random 256 bit key and AES-CBC. Messages also contain an ephemeral ECDH key in an unencrypted chunk ("Ephemeral" in Figure 2).

doi: 10.2313/NET-2015-09-1_03

Envelope	Tracing	Unencrypted
	Ephemeral Origin	Unencrypted AOR
	Destination	ADR
Metadata	Common Fields	AR
	Other Fields	AR
Content	Text	AR
	Attachment	AR
Signature	User	AOR
	Organization	AODR

Figure 2: The example of an D/MIME message with access: A=Author, R=Recipient, D=Destination organization, O=Origin organization

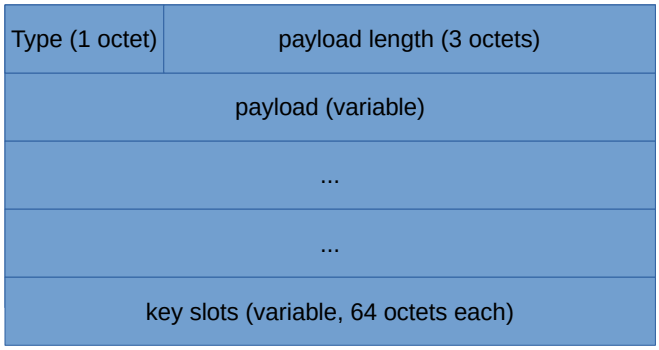

Figure 3: Layout of a chunk in an D/MIME message

This key is used together with the key from an actor (recipient user signet or recipient domain organization signet) to generate a Key Encryption Key (KEK) for this actor:
Sending:

$$\text{KEK} = ECDH(Ephemeral_{private}, Actor_{public})$$

Recieving:

$$\text{KEK} = ECDH(Actor_{private}, Ephemeral_{public})$$

The Initialisation Vector (IV) used for the AES CBC is XORed with 16 bytes of random data, which are encrypted with the KEK along with the result of the XOR and the 256bit key. This makes the first 32 byte different for each keyslot, before encrypting them. The idea is "to prevent a variety of known, and future differential cryptanalysis attacks"[13, p.61]. The result (Figure 5) is then stored in the keyslot of that actor. Using this mechanism the sender can limit, which parts of the message are visible to each actor.

At the end of a D/MIME message, it is signed by both the user and the organization. This is done by signing each hash of the chunks and then the complete content. An example message can be seen in Figure 2.

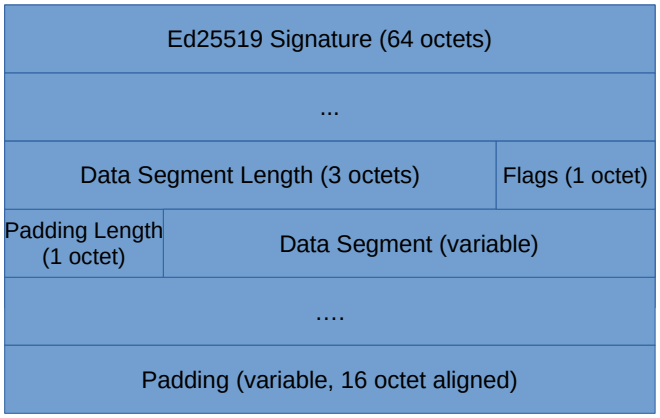

Figure 4: Layout of an encrypted payload

A D/MIME Message only has one recipient, which means that for each recipient the sender has to create a seperate D/MIME message. Since the recipient is stored in a chunk, which can only be read by the destination domain, the actual recipient of the message is hidden to third parties. If an attacker breaks the TLS encryption, he would still need to break the encryption of the D/MIME message to recover this information.

Another benefit of the message format with included encryption and signature is, that it does not suffer from the problems described in [9]. In PGP it is prossible to forward signed messages to a third party and pretend the original signer was the origin and the new recipient the original recipient.

Consider the following scenario: Alice sends Bob a mail which is signed by her and encrypted for Bob:

$$\{\{\text{"Company secret"}\}_{sigAlice}\}_B$$

Bob can now decrypt this message and has now the signed message from Alice. He can create a message which has the "From:" header set to Alice's e-mail, reencrypt the message for Charlie and send it to him:

$$\{\{\text{"Company secret"}\}_{sigAlice}\}_C$$

To Charlie it looks like Alice sent him the message, as she signed the message. This is possible, because the headers in MIME are not signed in PGP.

In D/MIME the complete message is signed, including the sender and recipient used, when sending the e-mail. Forwarding such a mail would change these fields and thus require to resign the message. This is a big improvement over PGP.

Since all D/MIME messages are signed by the organization, faking the origin domain is not as easy as in regular MIME.

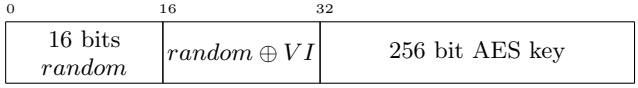

Figure 5: Structure of a unencrypted Keyslot

doi: 10.2313/NET-2015-09-1_03

2.3 DMTP

In the classical e-mail system, mails are sent by the user using an Mail User Agent(MUA) collects the mails from the user and hands them over to a Mail Transfer Agent(MTA)[8]. This MTA then uses SMTP to transfer the message to the destination domain.

DMTP is the replacement for SMTP[8] in the Dark Internet Mail Environment. It is very similar to SMTP but avoids using account names in the conversation. The idea is to remove the information which users are communicating from the protocol conversation. The recipient server can decrypt the chunk which contains the recipient and thus knows to which user the message should be delivered.

The specification describes how an SMTP connection advertises a DMTP capable mail server and how the connection can be upgraded to DMTP with a special STARTTLS command. All DMTP traffic is protected with TLS 1.2 with ephemeral (Elliptic Curve) Diffie-Hellman keys to provide Perfect Forward Secrecy.

DMTP is also used to request the user and organization signets for this domain. For the user signets it is possible to request all signets, that were ever signed by this organization.

Mails are only exchanged between MTAs over DMTP, users who want to send e-mails use DMAP instead. This is different from the current mail architecture, where the user sends the mails using SMTP.

2.4 DMAP

When an e-mail reches the mailbox of the user, he can retrieve it from a Mail Delivery Agent (MDA). In the current mail environment, there are different protocols which are used in this step. One of the more popular protocols is IMAP[4], which will be the model for DMAP [13, chap. 8].

This protocol will be similar to IMAP[4] but without server side search capabilities. The server side search would require the server to have access to the plaintext, which would defeat the purpose of using DMAP for having true End-to-End encryption. If a user creates a new key, he can ask the server over DMAP to sign it for him. When sending a mail over DMAP the server signs the message before handing it over to the MTA. The authentication will be done using a "cryptographic key process"[13, chap. 8]

When sending e-mails, mails are transfered via SMTP to the MTA. To preserve a copy of the sent mail, the client can store a copy on the IMAP server, but this requires retransmitting the same e-mail over IMAP. Using IMAP for sending mails was tried before [5][1], but is not widespread yet. Using DMAP to send mails is an improvement over the current protocols and seperates server-to-server communication and server-to-client communication.

The DMAP protocol is not specified currently and will probably be released with a later specification.

2.5 Protocol Stack Example

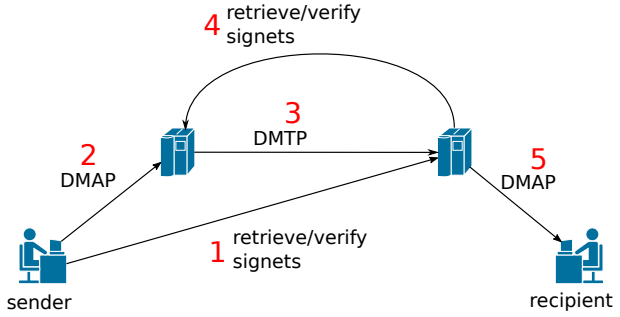

Figure 6: An example message transfer

To illustrate how DIME works, we will show an example where Alice on domain alice.com wants to send a message to bob at bob.com (Figure 6). First Alice, needs to check if Alice has the latest user signet of Bob. She contacts the server for bob.com and asks for the fingerprint of bobs latest key. If she has a different key, she can now ask the server to send her every key between the last key Alice knew and the curent key. She also requests the latest organization signet of bob.com.

After receiving these keys, she needs to verify, that each key signs the next one in the chain and that the signet is signed correctly by the POK or an SOK (where applicable) of bob.com. Now that she has the current key, she can encrypt the chunks and send it via DMAP to alice.com. The MTA on alice.com opens a connection to bob.com with DMTP and transfers the e-mail.

On bob.com, the server needs to verify the signature from alice.com. If he does not already have a matching signet, he must request it from alice.com and verify the signature is from the POK, that is stored in the DNS for alice.com. If the signature is valid and by a valid signet, he can now further process the message.

Now bob.com decrypts the destination chunk. This chunk contains the user that is the recipient on bob.com, in our case Bob. Now that the MTA of bob.com knows the user, it can deliver the mail to his inbox. To receive the message, bob opens a DMAP connection to bob.com and retrieves the message to his local computer. He can now decrypt the chunks and read Alice's e-mail.

The message contains the fingerprint of the signet Alice used for signing. To verify the signature, Bob can check if he already has this key. If not, he contacts the DMTP service for alice.com and asks for the signet with this fingerprint (if he already had a signet from Alice, he would request the signets in between as well).

3. SECURITY ANALYSIS

In the following section we will have a look at the security goals of DIME and the security decisions that were taken.

3.1 Security Goals

Dark Mail Specification states their goals are as follows:"

doi: 10.2313/NET-2015-09-1_03

1. Automate key management, which includes: creation, rotation, discovery and validation
2. Transparently encrypt and sign messages to ensure content confidentiality
3. Ensure the system is resistant to manipulation by Advanced Persistent Threats (APTs)
4. Link security to the complexity of a user's password, and the strength of an endpoint's defenses
5. Minimize the exposure of metadata
6. Give control back to the user"[13, chap. 3]

The current specification of DIME seems to have solved points one and four. Since the attack surface is smaller than with regular e-mail, it is more resilient to APTs. If an attacker had control over the mailserver in the classical mail environment, he would have access to all the metadata and content of the messages. Since the messages are End-to-End encrypted and the metadata is better protected, an attacker would gain mucch less from this kind of access. But this also depends on the account security, which can be controlled by the user (see Section 2.1.8). In practise it will also be improved by a widespread usage of DIME.

The DMAP specification is not released yet and it is the protocol the MUA uses to communicate with the mail server. This is where the password would be used to authenticate the user and to decrypt the encryption keys, if they are stored on the server. Until this part is specified, not much can be said about the attainment of goal three.

Goals two and six also depend on a widespread adoptation of DIME, since otherwise the legacy protocols would be used. These goals are achieved through the D/MIME format, when used with End-to-End encryption.

3.2 DNSSEC

DNSSEC is an extension to the DNS system that adds signing of DNS records to prevent forging of DNS records.

3.2.1 Architecture

DNSSEC adds several new Resource Records (RRs) to each zone. Each zone has two keys, a Key Signing Key (KSK) and a Zone Signing Key (ZSK). The hash of the KSK is stored in a DS record of the parent domain. The Key Signing Key is used to sign the DNSKEY records, where the ZSKs are stored. Every other record is signed by the ZSK. This makes it possible to frequently change the ZSK while keeping the KSK private key offline.

To prevent an attacker from simply sending NXDOMAIN (Domain does not exisist) responses, DNSSEC introduces a record type "NSEC", which points to the next secure record. If the client requests an unknown domain, the server sends this record to prove that it does not exist.

DANE[2] proposes an additional TLSA record, which is used to verify TLS certificates. For it to be trusted by an application, it has to be secured with DNSSEC. It can be used as an additional verification or as an alternate trust-anchor. It's use is not widespread, as DNSSEC secured zones are still very rare.

The TLS certificate validation idea in DIME is very similar to that of DANE. DANE offers more flexibility, since it is also possible to specify the CA, that issued the certificate.

3.2.2 Problems

DNSSEC's trustmodel assumes that you can trust the root zone and the Top Level Domains, which might not be the case. It seems to be very likely, that an intelligence service like the NSA, can get access to the DNSKEYs of the TLDs, that are controlled by the US, like ".com". With this key they are able to forge DNS responses, that are trusted for any domain in ".com".

Even though Eliptic Curve Cryptography in DNSSEC is specified, it is not required to be implemented. RSA is the only algorithm, that is mandatory to implement. RSA has the disadvantage that it's key sizes are relatively large and DNS has a limit for each message, effectively limiting the key size for DNSSEC.

The deployment of DNSSEC requires the use of EDNS, which allows bigger DNS packets than 512 bytes. With larger keysizes the response packets are getting close to 1500 bytes and would usually get fragmented in order to be transmitted.

3.3 Certificate Authentication

In DIME it is possible to verify the certificate with the DIME DNS record or by using CAs.

3.3.1 Certificate Authentication with DNSSEC

Dark Mail uses DNSSEC to verify TLS certificates and the POK via DNS. This involves some disadvantages. Each country code top level domain (ccTLD) is usually controlled by the respective country. This would include the DNSSEC keys. It would be very easy for a country to fake any DNS request for any domain in their ccTLD. It seems very likely that an intelligence agency is able to get these keys in their possession.

3.3.2 Certificate Authentication with CAs

As a fallback it is still possible to use a X.509 certificate issued by a trusted Certificate Authority. This makes it possible to use DIME on domains that are not secured with DNSSEC. For an attacker like the NSA, getting a valid certificate is not infeasible. For example Stuxnet is believed to be created by the American, and Isreali intelligence service had a valid signature for it's Windows kernel driver.

3.4 User Authentication

The specification suggests using a "pre-nounced hash" to store the password. The password should be used to authenticate to the server but should never be sent to the server. This way an attacker can not get the password by obtaining the account information.

If the account password is not sent over the wire it can be used to encrypt the private keys, if they are stored on the

doi: 10.2313/NET-2015-09-1_03

server. When loosing this password however, the encryption keys would be inevitably lost.

3.5 Metadata and Privacy

The D/MIME format allows to hide the information from the mail servers, if it is not necessary for them to see it. To see the complete metadata, an attack would either need the private keys of both servers or the private key of the sender or recipient.

Even though a lot of metadata is hidden through the DIME format, it is only beneficial if a domain has many users. If a user would like to setup his own server for his domain, any mail from or to his server can still be linked to him. The signets contain fields for onion addresses, so the transport could additionally go over TOR. This could make it more difficult to tell from the connections between the servers which were communicating.

4. RELATED WORK

There are several systems which try to achieve similar goals as DIME, some are presented in the following section.

4.1 Pretty Good Privacy (PGP)

PGP was developed in 1991 and is capable of encrypting and signing messages. It's main advantage over DIME is, that no change in the infrastructure is needed. Two people who want to exchange encrypted messages, can just install a PGP client and send encrypted messages after exchanging keys. The current Message Format Specification can be found in [7].

There are so called PGP keyservers which provide a database of PGP keys, which can be queried. They don't require any authentication so anyone can send a PGP key for any e-mail address. It is possible to query the server for specific key fingerprints.

Dark Mail was designed to fix some of the flaws of PGP. While Dark Mail tries to protect metadata, PGP only encrypts the message text, leaving the metadata in the open. The user signets from Dark Mail are used to verify, that the key for an e-mail address belongs to the account, which makes it harder to forge a key and simplifies the key validation.

This has to be done manually in PGP. Furthermore, combining this with signing the new key with the previous key, when rotating keys, allows to frequently change keys without any real drawbacks. Doing so creates a simple form of forward secrecy. Adding forward secrecy to PGP was tried before[11], but was not successful so far.

4.2 S/MIME

The Secure/Multipurpose Internet Mail Extensions[3](S/MIME) are another solution for encrypting e-mails. Instead of a Web of Trust, like in PGP, certificate authorities are used. In order to sign messages or receive encrypted messages, a user must apply for a certificate from a Certificate Authority. There are some Certificate Authorities that issue them for free.

Most modern mail clients include support for S/MIME. It's advantage over PGP is, that it does not require any user interaction when trusting other certificates.

4.3 Bitmessage

Bitmessage is a P2P messaging system, that tries to provide sender and receiver anonymity. Messages are sent by broadcasting them in the network, which tries to hide the origin of a message.

The message is stored in the network for a certain time, after which it is deleted. To prevent spam flooding the network, every message needs a proof of work. The messages don't disclose the recipient, so every bitmessage client has to try to decrypt every message in the network. The messages are stored in a so called blockchain, similar to that of Bitcoin.

4.4 Pond

Pond[12] is a asynchronous messaging system similar to e-mail with very strong anonymity. Message transmissions are done over TOR[10] and always use the same amount of traffic, to complicate traffic analysis.

Before a message can be sent to someone, that person has to add the sender to a groupkey on the server, which is done to prevent spam.

4.5 Other Solutions

A very exhaustive list of projects trying to provide secure email can be found in [14].

5. CONCLUSION

Overall the Dark Mail initiative is a good improvement compared to the existing mail infrastructure. It allows for a transparent transition, to a more privacy preserving infrastructure.

The D/MIME format allows to hide information from actors, which they don't need to know and the signature protects the complete message including the metadata and not only the content.

The DMTP protocol is designed to leak less metadata and forces TLS encryption.

To be successful, it will take time, since all the exisiting e-mail software needs to be modified to support DIME completely. On a security perspective the strong trust in DNSSEC seems to contradict it's security goal and should be reconsidered. It would be better if additionally another trust anchor would be required.

References

[1] The Internet Email to Support Diverse Service Environments (Lemonade) Profile. RFC 5550, Internet Engineering Task Force, August 2009.

[2] The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, Internet Engineering Task Force, August 2012.

[3] Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751, Internet Engineering Task Force, January 2010.

[4] INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501, Internet Engineering Task Force, March 2003.

[5] Push Extensions to the IMAP Protocol (P-IMAP). Draft, Internet Engineering Task Force, March 2006. URL `https://tools.ietf.org/html/draft-maes-lemonade-p-imap-12`.

[6] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, Internet Engineering Task Force, November 1996.

[7] OpenPGP Message Format. RFC 4880, Internet Engineering Task Force, November 2007.

[8] Simple Mail Transfer Protocol. RFC 5321, Internet Engineering Task Force, October 2008.

[9] D. Davis. Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. In *Proceedings of the 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Massachusetts, USA.*, pages 65–78.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router, 2004.

[11] Internet Engineering Task Force. Forward Secrecy Extensions for OpenPGP, 2001. URL `http://tools.ietf.org/html/draft-brown-pgp-pfs-03`.

[12] A. Langley. Pond Technical Information. URL `https://pond.imperialviolet.org/tech.html`. [Accessed March 30th, 2015].

[13] L. Levison. Dark Internet Mail Environment: Architecture and Specifications, 2014. URL `https://darkmail.info/downloads/dark-internet-mail-environment-december-2014.pdf`. [Accessed March 30th, 2015].

[14] Open Tech Fund. Overview of projects working on next-generation secure email, 22.7.2014. URL `https://github.com/OpenTechFund/secure-email`. [Accessed March 30th, 2015].

doi: 10.2313/NET-2015-09-1_03

Web Privacy

Thomas Mauerer

Betreuer: Marcel von Maltitz
Seminar Future Internet
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: thomas.mauerer@tum.de

KURZFASSUNG

Tracking beschreibt die Methode, Daten über einen Nutzer in Erfahrung zu bringen. [1] Das Ziel ist, Nutzer wieder identifizieren zu können und umfangreiche Profile zu erstellen. Man unterscheidet zwischen expliziten und schließenden Verfahren. [1] Bei expliziten Verfahren werden Informationen direkt auf dem Computer des Nutzers gespeichert, wohingegen schließende Verfahren ihre Informationen aus Systemeigenschaften und durch den Nutzer vorgenommene Konfigurationen erhalten. Durch Einbinden von dritten Parteien, kann auch jenen Tracking ermöglicht werden. In dieser Arbeit werden verschiedene Tracking-Verfahren vorgestellt und es wird aufgezeigt, wie sich der einzelne Nutzer dagegen schützen kann. Alle genannten Verfahren werden im Zuge von Webinteraktion eingesetzt, d.h. sie beschreiben Web-Tracking. Die Themen App-Tracking und Smart-TV-Tracking werden in dieser Arbeit nicht behandelt. Zur besseren Veranschaulichung ist eine Beispielimplementierung vorhanden, welche unterschiedliche Tracking-Verfahren kombiniert.

Schlüsselworte

Cookies, HTML5 Storage, Evercookies, Fingerprinting

1. EINLEITUNG

Der Begriff „Web Privacy" heißt übersetzt so viel wie „Datenschutz im Internet". Konkret geht es bei diesem Thema also darum, welche Informationen über einen Nutzer gesammelt werden und wem diese Informationen zufließen. [20] Diese Seminararbeit beschäftigt sich mit verschiedenen Tracking-Verfahren und möglichen Schutzmaßnahmen dagegen.

Web-Tracking hat neben der eindeutigen Wiedererkennung eines Nutzers auch das Ziel, umfangreiche Profile über diese zu erstellen. Zu einem Profil gehören zum Beispiel Alter, Beruf, Geschlecht, Hobbies, Interessen, politische Einstellung, sexuelle Orientierung, Trinkgewohnheiten, etc. [1] Die gesammelten Daten dienen heutzutage als eine Art Rohstoff, d.h. die Daten können zu Geld verwertet werden. Beispielsweise können dadurch neue Kunden gewonnen werden. Auch ist es für Unternehmen möglich, sich am Markt zu etablieren, da dies ein ständiges Feedback der Nutzer voraussetzt, welches durch Tracking gewonnen wird. Eine weitere Einnahmequelle ist zielgerichtete Werbung. Damit dieses Konzept funktioniert, muss man zuerst die Interessen der Nutzer kennen.

Tracking-Methoden können unterschiedlich gegliedert werden. Eine mögliche Differenzierung ist First-Party-Tracking und Third-Party-Tracking. Als First-Party wird immer die Website bezeichnet, zu der der Nutzer bewusst eine Verbindung aufgebaut hat. Dass diese Website selbst an den Daten des Nutzers interessiert ist, könnte zum Beispiel daran liegen, dass sich die Website über (zielgerichtete) Werbung finanziert. Third-Partys dagegen sind Websites, die Inhalte beisteuern, wenn gleich der Nutzer nicht aktiv eine Verbindung zu diesen aufgebaut hat. Dies funktioniert durch Einbinden einer Third-Party in die Website einer First-Party. Mögliche Gründe hierfür könnten sein, dass die beiden Unternehmen hinter den Websites zusammengehören (z.B. Google und Youtube) oder dass die Third-Party dafür bezahlt, Daten sammeln zu dürfen.

Gerade die genannte Zusammengehörigkeit von Unternehmen spielt auch eine wichtige Rolle beim Tracking. Auf diese Weise können nämlich Daten zusammengeführt werden und noch umfangreichere Profile über einzelne Nutzer erstellt werden. Dies dürfte zum Beispiel auch ein Grund für *Facebook* gewesen sein, 19 Milliarden US-Dollar für die Übernahme von *WhatsApp* aufzubringen, auch wenn Mark Zuckerberg (Gründer und Vorsitzender von Facebook) es damit begründet, die gesamte Welt vernetzen zu wollen. [19]

Grundsätzlich muss Tracking von zwei verschiedenen Seiten betrachtet werden, um einschätzen zu können, ob dies positiv oder negativ ist. Aus Sicht der Webseitenbetreiber hat Tracking durchaus positive Effekte. Daten sind mittlerweile nämlich zu einer Art Zahlungsmittel geworden. Da viele Websites ihre Inhalte für Nutzer kostenlos zur Verfügung stellen, ist schlichtweg eine andere Einnahmequelle erforderlich. Aus Sicht der Nutzer hat Tracking allerdings zumeist einen negativen Beigeschmack. Dies liegt an der Art und Weise, wie Tracking betrieben wird. So wird der Nutzer in der Regel nicht umfassend darüber informiert, welche Daten auf welche Weise erhoben werden und wem diese Informationen zufließen. Diese Seminararbeit hat das Ziel, den Leser über verschiedene Tracking-Verfahren zu informieren und aufzuzeigen, wie man sich gegen solche Verfahren schützen kann. Einen hundertprozentigen Schutz gibt es allerdings nicht (zumindest in den Augen des Autors). Natürlich ist das Thema „Tracking" viel zu umfangreich, um jedes einzelne Verfahren zu beschreiben. Deshalb wurde in dieser Arbeit eine Auswahl der bekanntesten Verfahren (empirische Wahrnehmung des Autors) getroffen.

doi: 10.2313/NET-2015-09-1_04

In Kapitel 2 und 3 werden verschiedene Angriffsmöglichkeiten beschrieben. Kapitel 4 veranschaulicht verschiedene Tracking-Verfahren in Form einer Beispielimplementierung. Kapitel 5 beschäftigt sich anschließend mit verschiedenen Schutzmaßnahmen.

2. ANGRIFFSMÖGLICHKEITEN

2.1 Explizite Verfahren

Als explizite Tracking-Verfahren werden alle Verfahren bezeichnet, bei denen der Tracker Informationen auf dem Computer des Nutzers abspeichert, welche er später zur Wiedererkennung abfragen kann. [1] Explizite Verfahren stellen vermutlich die älteste Art von Tracking dar, weil diese wesentlich weniger komplex sind als schließende Verfahren (siehe Abschnitt 2.2). Hierbei werden im Grunde Mechanismen missbraucht, welche für andere Aspekte konzipiert wurden. Der Nachteil für Tracker besteht allerdings darin, dass diese Verfahren eine eindeutige Spur auf dem Computer des Nutzers hinterlassen. Ein erfahrener Nutzer kann somit die gespeicherten Informationen manipulieren oder ganz löschen.

2.1.1 HTTP-Cookies

Die einfachste Art, Informationen auf dem Computer des Nutzers abzuspeichern, stellen Cookies dar. Ein Cookie ist eine Textdatei, die im Webbrowser gespeichert und zwischen Webserver und Webbrowser ausgetauscht wird. Ein Cookie muss mindestens über einen serverseitig eindeutigen *Namen* verfügen, optional sind auch noch weitere Attribute möglich. Die wichtigsten sind der *Wert* des Cookies (Information), das *Ablaufdatum* und der *Pfad* auf dem Server, für den das Cookie verfügbar ist. Wird kein Ablaufdatum angegeben, existiert das Cookie nur bis zum Ende der Sitzung, also bis der Browser geschlossen wird. Wie viel Speicherplatz für ein Cookie zur Verfügung steht, ist browserabhängig, jedoch schreibt die Spezifikation vor, dass ein Browser mindestens 4096 Bytes zur Verfügung stellen muss. [15]

Der eigentliche Sinn von Cookies besteht darin, verschiedene Einstellungen zu speichern. Zum Beispiel kann die bevorzugte Sprache gespeichert werden, sodass der Webserver bei einem zweiten Besuch diese Information sofort zur Verfügung hat und gleich den „richtigen" Inhalt liefern kann. Allerdings wurde auch schnell erkannt, dass man Cookies dazu nutzen kann, um einen Besucher wieder zu identifizieren, womit man beim Thema Tracking ist. Hierfür muss der Server lediglich jedem Nutzer einen eindeutigen Wert zuweisen, welchen er in einem Cookie speichert. Somit kann der Nutzer zu einem späteren Zeitpunkt unmissverständlich wieder erkannt werden. Es fällt auf, dass die häufigsten vorkommenden Bezeichner *utma, utmb, utmc* und *utmz* heißen. Dies sind bekannte Cookies von *Google Analytics*, welche genau den genannten Zweck erfüllen. [21] Das beweist, dass diese Methode tatsächlich oft eingesetzt wird.

Grundsätzlich kann ein Webserver nur Cookies der eigenen Domain auslesen. Angenommen es existiert auf dem Computer des Nutzers ein Cookie von *www.example-one.com* und ein Cookie von *www.example-two.com*. Ein Dienst, welcher unter *www.example-one.com* läuft, hat also keine Möglichkeit, das Cookie von *www.example-two.com* auszulesen. [15] Gerade für Tracking-Mechanismen ist es aber wichtig, Informationen über Nutzer über die eigenen Websites hinaus zu erlangen, da somit wesentlich mehr Informationen in Erfahrung gebracht und folglich umfangreichere Profile erstellt werden können. Um dies zu bewerkstelligen, werden sogenannte *Third-Party-Cookies* eingesetzt, d.h. Cookies, die von einem anderen Webserver gesetzt werden, als von dem auf dessen Website man gerade ist. Hierzu muss der Webbrowser dazu gebracht werden, eine Verbindung zu der Third-Party aufzubauen, sodass auch diese Cookies setzen kann. Dies geschieht über das Einbinden von externen Elementen und wird im Detail in Abschnitt 3 erläutert.

Cookies sind heutzutage immer noch das meist verwendete Tracking-Verfahren, weil sie einerseits mit sehr geringem Aufwand gesetzt, geändert oder gelöscht werden können und andererseits von jedem Browser unterstützt werden. [2] Außerdem haben die meisten Nutzer nicht das nötige Wissen, mit Third-Party-Cookies umzugehen. Die Nachteile für Tracker sind allerdings das Ablaufdatum und der geringe Speicher.

Nach der EU-Richtlinie 2009/136/EG (Artikel 5 Absatz 3) muss eine Website Nutzer darüber informieren, zu welchem Zweck Cookies gesetzt werden und diese müssen ihre Einwilligung abgeben. [22] In der Regel befinden sich die Bedingungen im Impressum der Website und der Nutzer stimmt implizit durch die Nutzung der Website zu.

2.1.2 HTML5 Storage

HTML5 ist der Nachfolger des bewährten HTML4-Standards. HTML5 bietet neue Features, wie beispielsweise das **<video>**-Element, welches in Zukunft *Flash* ersetzen könnte, oder das **<canvas>**-Element, welches direktes Zeichnen im Browser ermöglicht (siehe dazu auch Abschnitt 2.2.2). Auch wurde mit HTML5 eine neue Speichertechnologie eingeführt, welche sich untergliedern lässt in *Session-Storage* und *Local-Storage* [9]. In dieser Arbeit wird nur *Local-Storage* betrachtet. Dies funktioniert im Grunde ähnlich wie HTTP-Cookies, d.h. auch hier werden Daten im Browser des Nutzers gespeichert. Local-Storage hat allerdings im Vergleich zu Cookies den Vorteil für Tracker, dass der verfügbare Speicher mit fünf Megabyte deutlich höher ist und kein Ablaufdatum vorhanden ist. Gespeicherte Daten sind also so lange verfügbar, bis sie entweder durch die Website oder manuell durch den Nutzer gelöscht werden. [10]

Genau wie bei Cookies kann auch Local-Storage für Tracking eingesetzt werden. Eine Möglichkeit wäre auch hier, eindeutige Identifikationswerte zu speichern, sodass die Website einen Nutzer bei einem zweiten Besuch sofort wieder identifizieren kann. Auch das Einbinden von dritten Parteien kann analog zu Cookies erfolgen. Gerade das fehlende Ablaufdatum macht Local-Storage deutlich attraktiver für Tracking-Zwecke als Cookies.

Im Gegensatz zu Cookies können Local-Storage-Daten allerdings nur clientseitig mittels JavaScript gelesen werden. Die Daten werden auch nicht im HTTP-Header an den Server gesendet. [9] Würde eine Website anstatt Cookies nur Local-Storage einsetzen, hätte dies den Vorteil, dass die Website schneller laden kann, da weniger Daten an den Server gesendet werden. Für Tracker ist dies allerdings eher nachteilig, da der Server somit keinen direkten Zugriff auf die Daten hat. Ein *Workaround*, um die Daten dennoch an den Server zu senden, kann mittels AJAX (Asynchronous JavaScript and

XML) erfolgen. Dies wird auch in der Beispielimplementierung in Abschnitt 4 angewendet.

HTML5 wird mittlerweile von den meisten Browsern unterstützt [10], sodass davon ausgegangen werden kann, dass Tracking-Verfahren mittels Local-Storage in Zukunft zunehmen werden. Dies zeigt auch der Web-Privacy-Zensus von 2014 im Vergleich zu 2012. [2]

2.1.3 Flash-Cookies

Local Shared Objects (LSOs), umgangssprachlich auch *Flash-Cookies* genannt, wurden von der Firma *Adobe* eingeführt, um Daten auf dem Computer des Nutzers zu speichern. [3] Der Name *Flash-Cookies* ist eine Anlehnung an HTTP-Cookies, da diese ähnliche Funktionalitäten bieten. Flash-Cookies können allerdings nur dann gesetzt werden, wenn der Nutzer ein Flash-Plugin für den Browser installiert und aktiviert hat. Da jedoch viele Websites auf Flash-Technologien basieren bzw. basierten, wie beispielsweise das Videoportal *YouTube*, haben fast alle Nutzer - wenn auch nicht immer bewusst - ein entsprechendes Plugin installiert. Zwar verzichtet *YouTube* seit 2015 standardmäßig in den Browsern *Google Chrome, Internet Explorer 11* und *Safari 8* auf Flash und setzt stattdessen HTML5 <video> ein, doch werden alle anderen Browser immer noch über Flash bedient. [23]

Flash-Cookies wurden dafür entwickelt, um Einstellungen wie beispielsweise die Lautstärke zu speichern. Diese Technologie kann aber auch für Tracking-Zwecke eingesetzt werden, indem wiederum eindeutige Identifikationswerte gespeichert werden. Flash-Cookies haben genau wie HTML5 Storage kein Ablaufdatum, d.h. sie sind solange persistent im Speicher, bis sie manuell gelöscht werden. Die Speichergröße liegt mit 100KB zwar deutlich unter dem verfügbaren Speicher von HTML5 Storage, jedoch höher als bei HTTP-Cookies. [3]

Außerdem macht die Tatsache, dass Flash-Cookies nicht an einem browserspezifischen Ort, sondern zentral auf der Festplatte des Nutzers gespeichert werden, diese sehr attraktiv für Tracking. Aus diesem Grund können gespeicherte Daten browserunabhängig verwertet werden. Wenn ein Nutzer also verschiedene Browser verwendet, um im Internet zu surfen, sind seitenübergreifende Tracking-Verfahren, welche auf Flash-Cookies basieren, mächtiger als beispielsweise HTTP-Cookies oder HTML5 Storage.

Die genannte Browserunabhängigkeit ist mit Sicherheit auch ein Grund dafür, dass Flash-Cookies immer wieder mit dem Stichwort *Respawning* in Verbindung gebracht werden. Respawning bezeichnet die Fähigkeit, durch den Nutzer gelöschte Cookies wiederherzustellen. Hierzu muss im Grunde nur eine exakte Kopie des HTTP-Cookies in einem Flash-Cookie gespeichert werden. Auch kann durch Respawning die Browserabhängigkeit von HTTP-Cookies „überwunden" werden. Besucht ein Nutzer beispielsweise eine Website mit Mozilla Firefox und anschließend mit Google Chrome, kann die Website mithilfe von Respawning in Firefox gesetzte Cookies sofort in Chrome übertragen und somit den Nutzer wieder identifizieren. [4]

Insgesamt bieten Flash-Cookies also eine gute Ausgangssituation für Tracking. Dies liegt vermutlich auch daran, dass die meisten Nutzer überhaupt nicht wissen, dass Flash-Cookies auf ihrem Computer gespeichert sind. Der Web-Privacy-Zensus [2] zeigt allerdings, dass Flash-Cookies seit 2012 weniger geworden sind. Dies liegt daran, dass Flash aufgrund von Sicherheitslücken immer wieder in der Kritik steht und seit der Einführung von HTML5 durch das **<video>**-Element eine einfache Alternative zu Flash zur Verfügung steht. [10] Der Streaming-Dienst *Netflix* zum Beispiel unterstützt HTML5 offiziell zumindest bereits für den Internet Explorer und Google Chrome für Windows und Safari und Chrome für Mac OS X. [17]

2.1.4 Evercookies

Evercookies sind keine neue Technologie, sondern lediglich eine Kombination aus den bereits genannten Verfahren und noch weiteren. Aus Sicht eines Trackers ist es wünschenswert, dass Daten persistent auf dem Computer eines Nutzers gespeichert werden. Dazu zählt, dass die Daten kein Ablaufdatum haben und sie nicht vom Nutzer gelöscht werden können. Dies wird theoretisch durch einen Evercookie erreicht. [16] Nutzer mit viel Hintergrundwissen können allerdings trotzdem in der Lage sein, einen Evercookie zu löschen.

Das Ziel eines Evercookies ist es, sich an so vielen verschiedenen Stellen wie möglich im Computer zu manifestieren, sodass der Nutzer den Überblick verliert. Kombiniert wird die redundante Speicherung anschließend mit Respawning-Techniken (siehe dazu Abschnitt Flash-Cookies 2.1.3). Eine JavaScript-basierte API, welche einen Evercookie erzeugt, wurde von Samy Kamkar entwickelt. Laut Angaben auf seiner Homepage nutzt ein Evercookie bis zu 16 verschiedene Technologien, um sich redundant auf dem Computer des Nutzers zu speichern, darunter HTTP-Cookies, Flash-Cookies, HTML5 Storage, ETags und Silverlight-Cookies, wobei die beiden letzten Verfahren in dieser Arbeit nicht behandelt werden. [16]

Aufgrund der persistenten Speicherung bieten Evercookies somit die mit Abstand besten Voraussetzungen für explizite Tracking-Verfahren. Es ist allerdings unklar, wie viele Websites tatsächlich Evercookies einsetzen.

2.2 Schließende Verfahren

Als schließende Tracking-Verfahren werden alle Verfahren bezeichnet, bei denen der Tracker versucht, einen Nutzer anhand seiner Konfigurations- und Systemeigenschaften zu identifizieren. Schließende Verfahren zählen als fortgeschrittene Tracking-Verfahren, da hierbei im Gegensatz zu expliziten Verfahren (siehe Abschnitt 2.1) keine Spur auf dem Computer des Nutzers hinterlassen wird. [1] Dies macht es für den Nutzer bedeutend schwieriger, sich gegen Tracking zu schützen.

2.2.1 Klassisches Fingerprinting

Das erste Beispiel eines schließenden Verfahrens beschreibt klassisches Fingerprinting. Hierbei versucht die Tracking-Website, meistens mithilfe von JavaScript Eigenschaften des Browsers bzw. des Computers in Erfahrung zu bringen. Die gesammelten Daten können anschließend in einer Datenbank gespeichert und in Zukunft dazu verwendet werden, um einen Nutzer bei einem zweiten Besuch wieder zu identifizieren. Da der durchschnittliche Nutzer nur selten seine

Konfigurationseigenschaften ändert, ist dies auch ein sehr zuverlässiger Tracking-Mechanismus. [5]
Die Beispielimplementierung in Abschnitt 4 zeigt außerdem, dass der Nutzer keinen Einfluss darauf hat, an welche dritten Parteien die gesammelten Daten weitergegeben werden, wenn er nicht gerade aktiv versucht, die Erhebung der Daten zu verhindern.

Das Projekt *Panopticlick* von der Electronic Frontier Foundation [7] hat gezeigt, dass bereits wenige Eigenschaften ausreichend sind, um einen Nutzer eindeutig identifizieren zu können. Zu diesen Eigenschaften gehören die Zeitzone, die Bildschirmauflösung und Farbtiefe, die Liste der installierten Browser-Plugins und System Fonts (Schriftarten), sowie der User-Agent, welcher standardmäßig ohnehin im HTTP-Header an den Server gesendet wird.

Viele der genannten Eigenschaften können bereits ohne Aufwand mithilfe des JavaScript-Objekts *navigator* ausgelesen werden. Lediglich die Liste der installierten Browser-Plugins und System Fonts benötigen tiefgreifendere Techniken.

Klassisches Fingerprinting kann außerdem dazu verwendet werden, einen Nutzer über verschiedene Endgeräte hinweg zu identifizieren. Dies ist allerdings aus Tracker-Sicht gesehen sehr schwierig und setzt Tracking über einen langen Zeitraum voraus. Hierbei wird nämlich davon ausgegangen, dass unterschiedliche Nutzer unterschiedliche Gewohnheiten bezüglich Surfverhalten und Konfigurationen haben, was diese eindeutig identifizierbar macht. Erkennt man beispielsweise, dass ein Nutzer immer wieder die gleichen Websites aufruft (möglicherweise auch in der gleichen Reihenfolge), kann man nach langer Beobachtung darauf schließen, dass es sich um den selben Nutzer handeln muss und dieser lediglich verschiedene Endgeräte verwendet. [1]

2.2.2 Canvas-Fingerprinting

Ein relativ neues Verfahren des Fingerprintings beschreibt das sogenannte Canvas-Fingerprinting. Das **<canvas>**-Element wurde mit HTML5 eingeführt und ermöglicht direktes Zeichnen im Browser mittels JavaScript. Canvas-Fingerprinting bietet ein hohes Potential für Tracking, für welches es bis heute (Stand 2015) keine optimalen Gegenmaßnahmen gibt. [6] (Quelle von 2012, seitdem allerdings keine neuen brauchbaren Verfahren bekannt)

Beim Canvas-Fingerprinting wird in der Regel ein beliebiger Text mithilfe des **<canvas>**-Elements im Browser des Nutzers gezeichnet. Der Fingerprint ergibt sich dadurch, dass jeder Browser aufgrund von Soft- und Hardwarekonfigurationen den Text unterschiedlich darstellt. So spielen beispielsweise das verwendete Betriebssystem, die Liste der installierten System Fonts, die Grafikkarte oder auch die Grafikkartentreiber eine Rolle. [6] Das entstandene Bild wird anschließend mit einer Hash-Funktion in eine binäre Darstellung verwandelt und an den Server zur Verwertung geschickt.

Mowery und Shacham berichteten in ihrer Arbeit *Pixel Perfect: Fingerprinting Canvas in HTML5* zum ersten mal über Canvas-Fingerprinting. Die beiden Autoren haben gezeigt, dass Unterschiede in der Darstellung sogar mit bloßem Auge deutlich erkennbar sind. Spätestens aber nach dem Hashen in eine binäre Darstellung ergeben sich eindeutige Unterscheidungen. [6] Nach empirischer Wahrnehmung ändern sich die Systemeigenschaften beim durchschnittlichen Nutzer nur selten. Dies zeigt, dass Canvas-Fingerprinting ein effektives Verfahren ist, Nutzer eindeutig wieder zu identifizieren.

Für den Fingerprint hat das gezeichnete Bild überhaupt keine Bedeutung, sondern lediglich die binäre Darstellung. Deshalb wird Canvas-Fingerprinting meistens im Hintergrund und für den Nutzer nicht sichtbar vollzogen. Dies geschieht im Bruchteil einer Sekunde.
2014 betrieben circa 5,5% der am besten bewerteten Websites aktives Canvas-Fingerprinting. [4]

2.2.3 History-Sniffing

History-Sniffing beschäftigt sich in erster Linie nicht damit, einen Nutzer wieder zu identifizieren, sondern damit, weitere Informationen über einen Nutzer zu erfahren. Konkret geht es darum, in Erfahrung zu bringen, welche Websites ein Nutzer bereits besucht hat. Dies gibt nämlich Aufschluss über die Interessen des Nutzers und ist somit von großer Bedeutung für Tracker.

Die meisten Browser speichern in der Standardeinstellung einen Verlauf über sämtliche besuchte Websites. Diese Einstellung wird auch von den meisten Nutzern nicht geändert. Von Sniffing-Verfahren, welche Sicherheitslücken ausnützen, um den Verlauf des Browser auszulesen, soll in dieser Arbeit abgesehen werden. Stattdessen soll ein anderes Verfahren beschrieben werden, welches allerdings aufgrund der Reaktion der Browserhersteller so nicht mehr funktioniert. Das Verfahren basierte darauf, dass Browser besuchte Hyperlinks in einer anderen Farbe darstellen, als nicht besuchte. Mithilfe von JavaScript war es möglich, die Farbe eines Hyperlinks zu ermitteln und daraus zu schließen, welche Seiten der Nutzer bereits besucht hat. [1] Browserhersteller haben aus datenschutzrechtlichen Gründen darauf reagiert. In Mozilla Firefox funktioniert beispielsweise die Funktion *getComputedStyle* nicht mehr, sodass die Methode keine wirkliche Relevanz mehr hat. [8]

Das Verfahren zeigt allerdings, wie kreativ Tracker sind und wie mächtig JavaScript ist, sodass mit Sicherheit davon ausgegangen werden kann, dass in Zukunft wieder ähnliche Verfahren entwickelt werden.

3. EINBINDEN VON DRITTEN PARTEIEN

Unter „Einbinden von dritten Parteien" versteht man, dass ein Webseitenbetreiber im Hintergrund Elemente eines Trackers einbindet und somit eine Verbindung zu diesem herstellt. Das Ziel dabei ist es, dritten Parteien die Möglichkeit zu bieten, Tracking zu betreiben. Auf diese Weise können auch dritte Parteien explizite oder schließende Tracking-Verfahren anwenden. [1]

In vielen Fällen wird das Einbinden externer Elemente überhaupt nicht verschleiert, sondern direkt durch sichtbare Elemente durchgeführt. Den meisten Nutzern ist allerdings nicht bewusst, dass hierdurch auch Tracking betrieben wird.

Das technische Werkzeug hierfür sind *I-Frames*, welche mit HTML4 eingeführt wurden. Hierbei handelt es sich um ein

doi: 10.2313/NET-2015-09-1_04

Element, welches das parallele Laden aus verschiedenen Quellen ermöglicht. [1] Beispielsweise stammen Grafiken oder Videos oft nicht direkt von der Website, sondern von dritten Parteien, wie *YouTube*. Auch befinden sich auf vielen Websites Wegbeschreibungen in Form von eingebundenen *Google Maps*-Landkarten. Neben Tracking-Zwecken hat dies auch den einfachen Grund, um Speicherplatz einzusparen, da die Inhalte somit nicht redundant auf jedem einzelnen Server gespeichert werden müssen.

Auf vielen Websites befinden sich außerdem sogenannte *Social-Plugins* von *Facebook, Twitter* und *Google+*. Diese Plugins bieten noch bessere Tracking-Möglichkeiten, da hiermit besuchte Websites eines Nutzers in Erfahrung gebracht werden können und diese Daten mit der echten Identität des Nutzers verknüpft werden können. [1] Voraussetzung hierfür ist natürlich, dass der Nutzer auf einer dieser sozialen Plattformen registriert ist und hierbei seine echten Daten angegeben hat. Bei 1,39 Milliarden Facebook-Nutzern (Stand: 2015) [18] ist die Wahrscheinlichkeit hierfür allerdings sehr hoch. Dies zeigt, wie effektiv diese Tracking-Methode ist.

Die zweite Art, Elemente eines Trackers einzubinden, sind unsichtbare Elemente. Hierfür kann zum Beispiel ein PHP-Skript einer dritten Partei in die Website eingebunden werden. Dies wird auch in der Beispielimplementierung in Abschnitt 4 gezeigt. Da PHP serverseitig ausgewertet wird, hat der Nutzer somit auch keine Möglichkeit, zu erkennen, welche Daten an das eingebundene PHP-Skript übergeben werden.

4. BEISPIELIMPLEMENTIERUNG

Im Folgenden werden einige der genannten Tracking-Techniken in einer Beispielimplementierung kombiniert. Konkret werden HTML5 Storage, Fingerprinting und die Einbindung eines externen PHP-Skripts verwendet. Das Beispiel wurde in einem lokalen Netzwerk getestet. Der Aufbau wird in Abbildung 1 verdeutlicht.

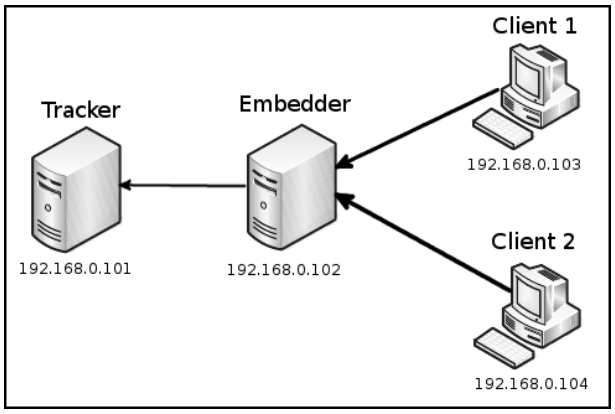

Abbildung 1: Aufbau zur Implementierung

Abbildung 1 zeigt, dass der Embedder-Server als First-Party eine Website anbietet, auf welche zwei verschiedene Clients zugreifen. Im Hintergrund wird eine Verbindung zu dem Tracker-Server aufgebaut und diesem verschiedene Daten gesendet. Der Tracker-Server wiederum kann die empfangenen Daten anschließend in einer Datenbank speichern. Er agiert hier also ausschließlich als Third-Party.

Listing 1 zeigt einen Ausschnitt aus *index.php* des Embedder-Servers. Die Zeilen 16 bis 19 prüfen, ob der Nutzer die Website zum ersten Mal besucht. Ist dies der Fall (oder hat der Nutzer Local-Storage zurückgesetzt), wird eine Local-Storage Variable namens „counter" angelegt und diese bei jedem neuen Besuch ausgelesen und um eins inkrementiert.

In den Zeilen 21 und 22 wird Fingerprinting betrieben. Da dieses Beispiel nur zur Veranschaulichung dienen soll, werden hier lediglich die Bildschirmauflösung und der User-Agent ausgelesen. Das Paper *User Tracking on the Web via Cross-Browser Fingerprinting* [5] hat allerdings gezeigt, dass alleine der User-Agent schon in etwa 50 Prozent der Fälle ein eindeutiges Erkennungsmerkmal ist.

Die Zeilen 24 bis 26 lösen das Problem, die mit JavaScript ausgelesenen Variablen an PHP zu übergeben. Hierfür wird AJAX (Asynchronous JavaScript and XML) verwendet, d.h. die Daten werden in einem XMLHttpRequest verpackt über *GET-Variablen* zurück an den Index gesendet.

Im PHP-Teil (Zeilen 1 bis 8) können diese GET-Variablen nun verwendet werden. Zusätzlich wird noch die IP-Adresse des Nutzers ausgelesen. Alle diese Daten (IP-Adresse, Anzahl der Besuche, Bildschirmauflösung und User-Agent) werden nun an den Tracker übergeben (wieder über GET-Variablen). Hierzu wird lediglich ein PHP-Skript des Trackers mithilfe des *file*-Befehls eingebunden.

Auf der Seite des Tracker-Servers können die empfangenen Daten nun ausgelesen und in einer Datenbank gespeichert werden. Existiert eine IP-Adresse bereits in der Datenbank, wird lediglich die Anzahl der Besuche aktualisiert. Natürlich könnte der Tracker auch zusätzlich als First-Party eine eigene Website anbieten und somit selbst Informationen über die Besucher ermitteln. Diese Daten können anschließend mit denen aus der Datenbank abgeglichen werden, um auf diese Weise umfangreichere Erkenntnisse über die Nutzer zu erhalten.

```
1  <?php
2    $ip = $_SERVER['REMOTE_ADDR'];
3    $count = $_GET['count'];
4    $resolution = $_GET['resolution'];
5    $userAgent = $_GET['userAgent'];
6
7    @file('http://192.168.0.101/tracker/index.php
         ?ip='.$ip.'&count='.$count.'&resolution='
         .$resolution.'&userAgent='.urlencode(
         $userAgent));
8  ?>
9
10 [...]
11
12 You have visited this page <span id="count"></
       span> times.
13
14 <script type="text/javascript">
15
16   if(!localStorage['counter'])
17     localStorage['counter'] = 0;
18   localStorage['counter'] = parseInt(
         localStorage['counter']) + 1;
19   document.getElementById('count').textContent
         = localStorage['counter'];
20
```


doi: 10.2313/NET-2015-09-1_04

```
21    var resolution=[screen.width,screen.height];
22    var userAgent = navigator.userAgent;
23
24    xmlhttp = new XMLHttpRequest();
25    xmlhttp.open('GET','index.php?count='+
          localStorage['counter']+'&resolution='+
          resolution+'&userAgent='+userAgent, true)
          ;
26    xmlhttp.send();
27
28  </script>
```

Listing 1: Auszug aus Embedder: Index.php

Das Beispiel verdeutlicht, mit welch einfachen Mitteln Tracking möglich ist und wie „machtlos" der Nutzer dagegen ist. Schaut sich der Nutzer beispielsweise den Seitenquelltext der Website in seinem Browser an, könnte er lediglich noch erkennen, dass hier verdächtige Werte an den Server zurückgesendet werden. Was der PHP-Teil mit den Daten allerdings macht, ist für den Nutzer nicht erkennbar, da PHP serverseitig ausgewertet wird und somit überhaupt nicht im Seitenquelltext erscheint. Zu beachten ist allerdings, dass die Implementierung wirklich nur zur Veranschaulichung dienen soll und bewusst einfach gehalten wurde. Es wurde beispielsweise kein Fokus auf Sicherheitsaspekte (z.B. SQL-Injection) gelegt. Auch sollte in einem echten Anwendungsfall nicht der *file*-Befehl verwendet werden. Dies hat nämlich den Nachteil, dass die Website auf die Antwort des Tracker-Servers warten wird. Ist der Tracker-Server nicht erreichbar, wird die komplette Website nicht laden können.

5. SCHUTZMAßNAHMEN

Die vorangegangenen Abschnitte haben sowohl gezeigt, wie viele Tracking-Arten es gibt, als auch dass Personen, welche Tracking betreiben, immer kreativer werden. Das beste Beispiel hierfür ist *Canvas-Fingerprinting* (siehe Abschnitt 2.2.2). Es verlangt einiges an Kreativität, ein Tracking-Verfahren zu konstruieren, welches auf einem Element zum direkten Zeichnen im Browser basiert.

Aus diesen Gründen kann Tracking wohl kaum komplett verhindert werden. Dennoch gibt es Möglichkeiten für den Nutzer, Tracking zumindest einzuschränken.

5.1 Strategie: Blockieren

Die einfachste und zugleich auch effektivste Art, etwas zu verhindern, besteht darin, es komplett zu blockieren.

5.1.1 Explizite Verfahren

Alle guten Browser bieten die Möglichkeit, *Third-Party-Cookies* zu blockieren. Die Standard-Einstellung lässt in den meisten Fällen allerdings Third-Party-Cookies zu, sodass dies bereits ein technisches Wissen des Nutzers voraussetzt. In der Regel hat der Nutzer durch das Blockieren keinen Nachteil, da Third-Party-Cookies meistens für Tracking-Zwecke eingesetzt werden. [1] Gelegentlich kann es aber auch zu Funktionseinbußen kommen. Beispielsweise werden Third-Party-Cookies benötigt, um Kommentare zu YouTube-Videos abzugeben. Dies liegt daran, dass das Kommentieren bei YouTube über einen *Google+*-Account abgehandelt wird, wobei *Google+* in diesem Fall dann die Rolle einer Third-Party einnimmt und deshalb Third-Party-Cookies erfordert. [24]

Dass First-Party-Cookies auch für Tracking-Zwecke eingesetzt werden können, wurde bereits in Abschnitt 2.1.1 erklärt. Dies zu verhindern, ist nicht so einfach möglich. Zwar bieten die meisten Browser auch die Möglichkeit, First-Party-Cookies zu blockieren, doch funktionieren dadurch einige Websites nicht mehr richtig. Beispielsweise setzen *Facebook* oder *Amazon* Cookies voraus, um sich überhaupt anmelden zu können. Eine Kompromisslösung, welche allerdings einen höheren Verwaltungsaufwand nach sich zieht, ist das Verwenden einer *White-List*. Das bedeutet, man erlaubt das Setzen von Cookies nur den Websites, denen man vertraut bzw. die ohne Cookies nicht funktionieren. Alle anderen Websites werden daran gehindert, Cookies zu setzen.

Das Verwenden einer White-List (oder auch einer Black-List) kann für einen Nutzer allerdings problematisch sein in Bezug auf klassisches Fingerprinting. Das individuelle Anlegen dieser Listen kann ausschlaggebend dafür sein, dass man eindeutig wieder identifiziert wird. Schützt man sich also gegen ein bestimmtes Verfahren, macht man sich gleichzeitig angreifbar für ein anderes Verfahren. Dies zeigt ein weiteres Mal, dass Tracking nicht komplett verhindert werden kann.

In Mozilla Firefox ist durch das Blockieren von Cookies auch HTML5 Storage betroffen. Für den Umgang mit Flash-Cookies benötigt es allerdings einen lokalen Einstellungsmanager für den Flash-Player [1], da Flash-Cookies browserunabhängig gespeichert werden.

Ein weiterer Ansatz zum Verhindern von expliziten Verfahren, ist die Verwendung des *Private-Modes* (teilweise Inkognito-Mode genannt). Dies wird von allen guten Browsern angeboten, darunter *Mozilla Firefox, Google Chrome, Safari* oder auch der *Internet-Explorer*. Durch diesen Modus werden keine Informationen auf dem Computer des Nutzers dauerhaft gespeichert. [25] Cookies, HTML5-Storage und ähnliches sind nur temporär verfügbar und werden nach Beenden des Modus gelöscht. Auf diese Weise erfährt der Nutzer keine Funktionseinbußen, obwohl explizites Tracking weitestgehend unterbunden wird. Der *Private-Mode* bietet allerdings keinen Schutz gegen schließende Verfahren.

5.1.2 Schließende Verfahren

Schließende Tracking-Verfahren, wie klassisches Fingerprinting oder Canvas-Fingerprinting, beruhen in der Regel immer auf JavaScript. Mit der Erweiterung *NoScript* [11] für Mozilla Firefox ist es möglich, JavaScript komplett zu blockieren. Dies verhindert die genannten Tracking-Verfahren. Allerdings gibt es heutzutage fast keine Websites mehr, die ohne JavaScript richtig funktionieren. Somit ist diese Methode eher ein theoretischer, als praktisch anwendbarer Schutz. Als Kompromiss kann auch hier wieder eine White-List verwendet werden.

Ein weiteres Addon für Mozilla Firefox, welches angeblich Canvas-Fingerprinting verhindern soll, heißt *CanvasBlocker* [12] und wurde im Zuge dieser Seminararbeit genauer getestet. Obwohl das Addon überwiegend positive Bewertungen erhalten hat, war das Ergebnis der Tests nur mäßig zufriedenstellend. Dies verdeutlicht ein weiteres Mal, dass es gegen Canvas-Fingerprinting bis heute (Stand: 2015) keine wirklich

brauchbare Gegenmaßnahme gibt.
Das Tool wurde in der Einstellung „um Erlaubnis fragen" getestet. Auf vielen Websites, darunter *bundesliga.de, kawasaki.de* und *audi.de*, wurden versteckte Canvas-Elemente gefunden und diese nach Nachfrage blockiert. Ein Unterschied der Funktionalität dieser Seiten mit aktiviertem oder deaktiviertem CanvasBlocker konnte nicht festgestellt werden. Dies deutet stark auf Canvas-Fingerprinting hin, muss aber nicht zweifellos der Fall sein. Beispielsweise wurde auf der Website *maps.google.de* auch ein verstecktes Canvas-Element gefunden. Es stellte sich jedoch heraus, dass dieses versteckte Element für die Suchmaske verantwortlich ist und die Website somit mit aktiven CanvasBlocker unbrauchbar ist (abgesehen von der Verwendung einer White-List).

```
1 <canvas id="can" width="100" height="100">
2 </canvas>
3
4 <script type="text/javascript">
5   var canvas = document.getElementById("can");
6   var context = canvas.getContext("2d");
7   context.fillRect(0,0,100,100);
8 </script>
```

Listing 2: JavaScript Code für ein Quadrat

Listing 2 zeigt einen JavaScript Code, welcher ein 100x100 Pixel großes Quadrat zeichnen sollte. Aus einem unersichtlichen Grund wurde der Code mit aktiviertem CanvasBlocker ohne Nachfrage komplett blockiert, obwohl die Einstellung „um Erlaubnis fragen" aktiviert war. Dies zeigt deutlich, dass das Tool mit Vorsicht zu genießen ist und nicht optimal gegen Canvas-Fingerprinting schützt.

Auch der *Tor Browser* versucht aktiv, Canvas-Fingerprinting zu verhindern. Die verwendete Strategie ist die gleiche wie beim *CanvasBlocker*. Der Browser frägt nach, ob ein <canvas>-Element eingebunden werden darf, falls eine Website ein solches verwenden möchte. [26] Der Tor Browser wurde allerdings in dieser Seminararbeit nicht getestet.

Eines der beliebtesten Browser-Addons überhaupt ist *Adblock-Plus* [13], welches für diverse Browser verfügbar ist. Das Addon hat hauptsächlich die Aufgabe, in Websites integrierte Werbungen zu blockieren. Technisch realisiert wird dies über verschiedene *Black-Lists*. Das bedeutet, dass die Website nach Elementen durchsucht wird, welche sich in einer der verwendeten Black-Lists befinden und diese daraufhin blockiert werden. *Adblock-Plus* kann aber auch als Schutz gegen Tracking betrachtet werden. Zum einen existiert eine Black-List[1], welche Social-Plugins blockiert, zum anderen wird Werbung blockiert, welche auch für Tracking eingesetzt werden kann. Dies liegt daran, dass Werbung in der Regel von dritten Parteien eingebunden wird und somit Third-Party-Tracking betrieben werden kann (siehe Abschnitt 3), was dadurch unterbunden wird.

5.2 Strategie: Ergebnisse verfälschen

Eine zweite Strategie ist es, Tracking grundsätzlich nicht zu blockieren, sondern unbrauchbare Ergebnisse zu liefern.

[1]https://easylist-downloads.adblockplus.org/fanboy-social.txt

Auf diese Weise haben die durch Tracking erhobenen Daten keine Bedeutung. Inwieweit diese Technik realisiert werden kann und brauchbar ist, konnte im Zuge dieser Seminararbeit nicht geklärt werden. Dies wird für zukünftige Arbeit offen gelassen. Dass es technisch möglich sein muss, kann am Beispiel eines *User Agent Overriders* [14] gezeigt werden. Hiermit ist es beispielsweise möglich, auf einem Linux-System mit Firefox, den User-Agent eines Windows-Systems mit Internet-Explorer anzunehmen. Auf ähnliche Weise sollte es möglich sein, durch JavaScript ausgelesene Daten zu verfälschen. Hieran schließt sich allerdings die Frage an, wie unterschieden werden soll, wann echte und wann verfälschte Ergebnisse geliefert werden. Natürlich haben die ausgelesenen Daten neben Tracking auch sinnvolle Zwecke. Beispielsweise kann die Bildschirmauflösung dazu verwendet werden, um die Website optimal darzustellen und sollte aus diesen Gründen nicht manipuliert werden.

Eine weitere Methode, welche auf der gleichen Strategie basiert, ist die Verwendung eines Anonymisierungsproxys. Auch hier werden Ergebnisse verfälscht bzw. verschleiert, in diesem Fall die eigene IP-Adresse. Bei der Verwendung eines Proxys wird eine Anfrage an einen Server nicht direkt ausgeführt, sondern über den Proxy umgeleitet. Das bedeutet, dass der Proxy als Stellvertreter die Anfrage an den Server stellt und die Antwort an den eigentlichen Nutzer sendet. [1] IP-basierten Tracking-Verfahren kann auf diese Weise entgegengewirkt werden, da die eigene IP-Adresse dem Server nicht bekannt ist.

6. VERWANDTE ARBEITEN

Diese Arbeit hat in erster Linie das Ziel, einen Gesamtüberblick über Tracking und Schutzmaßnahmen zu vermitteln. Deshalb werden hier viele verschiedene Verfahren erläutert. Natürlich kann auf diese Weise aber nicht jedes einzelne Verfahren im Detail erklärt werden. An dieser Stelle soll auf andere Arbeiten verwiesen werden. Beispielsweise werden in *Pixel Perfect: Fingerprinting Canvas in HTML5* von K. Mowery und H. Shacham [6] viele Details über Canvas-Fingerprinting erklärt. Auch die Arbeit *User Tracking on the Web via Cross-Browser Fingerprinting* [5] beschäftigt sich mit dem Thema *Fingerprinting*. Die vier Autoren haben ein ähnliches Projekt wie *Panopticlick* [7] entwickelt und interessante Statistiken dazu erstellt. Für mehr Details im Bereich expliziter Tracking-Verfahren kann die Arbeit *The Web Never Forgets: Persistent Tracking Mechanisms in the Wild* [4] herangezogen werden. Hier werden Aspekte wie Cookie-Syncing, Evercookies und Flash behandelt.

7. ZUSAMMENFASSUNG

Web-Tracking ist eine allgegenwärtig eingesetzte Technik, um Informationen und Interessen eines Nutzers in Erfahrung zu bringen. Ziel ist es unter anderem, einen Nutzer bei einem zweiten Besuch einer Website wieder identifizieren zu können.

Explizite Verfahren sind Verfahren, bei denen der Tracker Informationen auf dem Computer des Nutzers abspeichert. Bei den gespeicherten Werten handelt es sich meistens um eindeutige Identifikationswerte. Dies kann in Form von HTTP-Cookies, HTML5 Storage, Flash-Cookies oder sogenannten Evercookies geschehen.

Schließende Verfahren sind Verfahren, bei denen der Tracker einen Nutzer anhand von Konfigurations- und Systemeigenschaften wieder identifizieren kann. Beim klassischen Fingerprinting werden mittels JavaScript verschiedene Werte ausgelesen, um den Nutzer eindeutig zu identifizieren. Dazu gehören der User-Agent, die Liste der installierten Browser-Plugins sowie System Fonts, die Bildschirmauflösung oder die Zeitzone. Beim Canvas-Fingerprinting wird dagegen ein eindeutiger Fingerprint mithilfe des HTML5 Canvas-Elements erstellt. Dies ist möglich, da jeder Browser aufgrund von Hard- und Softwarekonfigurationen Text in einem Canvas-Element anders darstellt und somit eindeutig ist.

Unter „Einbinden von dritten Parteien" versteht man, dass ein Webseitenbetreiber Elemente eines Trackers einbindet und somit eine Verbindung zu diesem herstellt. Ziel hierbei ist es, dritten Parteien Tracking zu ermöglichen. Dies kann in Form von Grafiken, Videos oder auch PHP-Skripten erfolgen.

Um sich gegen Tracking zu schützen, gibt es zwei verschiedene Strategien. Bei der ersten Strategie versucht man, Tracking komplett zu blockieren. Dies kann zum Beispiel durch statisches *Black-Listing* oder *White-Listing* geschehen. Bei der zweiten Strategie versucht man, unbrauchbare Daten an Tracker zu übergeben, sodass Tracking quasi keine Bedeutung hat. Dies kann zum Beispiel durch die Verwendung von Anonymisierungsproxys erreicht werden.

Da Personen, die Tracking betreiben allerdings sehr kreativ sind und zudem sehr viele verschiedene Tracking-Verfahren existieren, wird es nicht möglich sein, Tracking komplett zu verhindern, ohne dass dabei die Funktionalität der Websites leidet.

8. LITERATUR

[1] M. Schneider, M. Enzmann, M Stopczynski, *Web-Tracking-Report 2014*, 2014, FRAUNHOFER VERLAG

[2] I. Altaweel, J. Cabrera, H. Choi, K. Ho, N. Good, C. Hoofnagle, *Web Privacy Census: HTML5 Storage Takes the Spotlight As Flash Returns*, 2014

[3] M. Ayenson, D. Wambach, A. Soltani, N. Good, C. Hoofnagle, *Flash Cookies and Privacy II*, 2011

[4] G. Acar, E. Eubank, S. Englehardt, M. Juarez, A. Narayanan, C. Diaz, *The Web Never Forgets: Persistent Tracking Mechanisms in the Wild*, 2014

[5] K. Boda, A. Földes, G. Gulyás, S. Imre, *User Tracking on the Web via Cross-Browser Fingerprinting*, 2012, Budapest University of Technology and Economics

[6] K. Mowery, H. Shacham, *Pixel Perfect: Fingerprinting Canvas in HTML5*, 2012, University of California

[7] *Panopticlick: How Unique - and Trackable - Is Your Browser*, `https://panopticlick.eff.org` - last visited: 5. Mai 2015

[8] *Privacy-related Changes coming to CSS:visited*, `https://hacks.mozilla.org/2010/03/privacy-related-changes-coming-to-css-vistited/` - last visited: 5. Mai 2015

[9] *Web Storage*, `http://www.w3.org/TR/webstorage/` - last visited: 5. Mai 2015

[10] *Dive Into HTML5*, `http://diveintohtml5.info/storage.html` - last visited: 5. Mai 2015

[11] *NoScript*, `https://noscript.net/` - last visited: 5. Mai 2015

[12] *CanvasBlocker*, https://addons.mozilla.org/de/firefox/addon/canvasblocker/ - last visited: 5. Mai 2015

[13] *Adblock Plus: Für ein Web ohne nervige Werbung*, `https://adblockplus.org/de/` - last visited: 5. Mai 2015

[14] *User Agent Overrider*, `https://addons.mozilla.org/de/firefox/addon/user-agent-overrider/` - last visited: 5. Mai 2015

[15] *HTTP State Management Mechanism*, `http://tools.ietf.org/html/rfc6265` - last visited: 5. Mai 2015

[16] *Evercookie - never forget*, `http://samy.pl/evercookie/` - last visited: 5. Mai 2015

[17] *Netflix-Systemvoraussetzungen*, `https://help.netflix.com/de/node/23742` -last visited: 5. Mai 2015

[18] *Börsenbericht: Die ersten offiziellen Facebook-Nutzerzahlen im Jahr 2015*, `http://allfacebook.de/zahlen_fakten/facebook-nutzerzahlen-2015` - last visited: 5. Mai 2015

[19] *Facebook übernimmt WhatsApp*, `https://www.tagesschau.de/wirtschaft/facebook460.html` - last visited: 5. Mai 2015

[20] *Gabler Wirtschaftslexikon*, http://wirtschaftslexikon.gabler.de/Definition/datenschutz.html - last visited: 5. Mai 2015

[21] *Google Analytics Cookie Usage on Websites*, https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage - last visited: 5. Mai 2015

[22] *Amtsblatt der Europäischen Union*, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:de:PDF - last visited: 5. Mai 2015

[23] *YouTube now defaults to HTML5 <video>*, `http://youtube-eng.blogspot.de/2015/01/youtube-now-defaults-to-html5_27.html` - last visited: 5. Mai 2015

[24] *Commenting in Youtube*, `http://superuser.com/questions/677942/how-do-i-allow-commenting-in-youtube-through-google-if-i-have-third-party-coo` - last visited: 5. Mai 2015

[25] *Privater Modus - Kontrolle über die von Firefox gespeicherten Daten behalten*, `https://support.mozilla.org/de/kb/privater-modus` - last visited: 5. Mai 2015

[26] *The Design and Implementation of the Tor Browser*, `https://www.torproject.org/projects/torbrowser/design/` - last visited: 5. Mai 2015

Survey of Concepts for QoS improvements via SDN

Atanas Mirchev
Supervisor: Lukas Schwaighofer, Daniel Raumer
Seminar Future Internet SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: ga75lar@mytum.de

ABSTRACT

Considering the current state of the Internet, the available approaches to providing Quality of Service (QoS) for single services or specific tenants are limited and inflexible. Providers need a better solution that is scalable and that allows fine-grained tuning of network traffic. Recently, different SDN enabled QoS frameworks have emerged, offering a lot of possibilities for network reconfiguration and high level definition of policies. This paper gives a detailed summary of available QoS mechanisms that rely on SDN and compares their strengths and weaknesses. In the spirit of the survey, novel ideas for the future of QoS are discussed. SDN brings a lot of freedom and monitoring possibilities as a QoS domain, but the Internet has yet to adapt to this innovative concept: at the time of writing most of the existing solutions are prototypes.

Keywords

Software Defined Networking, Quality of Service, QoS via SDN, OpenFlow, OpenQoS, FlowQoS, QoS methods, SDN Approaches to QoS

1. INTRODUCTION

Today's networking consists of discrete sets of protocols that specify how hosts in different networks can be connected reliably. However, protocols tend to be defined in isolation and are designed to solve a specific problem. This results in high network complexity, which is a major limitation with regard to overall networking and providing Quality of Service (QoS). Due to this fact, today's networks are static. Typically all of the control decisions (e.g. how packets should be forwarded) are taken at the separate forwarding devices. This proves to be an obstacle considering the dynamic nature of QoS: IT administrators must configure each vendor's equipment separately, adjusting parameters (such as bandwidth) to meet the predefined rules and policies. This approach cannot dynamically adapt to the constantly changing application and user demands.

To tackle this problem, the concept of Software Defined Networking [1] has emerged. It focuses on decoupling the control plane from the forwarding plane (*Figure 1*), therefore leaving the existing routers and switches as simple forwarding devices. The control logic is instead centralized and deployed on a server (commodity hardware), called an SDN controller, which allows for easier network management and monitoring, while improving extensibility and scalability possibilities. The **Northbound API** (c.f. *Figure 1*) of an SDN controller is the public interface that other higher level applications can access. It conceals the actual implementation and is used to dynamically define abstract rules which are then enforced on the network by the controller. This also includes control policies and setting up traffic priorities that can be used for QoS. The **Southbound API** (c.f. *Figure 1*) is the interface between the controller and the forwarding plane. The standard protocol for communication over this API is called OpenFlow. It enables the SDN controllers to dynamically configure all forwarding devices and allows for more sophisticated traffic management. The configuration is done by defining rules and actions (called "flow table entries") in a switch's flow table. Those can then periodically or on demand be changed by the controller, according to the chosen policy. The protocol also specifies that new unmatched packets should be redirected through the controller (so that it can identify them and assign new rules to the forwarding elements).

"OpenFlow Configuration and Management Protocol" (OF-Config) and the "Open vSwitch Database Management Protocol" (OVSDB) act as specific extensions to OpenFlow and help the controller to configure the forwarding devices according to the defined rules.

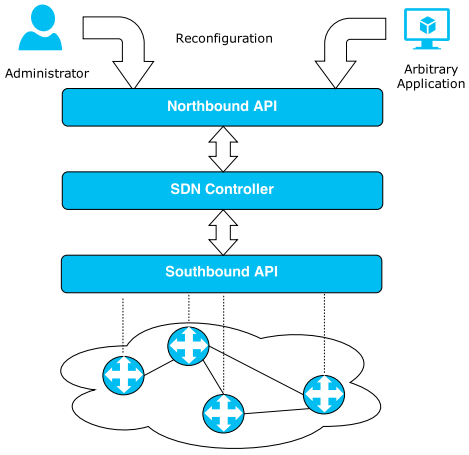

Figure 1: Simplified representation of the SDN architecture

In the context of this paper, an important part of networking is the overall performance of a connection, called Quality of Service. QoS comprises requirements on all major aspects of

doi: 10.2313/NET-2015-09-1_05

data transmission, such as response time, jitter, interrupts, etc. In order to provide QoS, application flows[1] need to be differentiated as they compete for bandwidth. Then network resources have to be allocated to ensure the precedence of the higher-priority traffic. That allows for the appropriate network resource distribution. This process often requires knowledge of the current network state, so that the right decisions with regard to packet forwarding can be made. That is why network monitoring is an important aspect of QoS. Ultimately, the goal is to also ensure network convergence, meaning that the network resources should be utilized as evenly as possible.

Without consideration of the available SDN methods, QoS relies on end-to-end agreements between hosts and Service-level Agreements (SLAs) between provider and user. While this approach is robust, it is best suited for best-effort services and doesn't allow for fine grained traffic engineering. Multimedia streaming and VoIP flows, on the other hand, require timely delivery over robustness and must be handled separately. The current "hop-by-hop decision" architecture of the Internet is sometimes difficult to monitor, mainly because of the many different vendor-specific firmwares at use [6]. There is no standardized way for specifying high level traffic control policies and restrictions with regard to the depth of traffic differentiation exist.

With the introduction of the SDN controller, a decoupled control plane is established. Besides translating the requirements of the application layer, it provides applications with an abstract view of the network. The network state is obtained for example via sampling of packets passing through the controller and can consist of different data, such as statistics or events. Using this information, control policies and SLAs can be specified by an administrator at a higher abstraction level and even dynamically adjusted. These mechanisms have proven to be beneficial for QoS. [2] [4] [8].

In the rest of the paper different techniques and possibilities for providing QoS via SDN are discussed. Chapter 2 presents viable approaches and methods in this area. Chapter 3 categorizes existing proposed solutions, while comparing their weak and strong points. In chapter 4, actual figures and results from the different frameworks are summarized and the overall viability of SDN as a QoS domain is assessed. In the last chapter, novel approaches and ideas for the future are proposed and concluding remarks are given.

2. APPROACHES TO PROVIDING QOS

There are two main categories of QoS techniques that need to be taken into consideration: **approaches proposed before the introduction of SDN** and **SDN enabled technologies**.

When considering **traditional QoS strategies**, without the involvement of SDN, two main types have been standardized. *Integrated services* [10] (short: IntServ) is a fine-grained, per flow traffic control architecture. The idea behind it is that every network element (router, switch) has to individually reserve resources for **each** flow. This is hard to accomplish in the current Internet [4]. First, routers have limited computational resources, that prohibits the classification of all possible application flows in the device itself. Second, this approach is not scalable, as every rooter on a flow's route needs to support Integrated Services and store all possible states and informations for the different flows. This is generally difficult to achieve (vendor lock-ins, limited memory resources). Therefore, this method is only applicable to small scale networks.

The second approach, *Differentiated services* [11] (short: DiffServ), is a coarse-grained traffic control architecture, relying on the 8-bit DS field (in place of the outdated TOS field) in the IP header. This field supports up to 64 different classes of traffic [15]. DiffServ routers then decide on per-hop basis how to forward packets based on their class. While this technique is applicable to bigger networks (since only a constant number of 64 classes need to be differentiated), it is static (because of the predefined number of classes) and lacks the ability to fine-tune the QoS of separate flows. For example, a use case with eight tenants having eight application types of traffic each easily reaches the limit of the DS field. This is not an unusual scenario for a single Autonomous System (AS). Furthermore, the specification of policies and the classification of traffic in DiffServ is done at the boundaries of DiffServ domains (Autonomous Systems) [4]. Because of this, there are no end-to-end guarantees across domains for the user, as the DS-classes can be interpreted differently in each one. To solve this, RFC 2638 [12] introduces a Bandwidth Broker for each domain, that has some knowledge on the policies in use. To ensure end-to-end QoS, bandwidth brokers need to communicate with each other across domains so that policies are interpreted properly. This is similar to the SDN enabled approaches, as it extracts the logic into a centralized agent. In fact, some of the proposed frameworks extend the idea of DiffServ by utilizing SDN [4]. However, provided the lack of centralized controllers in the current Internet architecture, DiffServ cannot be implemented globally, as there is no standardization of router reconfiguration protocols amongst different vendors. Each Broker relies on vendor-specific device reconfiguration, which imposes a restriction on the network.

In contrast to the described traditional methods, **SDN enabled approaches** tackle all of the problems described above. Through the higher level of abstraction provided by the decoupled controller, one can specify policies without the need to reconfigure low-level settings at each of the forwarding devices. The set of policies and also the different flow classes are unrestricted, allowing for fine-grained tuning based on the needs of the user (in contrast to the maximum of 64 predefined classes in the DS field). The rules can therefore be defined per flow (if necessary) and the controller has the task to apply them properly to the different network elements. In this regard, there are two main aspects to consider:

- providing QoS for a specific tenant or a business customer flow
- providing QoS for a specific application flow

A number of different SDN enabled solutions to those prob-

[1] In the sense of networking and SDN, a flow defines any given set of packets that share some common characteristics (e.g. all packets for a given HTTP connection)

doi: 10.2313/NET-2015-09-1_05

lems exist. The most common method, used by frameworks considered in this survey, is a form of virtual slicing of the available bandwidth. This can be classified as **resource reservation**, where each flow gets assigned a part of the overall transmission capacity. In contrast to that, **per-flow dynamic routing** has been proposed as a viable alternative that does not directly assign any of the resources to a flow. Moreover, approaches with specific focus on the actual **packet en-queuing** and **frameworks for policy enforcement** have been discussed as well. Many of the SDN enabled frameworks also require a form of sampling to constantly monitor the network [2] [8]. If all nodes are monitored, the gathered information can be extrapolated to the whole network [6], thus creating an overview of the network state. This approach creates overhead and needs to be applied in controlled manner.

In the following chapter, each of these aspects is covered in depth and example implementations are presented.

3. QOS ENABLED FRAMEWORKS

The following categories are the most prominent ways in which QoS can benefit from the concept of Software Defined Networking. For each category, first a short introduction to the approach is given. Then one or more frameworks are presented. Each framework's most notable features are explained and the results of available tests are discussed. It is important to notice that the described frameworks do not solely rely on the principles of the category they represent. The distribution among the different sections is based on the main focus of the specific solution.

3.1 Resource reservation frameworks

This is the most common solution for providing QoS, hence the large number of frameworks that fall into this category. Typically, frameworks of this type consist of two main modules: a flow classifier component and an SDN-based rate shaper. The classifiers read the packets' fields and attempt to assign a certain priority to the different flows based on the policies defined in the controller. The rate shapers, on the other hand, install resource reservation rules in an OpenFlow enabled switch based on this classification. The current OpenFlow implementations for the data plane elements (like Open vSwitch) don't explicitly support per-flow rate shaping, so most of the frameworks in this survey attempt to give some form of a universal solution for this problem.

An example for such a framework is **FlowQoS** [3]. This prototype targets small scale networks and utilizes SDN to reconfigure home routers according to a set of rate shaping policies defined by the user. The classification of the different flows is done by two separate modules. The first module performs initial classification of web traffic (HTTP and HTTPS), determined by the respective port numbers (80 and 443). Then further classification of this web traffic is done based on the DNS responses, matching the A and CNAME records against a list of predefined regular expressions. The second module handles the classification of other flows (non HTTP/HTTPS). They are classified based on the flow-tuple[2], the first four bytes sent and received and the sent and received payload sizes. After flows have been successfully classified, the rate shaping takes place. To tackle the lack of an OpenFlow per-flow rate shaping implementation, the framework introduces two "virtual" switches inside the home router. The different inner-switch connections between those two are then configured via Linux's **tc** utility and assigned different rates specified by the controller (predefined by the user). This is a simplified form of network slicing. For more information, please refer to [3]. This is a limited approach, as it only shows how QoS can be applied to a home router. Moreover, there is no data on executed tests and results, compared to the following frameworks. FlowQoS is a small, proof of concept framework that shows how SDN can be used to enable QoS and at the same time simplifies QoS configuration for single users. It can replace the need of manually using Linux's tc utility or iptables tagging, which could be complicated. The authors don't state any further use cases in their paper.

Moving from small to large-scale solutions, a more general framework is a part of the **EuQoS project** [4]. The focus of this solution is to enable QoS for *business customer* flows and make it possible to prioritize them on demand. The classification in this approach is based on the value of the DS field in combination with the destination IP of a packet, as only different tenant flows need to be distinguished. This is very similar to the standard approach applied in DiffServ, but also differentiates the tenants based on their IP address.

All routers are reconfigured by the controller. For each regular routing entry in a router two OpenFlow flow table entries[3] are created, which match the same destination IP as the routing entry. One of them matches packets with TOS disabled (low priority) and the other one matches packets with TOS enabled (high priority). By doing this, best-effort and high-priority business traffic can be differentiated. Therefore the number of flow entries is twice the number of regular routing entries. From a scalability perspective, this is acceptable.

For rate shaping, two queues with transmission rates are configured at each router. One matches the TOS enabled packages, the other one handles the rest. This makes the network suitable for differentiation of flows coming from different hosts. However, it does not cover the second type of differentiation - flows of different applications.

In addition to the regular resource reservation, EuQoS focuses on two more aspects:

- **Inter-AS QoS** - For each Autonomous System, one SDN controller is operational. It creates a flow entry at the edge router that sets the TOS field to enabled for a given high-priority flow coming from another AS. Since the only modification of packet fields happens at the border of an AS and the forwarding decisions are based on the TOS value, regular DiffServ forwarding can also take place inside the AS. That is why standard BGP (Border Gateway Protocol) or OSPF (Open Shortest

[2]source IP, source port, destination IP, destination port, transport layer

[3]A flow table entry matches a certain flow (based on predefined rules) and defines an action to be applied to that flow (e.g. forward, drop, change field)

doi: 10.2313/NET-2015-09-1_05

Path) routing can be used at this point. This provides a dynamic way to give priority to certain tenant flows across multiple AS, for example when an SLA between a business customer and a provider is taking place. EuQoS further allows the existing DiffServ brokers to utilize the SDN Northbound API.

- **Failure recovery** - each time a connection fails, the framework automatically ensures that the high-priority traffic does not suffer and is restored to the predefined transmission rate. This also means that if not enough network bandwidth is available (because of the link failure), some of the best-effort packets will not be forwarded immediately.

The framework was tested on the OFELIA [4] testbed, with 100 nodes for single and multiple AS. The results show that the flows with higher priority do take precedence, even in cases where the bandwidth is not enough for all the traffic [4]. The extend of this test exceeds that of any of the frameworks covered in this survey, but EuQoS remains a prototype.

Another SDN framework that utilizes similar rate shaping and classification approaches was presented at **Princeton University** [9]. The main difference is that it proposes an adaptive flow aggregator for the case when QoS needs to be provided per application flow. Services with similar needs are bundled together and handled as one flow in the forwarding devices, which improves scalability [9]. This way, the solution can handle **both** application flows and customer flows. The Princeton framework has a second main focus, which is the *convergence* of the whole network. For that purpose, active sampling of the network state is required. This introduces overhead, bound to put some strain on the network, which is virtually unavoidable with the given goal. The authors don't quantify how much stress is put on the controller, but this aspect should be taken into consideration.

The framework has been tested in a real world setting with 3 OpenFlow switches and 4 hosts, which is a rather small testbed. It shows better utilization of network resources due to the described monitoring. This framework, similarly to the ones preceding it, remains only a prototype.

3.2 Per-flow Routing Frameworks

This concept distinguishes multimedia flows from regular data flows via a classifier, similar to the resource reservation technique. However, instead of reserving resources (in the sense of bandwidth slices) at each forwarding device, the controller dynamically places the high priority flows on QoS guaranteed routes [2]. This enables dynamic routing, for which SDN is highly advantageous. At the same time, regular data flows remain on their usual routes. In this context, QoS routing is viewed as a Constraint Shortest Path (CSP) problem [2]. For example, one possible constraint is the delay for a multimedia stream. Since this problem is NP-hard, a heuristic is proposed in the OpenQoS framework. To ensure that the path is really optimal, **congestion** of the network is also taken into account. OpenQoS considers a link to be congested, if its utilization exceeds 70% of the possible bandwidth [2].

The main advantage of this approach to resource reservation (as in section 3.1) is minimized latency and packet loss for the non-QoS flows [2].

A representative framework of this type is OpenQoS. It guarantees service (application traffic) delivery on an optimal path and focuses mainly on multimedia flows, such as VoIP or video streaming. The framework itself is based on Floodlight, which is a universal Java controller.

For the generation of effective routes for the high-priority service flows, the controller needs an overview of the current network state (expressed in packet loss, delay, etc. for each link). The network state is collected from the simple forwarding elements by actively sending FEATURE_REQUEST messages for each feature of interest (doing active sampling). The performance of the framework depends on the accuracy of the gathered data, therefore the framework queries the elements each second [2]. This requirement has a negative effect on scalability. Gathering precise data for large networks is inefficient. The computation of the most efficient CSP per flow introduces further overhead for the controller. However, the paper of QoSFlow does not quantify the strain on the SDN controller that the monitoring causes. This aspect of the framework needs to be further investigated.

Another advantage of OpenQoS is its attempt to define flows only using fields from lower OSI levels, as "the packet parsing complexity is lower compared to [..] upper layers", according to [2]. That is why OpenQoS differentiates multimedia flows based on fields in MPLS (Multiprotocol Label Switching), which is generally placed between the data link and network OSI layers.

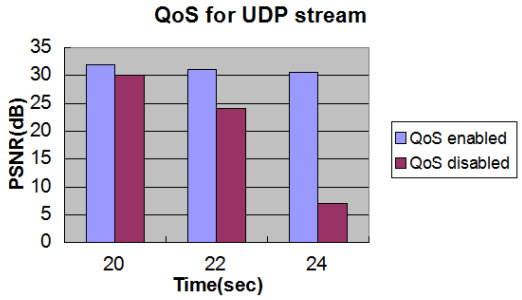

Figure 2: Three samples of PSNR in time, representing the results of an OpenQoS test [2]

The framework has been tested on a network with 3 Pronto 3290 switches, which is not sufficient to overrule the scalability concerns, and still remains a prototype. One of the executed tests compares the performance of two video UDP streams of the same content, with and without enabled QoS. In the period from second 20 to second 24 of the transmission, there is a significant difference in the PSNR (Peak Signal-to-Noise Ratio) of the two streams. The metric quantifies the quality of the received signal by measuring the power of the corrupting noise (loss) during transmission. Accordingly, higher values mean a better service quality. As

Figure 2 demonstrates, the UDP stream with enabled QoS is preserved better during the specified time frame. This in turn shows the effectiveness of the framework when services need to be prioritized.

3.3 Frameworks with Focus on Queue Management and Package Scheduling

The topic of packet scheduling and enqueuing is also important. OpenFlow version 1.3 specifies a standard FIFO queuing of all incoming packets [13]. This could prevent certain flows (e.g. multimedia streams) from meeting the QoS requirements if the network segment is congested. To overcome this limitation, a framework called **QoSFlow** is proposed, which enables the reordering of packets in a given queue. It attempts to introduce the traffic control capabilities of Linux into OpenFlow networks. In this context, the framework implements a couple of different packet scheduling mechanisms [13], the most notable of which is SFQ (Stochastic Fairness Queuing). This algorithms aims at giving each high-priority flow a fair chance to be forwarded, contrasting to standard FIFO queuing. Each flow is assigned to a bucket. Afterwards, in a round-robin manner, flow packets are popped one by one from the respective buckets in an effort to ensure fairness.

However, there are some restricting factors as well. The framework is limited to 8 queues per switch port. This limitation exists because of the used slicing mechanism specified in OpenFlow 1.0. While it is not caused by the framework itself, this fact still needs to be taken into account. Another disadvantage is that it requires that the switches run a Linux distribution instead of a vendor-specific firmware. Therefore this solution is not directly applicable to the Internet in its current state.

QoSFlow has been tested with up to 3 TPLink 1043ND commercial switches [13], which is a small test network. In one of the tests on the performance of the SFQ algorithm, the QoE (Quality of Experience) is again quantified by the Peak Signal-to-Noise Ratio (PSNR), similarly to the OpenQoS case. For two different video flows the behavior of SFQ and standard FIFO scheduling is compared. SFQ reaches a higher PSNR value (which is better) in both cases. According to the QoSFlow paper, the difference between the two approaches is 48.57 % in the first case and 68.57 % in the second case, in favor of SFQ against FIFO [13], meaning that the improved scheduling does lead to better video quality, as far as the user perception is concerned.

3.4 Frameworks for Policy Enforcement

The major problem with predefined SLAs, in regard to QoS, is that with the current Internet architecture there is not a single standardized and flexible way to apply those to the network. Most of the newest technologies that are applied to achieve those SLAs are also proprietary. Therefore the need for flexible and scalable management tasks emerges.

Via the Northbound API of an SDN controller, it is possible to dynamically define SLAs and then enforce them on the underlying forwarding plane through the southbound API. One of the proposed SDN-enabled solutions is PolicyCop, based on the Floodlight controller. It differs from other existing DiffServ approaches [8] as it is not restricted by static traffic classes and the coarse-grained granularity that comes with them. The framework is easily extensible, as it has a layered architecture. Each layer communicates with the others via RESTful JSON APIs, hence the possibility to easily add new layers or exchange old ones. It further allows to dynamically create new flow classes, in contrast to the static classes in DiffServ [8]. The solution is flexible, as it does not require any special resources, other than OpenFlow-enabled switches.

The policy management itself is done by 2 components: a Policy Validator and a Policy Enforcer. The Policy Validator monitors the network to ensure that policies are being applied properly. To make this possible, information per flow and per flow table needs to be actively gathered by the controller. This is achieved by inserting packet probes in the network.[8] Meanwhile, passive statistics are also gathered from packets redirected to the controller. This results in the monitoring of metrics describing the network state, like bandwidth usage, residual capacity, number of dropped packets, latency, error-rate and jitter [8]. For more information on the monitoring framework that PolicyCop uses, please refer to [16]. Similar to OpenQoS, this could lead to a serious overhead and puts the scalability of the framework in question. As a result of the inquired network state, the Policy Enforcer places new rules in the forwarding elements, to account for possible changes or deviations from the defined policies.

The network has been tested in an environment of 5 Open vSwitches and 4 hosts. PolicyCop successfully manages to restore broken policies and the corresponding flows' throughput in a scenario with 4 different services running simultaneously [8]. However, the provided testbed is too small and can only be seen as a "Proof of Concept".

3.5 Detailed comparison

From all presented frameworks, only **EuQoS** has been tested for a larger scale network. Although the tests of all other frameworks manage to show their capabilities, they can be considered insufficient because of the small test-bed sizes. In contrast to the others, **EuQoS** also directly supports standard DiffServ approaches. Nevertheless, that restricts the differentiation of flows to single hosts (high priority business customers), differentiation based on services and applications is not possible. The framework proposed by **Princeton** handles this problem well, aggregating similar services to a single flow in addition to the per-tenant flow classification. However, this system has been tested in a much smaller testbed than the EuQoS framework. Different from those two more general approaches, **OpenQoS** focuses on multimedia traffic only, using per-flow routing instead of resource reservation. However, while this technique can arguably be more efficient path-wise, it introduces an overhead due to active sampling of the system state, which is not necessarily required in a resource reservation framework. **QoSFlow**, on the other hand, is an example of a support framework that defines additional improvements of QoS via SDN methods, since it is concerned only with the actual order of the queues during the packet scheduling. Further investigation of the possible integration of this framework with the others is encouraged. The major problem in this case is that

all forwarding devices are required to run Linux, which is currently not realistic in a normal network setting. Lastly, **PolicyCop** focuses on the actual policy management, giving administrators full control to efficiently change the way traffic is handled. All of the covered frameworks are prototype solutions.

A detailed summary of all the relevant information can be found in *Apendix 1.*

4. VIABILITY OF SDN AS A QOS DOMAIN

Considering the data from the different framework tests, it is clear that there are advantages when using SDN controllers to provide QoS.

The **EuQoS** framework shows that it can transmit all high priority traffic without any loss in a medium data rate scenario (2.4 Mb/s to 7 Mb/s traffic per host) with a 30% to 70% ratio of high priority to best-effort traffic in the network [4]. **QoSFlow** suggests a possible gain in QoE of up to 48.7% when implemented properly [13]. As seen in section 3.4, **PolicyCop** can maintain and enforce policies once again when they have been broken.

However, the results of those frameworks (with the exception of EuQoS) were obtained in small networks, consisting of not more than 5 forwarding devices. Therefore, more thorough investigation is needed.

The configuration of traffic queues is another important aspect of QoS. Protocols such as OF-Config and OVSDB deal with the issue of configuration of resources (like queues), but are not properly supported by the mainstream OF controllers. A standalone solution for this problem called **QueuePusher** has been proposed, as an extension to the controller Floodlight. It is based on the OVSDB protocol and describes the mechanisms of building priority queues for the different QoS enabled flows [7].

This is an example of how some important aspects of SDN are still not available, despite their specification in the OpenFlow protocol.

Overall, the fact that a lot of the mentioned frameworks need to extend a controller or implement a missing protocol feature, combined with all of them being prototypes, shows that the concept of SDN still needs to be better accommodated. Nevertheless, all of the frameworks show significant gains in different aspects of QoS which cannot be neglected. While it is true that SDN introduces a single point of error [6] via the centralization of the controller, distributed solutions have been proposed [4], mitigating this concern.

5. RELATED WORKS

A comprehensive survey of the different aspects of SDN has already been conducted [1]. The work covers all major aspects of the SDN domain, also specifying a list of traffic engineering solutions with regard to QoS. However, a comparison or categorization of the SDN enabled QoS frameworks is not present. Also, no advantages and disadvantages are discussed and no specific implementation details or test figures are covered.

6. FUTURE OUTLOOK. CONCLUSION

The proposed QoS frameworks all benefit from the idea behind SDN. Since they are still prototypes, most of them are continuously developed further. **OpenQoS** proposes a future application of "per-flow routing" in the area of Multiple Description Coding (MDC). A source stream is encoded into independent bitstreams that can be decoded separately. There is a requirement that every bit stream should take a different path. This is achievable via the dynamic routing possibilities of OpenQoS.

Regarding the whole Internet, the Control Exchange Points (CXP) [14] has been proposed as an architectural model that is supposed to improve end-to-end QoS across different domains. This suggests a new type of business relationship between ISPs, where each provider defines a "partial path" for a flow and the CXP component combines all of them together.

In this paper different types of SDN enabled QoS solutions have been evaluated. While small scale frameworks are beneficial to the individual user, larger scale implementations represent ideas applicable to the whole Internet architecture. Despite the fact that Software Defined Networking is still a relatively new approach, the detailed categorization and comparison have revealed important positive effects in regard to QoS , achievable only through a decoupled and easily manageable control plane.

7. REFERENCES

[1] Diego Kreutz, Fernando Ramos, Christian Rothenberg, Siamak Azodolmolky, Steve Uhlig: *Software Defined Networking: A Comprehensive Survey*, page 25, Proceedings of the IEEE, Volume 103, Jan, 2015

[2] Hilmi Egilmez, S. Dane, K. Bagci, A. Tekalp: *OpenQoS: OpenFlow controller design for multimedia delivery with end-to- end Quality of Service over Software-Defined Networks*, APSIPA ASC, Hollywood, CA, 3-6 Dec, 2012

[3] M. Seddiki, Muhammad Shahbaz: *FlowQoS: QoS for the Rest of Us*, HotSDN?14, Chicago, 22 Aug, 2014

[4] S. Sharma, D. Staessens, D. Colle, D. Palma: *Implementing Quality of Service for the Software Defined Networking Enabled Future Internet*, EWSDN, Budapest, 1-3 Sep, 2014

[5] G. Araniti, J. Cosmas, A. Iera, A. Molinaro: *OpenFlow over Wireless Networks: Perfomance Analysis* BMSB, Beijing, 25-27 Jun, 2014

[6] Daniel Raumer, Lukas Schaighofer, Georg Carle: *MonSamp: A distributed SDN application for QoS monitoring* FedCSIS, Warsaw, 7-10 Sep, 2014

[7] David Palma et al., *The QueuePusher: Enabling Queue Management in OpenFlow*, EWSDN, Budapest, 1-3 Sep, 2014

[8] M. F. Bari et al., *PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks*, SDN4FNS, Trento, 11-13 Nov, 2013

[9] Wonho Kim et al. *Automated and Scalable QoS Control for Network Convergence*, INM/WREN, San Jose, CA, 27 Apr, 2010

[10] R. Braden et al., *Integrated Services in the Internet*

Architecture: An Overview, Internet Engineering Task Force, RFC 1633, Jun, 1994

[11] S. Blake et al., *An Architecture for Differentiated Services*, Internet Engineering Task Force, RFC 2475, Dec, 1998

[12] K. Nicolas et al., *A Two-bit Differentiated Services Architecture for the Internet*, Internet Engineering Task Force, RFC 2638, Jul, 1999

[13] A. Ishimori et al. *Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking*, EWSDN, Berlin, 10-11 Oct, 2013

[14] V. Kotronis et al. *Control Exchange Points: Providing QoS-enabled End-to-End Services via SDN-based Inter-domain Routing Orchestration*, ONS, Santa Clara, CA, 3-5 Mar, 2014

[15] K. Nichols et al. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, Network Working Group, RFC 2474, Dec, 1998

[16] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba *PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks*, NOMS, Krakow, 5-9 May, 2014

doi: 10.2313/NET-2015-09-1_05

	Framework	Scale	Type	Testbed	Flexibility	Guarantees
Resource reservation	FlowQoS	Small scale (home routers)	Prototype	-	Restricted to home routers	-
	EuQoS	Medium to large-scale (possible across multiple AS)	Prototype	OFELIA testbed, 100 nodes	Compatible with BGP and OSFP and therefore to the current Internet	Flow of prioritized tenant has precedence, failure recovery
	Princeton framework	-	Prototype	3 ProCurve 5406zl switches	-	Network wide optimization & convergence
Per-flow path optimization	OpenQoS	Medium scale (supposedly, since the network needs to be monitored)	Prototype	3 Pronto 3290 switches	Scalability problems	Multimedia traffic with enabled QoS performes better
Queuing and scheduling	QueryPusher	Any	Prototype	-	Floodlight extension	-
	QoSFlow	Any	Prototype	Up to three TPLink 1043ND	Requires Linux installed at each router; limited to 8 queues per switch port	Improvement in QoE for QoS enabled traffic (up to 48 %)
Policy Enforcement	PolicyCop	Medium scale (supposedly, since the network needs to be monitored)	Prototype	5 switches running Open vSwitch	Compatible with all switches with OpenFlow support, easily extensible due its layered architecture	Enforcement of policies

Table 1: Apendix 1

doi: 10.2313/NET-2015-09-1_05

Gaming in the Cloud: a Survey

Sebastian Neubauer
Betreuer: Daniel G. Raumer, Paul Emmerich
Seminar Future Internet SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: sebastian.neubauer@tum.de

KURZFASSUNG

In der heutigen Zeit entwickelt sich die Hardware für die Unterhaltungsbrache in einem rasanten Tempo stetig weiter. Konsequenz dieser Entwicklungen ist eine kontinuierlich steigende Erwartungshaltung von Nutzern gegenüber Spielen, ebenso wie stets ansteigende Anforderungen, die digitale Spiele an ihre Nutzer und Hardware stellen. Ununterbrochene Entwicklungen und die damit verbundenen Neuanschaffungen sind kostenintensiv. Hier stellt sich die Frage nach Möglichkeiten der Kostenspirale zu entkommen ohne auf den Komfort aktueller Spiele verzichten zu müssen. Eine mögliche Alternative zu konventionellem Gaming bildet das Konzept des Cloud Gamings. Verschiedenste Anbieter haben das Potenzial bereits erkannt und haben sich mit verschiedene Cloud Gaming Diensten auf dem Markt positioniert. Trotz der wachsenden Zahl an Angeboten und den Vorteilen, die Cloud Gaming bietet, bildet dieses Konzept bisher nur ein Nischenprodukt und konnte sich nur unter bestimmten Voraussetzungen als echte Alternative positionieren. Die folgende Arbeit zeigt diese Voraussetzungen und gewährt einen komprimierten Einblick in das Konzept des Cloud Gamings.

Schlüsselworte

Cloud Gaming, cloudgaming, gaming in the cloud, Gaming Anywhere, OnLive

1. EINLEITUNG

Gaming erfreut sich einer hohen Popularität. Mögliche Gründe hierfür sind vielfältiger Natur, Fakt ist, dass mehr als 34 Mio. Menschen in Deutschland regelmäßig digitale Spiele spielen. Dies entspricht knapp der Hälfte aller Deutschen. Der Umsatz für Spiele, Hardware, Zusatzangebote und virtuelle Güter verzeichnete im Jahr 2014 einen Umsatz von rund 2,6 Mrd. Euro [5]. Im Jahr 2013 verzeichnete der amerikanische Markt für digitale Spiele einen Umsatz 21,53 Mrd Dollar, hier konsumieren mehr als 56% regelmäßig digitale Spiele [12]. Weltweit prognostiziert das Beratungsunternehmen Garter einen Gesamtmarkt von 111 Milliarden Dollar [11].

Diese Zahlen zeigen das Potential dieser Branche. Der Markt unterliegt einem rasanten, stetigen Wandeln: Spiele und Hardware entwickeln sich ununterbrochen weiter. Neue leistungsstarke Hardware ermöglicht den Spieleentwicklern eine detailreiche Gestaltung der Spiele. Dies bedeutet unter anderem eine größere Weltenvielfalt für den User. Spielwelten werden größer, detailreicher, komplexer und realer. Wichtige Anteile in diesem Prozess haben die Hardware selbst und die Spieleentwicklung. Sie stehen in einer gegenseitigen Erwartungshaltung. Die Hardware fordert auf der einen Seite die Entwicklung der Spiele. Auf der anderen Seite forciert die Spieleentwicklung die Hardwareentwicklung, da aufgrund verbesserter Technologien und den dadurch erreichbaren Möglichkeiten die Erwartungshaltung an neue Spielen steigt.

Die stetige Verbesserung der Hardware und die wachsenden Anforderungen, die neu veröffentlichte Spiele an die Hardware stellen, erfordern ein regelmäßiges Aufrüsten der eigenen Hardware, um Spiele in bestmöglicher Qualität zu spielen. Der benötigte Speicherplatz erreicht bei aktuellen Spielen mehrere Gigabyte, dazu kommen große Arbeitsspeicher, leistungsfähige Prozessoren und starke Grafikkarten. Für das kürzlich erschienene Battlefield Hardline werden 60 Gigabyte Festplattenspeicher, 8 Gigabyte Arbeitsspeicher, Ein Quad- oder Sixcore Prozessor und 3 Gigabyte Grafikspeicher empfohlen [7].

Ein Vielzahl der Nutzer bezieht die Software mittlerweile als online Download [5]. Darüber hinaus erfordern viele Spiele bereits eine permanente Internetverbindung oder bieten nur noch einen online Multiplayer. Cloud-basierte Lösungen setzen hier an. Unter Cloud Diensten versteht man die Auslagerung der eigenen Daten, einzelner Berechnungen bis hin zu kompletter Software in externe Rechenzentren. Hier haben sich Technologien wie Software as a Service und Platform as a Service etabliert. Aus diesen Cloud-basierten Umsetzungen ist in den vergangen Jahren das Cloud Gaming entstanden. Die folgende Arbeit erklärt das Cloud Gaming Prinzip und geht auf bereits existierende Umsetzungen und Herausforderungen ein. Es werden verschiedene Cloud Gaming Anbieter und eine Open Source Lösung vorgestellt. Dazu wird ein Vergleich zwischen Cloud Gaming und dem konventionellem Gaming gezogen.

2. DAS CLOUDGAMING PRINZIP

Die Idee des Cloud Gaming basiert auf dem Software as a Service Prinzip. Hierbei werden einzelne Softwarekomponenten ausgelagert und auf externen Servern betrieben. Verantwortlich für die entsprechende Soft- und Hardware ist der Betreiber der Server der den Zugriff mittels standardisierter Schnittstellen ermöglicht [14, S.34]. Cloud Gaming stellt eine in erster Linie einfache Art des Software as a Service Prinzip dar. Die Game-Server hosten die verfügbaren Spiele und senden den Audio und Video Stream an den Client. Hierbei kann zwischen zwei möglichen Ansätzen unterschieden

doi: 10.2313/NET-2015-09-1_06

werden [3]. Beim Image-Based Cloudgaming wird das Spiel als live Videostream auf den Client übertragen, sämtliche Berechnungs- und Renderingvorgänge werden vom Server ausgeführt.

Das Instruction-Based Cloudgaming Verfahren lässt den Server grafische Informationen übertragen, welche durch den Client verarbeitet und gerendert werden müssen (vgl. Abschnitt 3.2 Gaming Anywhere). Von der Seite des Clients, werden in beiden Verfahren sämtliche Benutzereingaben (Maus, Tastatur, Touch) an den Server gesendet, dieser verarbeitet die neuen Informationen und setzt diese im Spiel um. Durch die Auslagerung auf externe Server, ist je nach genutztem Verfahren die nötige Leistung der Clients sehr gering. Dies ermöglicht es mit verschiedenen Endgeräten von beliebigen Standorten auf Spiele und Speicherstände zuzugreifen. Anbieter wie Onlive, StreamMyGame und GaiKai verwenden das Image-Based Cloud Gaming [3].

Beim Image-Based Cloud Gaming ist aufgrund der angesprochenen Auslagerung der Installation und Berechnung des Spiels und der damit einhergehenden niedrigen Anforderungen an die lokale Hardware ein Thin Client, der in der Lage ist, die empfangenen Informationen zu decodieren und auf einem Bildschirm darzustellen ausreichend. Neben PCs kommen so ebenfalls Notebooks, Tablets, Smartphones sowie TV-Geräte als mögliche Clients in Frage.

Cloud Gaming unterscheidet sich hier von Games on Demand und Browserspielen. Ein Großteil der Browserspiele sind Client seitige Browser-spiele welche eine Installation im Hintergrund ausführen und die Berechnung des Spiels lokal erfolgen lässt. Auch Games on Demand Anbieter stellen lediglich die Installationsdateien auf Abfrage online bereit. Eine lokale Installation ist weiterhin erforderlich.

2.1 Voraussetzung der Anwender

Wie in Abschnitt 2 bereits erwähnt, sind die Hardwarevoraussetzungen für Clients im Cloud Gaming gering. Aufwendige Installationen auf die lokal Festplatte, sowie leistungsstarke Grafikkarten, Prozessoren oder große Arbeitsspeicher werden überflüssig [23]. Dies ermöglicht es Spiele mit hohen Hardwareanforderungen auf Geräten zu spielen die bereits technisch überholt oder gar nicht erst für diese Art von Anwendungen konzipiert wurden. Verschiedene Anbieter bieten Set Top Boxen an, die das Cloud Gaming auf TV Geräten ermöglicht. Zusätzlich rücken Tablets und Smartphones in den Fokus der Spieler. Diese mobilen Geräte sind durch ihre Systemarchitektur nicht für aufwendige Spiele ausgelegt, die für leistungsstarke PCs konzipiert wurden. Entscheidende Unterschiede sind beispielsweise die Architektur des Prozessors und fehlende oder zu schwache grafische Prozessoren. Durch die geringen Größen mobiler Geräte spielt auch die thermische Belastbarkeit und Akkukapazität eine Rolle.

Um als Cloud Gaming Client in frage zu kommen, muss die eingesetzte Hardware muss im Grunde nur in der Lage sein, den eingehenden Audio- und Videostream in Echtzeit zu decodieren und darzustellen. Hierzu wird je nach verwendetem Endgerät noch ein Bildschirm benötigt. Darüber hinaus ist die Qualität und Geschwindigkeit der Breitbandverbindung entscheidend. Die von Cloud Gaming Anbietern empfohlene Breitbandverbindung beträgt mindestens 3 Mbps [21, 23]. In verschiedenen Studien [3, 9] wird dies bestätigt und gezeigt, dass die Breitbandverbindung über einen stabilen Downlink von 3-5 Mbps verfügen muss, um ein flüssiges Spielerlebnis zu garantieren. Verschieden Anbieter bieten dem User die Möglichkeit eine eigene Cloud Gaming Umgebung aufzubauen. Je nach Anwendungsfall ist eine Breitbandverbindung nicht zwingend erforderlich. Das eigene Netzwerk reicht beispielsweise aus um innerhalb seiner Wohnung über das TV Gerät oder vom Tablet aus auf die eigenen Spiele zuzugreifen.

2.2 Infrastruktur der Anbieter

Genaue Informationen zu der Infrastruktur der Anbieter waren im Rahmen dieser Arbeit nicht zugänglich. Wie Cloud Gaming umgesetzt werden kann wird im folgend mit Unterstützung verschiedener Beispiele beschrieben.

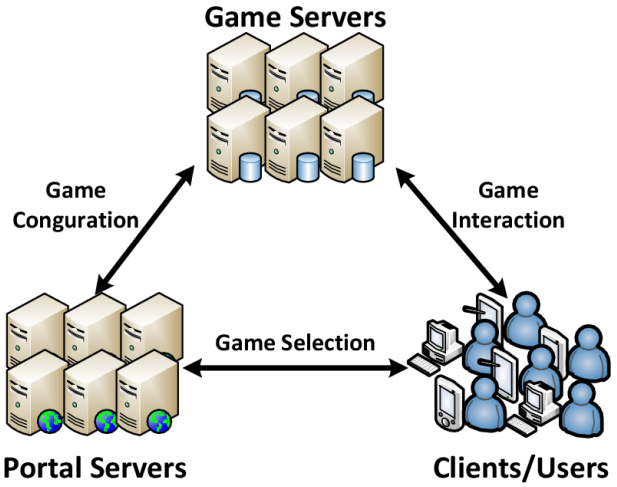

Abbildung 1: Infrastruktur eines Cloud Gaming Dienst (Abbildung aus [9]).

Der Anbieter stellt dem Nutzer mittels einfachem Portal-Server den Zugang zur Spielbibliothek bereit. Über das Bereitgestellte Portal, startet der Nutzer das gewünschte Spiel. Der Portal-Server dient hierbei zur Vermittlung zwischen Spieler und Game-Server. Ist diese Verbindung einmal aufgebaut erfolgt die weitere Kommunikation direkt zwischen dem Client und dem Game-Server (siehe Abbildung 1). Ausgeführt wird das Spiel auf einem Game-Server. Um mehreren Spielern zeitgleich einen Zugang zu ermöglichen stehen mittels Virtualisierung mehrere Instanzen zur selben Zeit zu Verfügung. NVIDIA bietet mit der GRID Technologie die Möglichkeit bis zu 24 Instanzen simultan auszuführen [15].

OnLive verwendet der Untersuchung [6] zufolge pro Spieler einen einzelnen Server. Jeder Server in den OnLive Rechenzentren verfügt über einen Teil der angebotenen Spiele. Die einzelnen Spiele werden nach ihrer Popularität gewichtet und mittels einem "worst-fit binpacking" Algorithmus auf den Servern verteilt. Um die Gesamtverzögerung zwischen Nutzereingabe und Spielreaktion möglichst gering zu halten, spielt neben der Hardware der Server auch die Verzögerung durch das Senden der Daten über das Netzwerk bzw. das Internet eine große Rolle [3]. Eine zu große Entfernung

doi: 10.2313/NET-2015-09-1_06

zum Rechenzentrum erhöht diese Verzögerung. Aus diesem Grund sollte stets das geographisch nächstgelegen Rechenzentrum gewählt werden.

2.3 Umsetzung

Gängigen Cloud Gaming Lösungen basieren auf dem Image-Based Prinzip (vgl. Abschnitt 2). Dies bedeutet, dass sämtliche Rechenoperationen auf dem Server stattfinden und der Client lediglich zur Eingabe der Nutzerinteraktionen und zur Ausgabe des Audio und Video Signals dient. Realisiert wird dies, indem auf Seiten des Servers so in das Spiel eingegriffen wird, dass es möglich ist, die Audio und Video Erfassung zu codieren und an den Client zu senden. Eine gängige Variante ist der Einsatz des Standard Video Codecs H.264 [26] der die vom Spiel erzeugten Bilder komprimiert und so das Streaming per Internet oder Netzwerk ermöglicht. Der H.264 Codec zeichnet sich durch eine hohe Kompressionsrate und gute Konfigurationsmöglichkeiten für Echtzeit Anforderungen aus [20]. Auf der anderen Seite müssen alle Nutzereingaben, welche Client-seitig getätigt werden an das Spiel übertragen und vom Server so in das Spiel eingebunden werden, als ob die Eingaben direkt im Spiel erfolgt. In Unterpunkt 3.2 wird anhand des Gaming Anywhere Beispiels eine mögliche Variante, wie sowohl Server- als auch Client-Seitig in das Spiel eingegriffen werden kann näher erläutert.

2.4 Herausforderungen

Die größte Herausforderung des Cloud Gamings ist es, dem Spieler das selbe Spielerlebnis zu bieten wie wenn das Spiel lokal installiert wäre. Hierzu gibt es verschiedene Studien [9, 3, 2, 1, 8] die die Qualität des Cloud Gamings gemessen und ausgewertet haben. Entscheidend für ein flüssiges Spielgefühl ist die Zeit von der Interaktion des Spielers bis zur Reaktion im Spiel (Latenz). Diese hängt bei lokal ausgeführten Spielen von der Leistung der Hard- und Software ab ab. Da beim Cloud Gaming das Spiel jedoch auf einem externen Server ausgeführt wird spielen bei der Entstehung der Latenz weitere Faktoren eine Rolle. Hierzu wurden für die Ausarbeitung in [3] verschiedene Messungen durchgeführt und die Zeit der Verzögerung als Response Delay (RD) benannt. Dieser Wert ist definiert als Summe der Zeit, die zur Übertragung der Signale vom Client zum Server und zurück benötigt wird (Network Delay (ND)) und der Zeit die der Server benötigt die empfangen Informationen an das Spiel weiter zu leiten und die aktualisierten Audio und Video packte zu codieren (Processing Delay (PD)). Dazu kommen die Zeiten, die das Spiel benötigt die Nutzereingaben umzusetzen und das Spiel zu rendern (Game delay (GD)), sowie die Zeit die vom Client benötigt wird die vom Server gesendeten Informationen zu empfangen, decodieren und auf dem Bildschirm auszugeben (Playout delay (OD)).

Somit gilt:

$$RD = ND + PD + GD + OD$$

Je nach Spielart, haben Spieler unterschiedliche Anforderungen First Person Shooter beispielsweise erfordern eine Latenz die 100 Millisekunden nicht überschreiten sollte [2]. Eine größere Verzögerung könnte bereits Nachteile für den Spieler mit sich bringen. Rollenspiele oder Echtzeit Strategiespiele dagegen sind in der Umsetzung der Eingaben des Nutzer

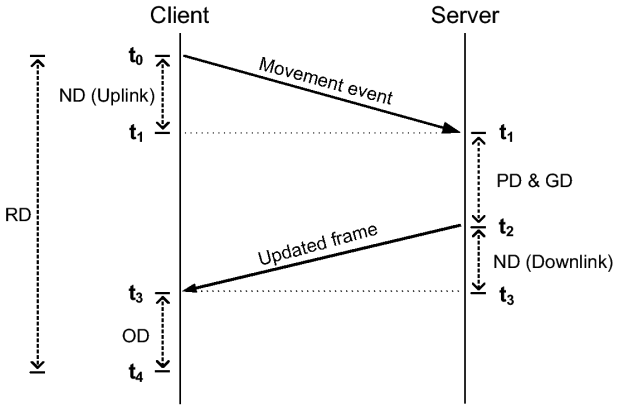

Abbildung 2: Darstellung des Response Delay (Abbildung aus [3])

deutlich flexibler. Die Ausführung [20] zeigt, dass diese Genres eine Latenz von bis zu 500 ms beziehungsweise 1000 ms zulassen. Dies begründet sich in der geringeren Spielgeschwindigkeit und Umsetzung von Aktionen innerhalb des Spiels. Die Untersuchungen in [1] zeigen jedoch, dass Spieler eine geringe Paketverlustrate von 0,1% und eine Latenz von 200 ms einer Latenz von 100 ms bei 1% Paketverlustrate vorziehen. Dies macht deutlich, dass neben den Verzögerung der Eingaben auch Paketverluste und die Qualität und Geschwindigkeit der Breitbandverbindung eine entscheidende Rolle spielen.

3. CLOUD GAMING ANWENDUNGEN

verschiedene Unternehmen haben das Potential des Cloud Gamings erkannt und bieten verschiedene Produkte am Markt an. Einen tieferen Einblick in die verwendeten Techniken der kommerziellen Anbieter zu erhalten ist oftmals nicht möglich. Aus diesem Grund können technische Hintergründe nur teilweise erläutert werden. In den folgen Abschnitten wird auf verschiedene kommerzielle und ein Open Source Anbieter eingegangen.

3.1 Kommerzielle Anbieter

Onlive. Der im Jahr 2010 in den USA gestartete Cloud Gaming Dienst OnLive musste auf Grund der Übernahme von wichtigen teilen des Unternehmens durch Sony seinen Dienst zum 30.04.2015 einstellen [25]. OnLive bot, mit mehreren monatlich zu bezahlenden Paketen, die Möglichkeit, Spiele, die sich bereits lokal per Steam im Besitz des Users befanden über die Cloud zu spielen. Zusätzlich hatte der User, je nach bezahltem Paket, Zugriff auf eine über 250 Spiele umfassende Bibliothek. Darüber hinaus ermöglichte das OnLive Game System mittels Set-Top-Box das Cloud Gaming ebenfalls auf dem TV. OnLive verfügte über mehrere geographisch verteile Rechenzentren. Meldet sich ein Spieler am OnLive Portal-Server an wurde das Rechenzentrum mit der geringsten Entfernung gewählt. Die von OnLive angebotenen Spiele waren in unterschiedlichen Kombinationen nach dem in Punkt 2.3 erläutertem Verfahren auf den Servern installiert. Der Spieler wählte ein Spiel aus und bekam vom Portal-Server einen für die Dauer der Sitzung eigenen Spielserver zugewiesen [6]. Dies und die optimale Zuordnung

doi: 10.2313/NET-2015-09-1_06

des Rechenzentrums sollten ein gutes Spielerlebnis und eine hohe Spielqualität ermöglichen. Finkel et al. zeigt in seinem Paper [6] neben der Umsetzung von OnLive ebenfalls, dass die Methode in der die Spiele auf den Servern verteilt sind eine hohe Verfügbarkeit gewährleisten, jedoch einen hohen bedarf an Speicherplatz benötigen.

NVIDIA. Mit der GRID Technologie hat NVIDIA eine eigene Hardware für Cloud Gaming Server entwickelt [16]. Mit 4 Grafikprozessoren unterstützt die GRID Technologie bis zu 24 simultane Anwender. NVIDIA vertreibt die Technologie und ermöglicht es Unternehmen eigene Cloud Dienste zu betrieben. Zusätzlich betreibt NVIDIA mit Hilfe der Amazon Cloud Infrastruktur einen eigenen Cloud Gaming Dienst. NVDIAs GRID Cloud Gaming Dienst ist nur für Produkte aus der NVIDIA eigenen SHIELD Palette verfügbar. Darunter fallen neben einer stationären und einer tragbare Konsole auch ein Tablet. Der Beta Test umfasste Server in Kalifornien und User aus der ganze Welt. Da trotz der weiten Entfernung zu den Servern laut NVIDIA keine nennenswerten Probleme mit der Latenz auftraten, wird NVIDIA seinen Cloud Gaming Dienst weiter ausbauen [17]. Aktuell ist der Dienst in Deutschland noch nicht verfügbar.

Neben dem Cloud Gaming über NVIDIA Server, ermöglichen GeForce-GTX Grafikkarten das Streamen des Spiels vom PC des Users auf mobile Geräte. So wird der eigene PC zum Cloud Gaming Server. Dadurch sind dem User verschiedene Möglichkeiten gegeben. Innerhalb des lokalen WLAN kann über ein SHIELD Tablet auf den TV gestreamt werden oder von unterwegs mit einer Internetverbindung direkt auf die eigenen Spiele zugegriffen werden.

Sony. Mit den übernahmen der Cloud Gaming Dienste Gaikai im Jahr 2012 und OnLive im Jahr 2015 möchte Sony den Ausbau seines Cloud Gaming Dienstes Playstation Now weiter vorran treiben. Dieser bietet Cloud Gaming Dinste für mehr als 100 Spiele für alle Sony Konsolen und bestimmte Smart TVs an. Empfohlen wird hierbei eine Breitbandgeschinwidigkeit von 5-12Mbps [18].

StreamMyGame. Seit 2007 ist StreamMyGame auf dem Markt aktiv [21]. Es wird die passende Software für eine eigene Cloud Gaming Umgebung angeboten. Der heimische Spiele PC wird zum Cloud Gaming Server und sendet den Video- und Audio- Stream auf ein beliebiges internetfähiges Endgerät. Auf dem Server werden die Video Signale mittels OpenGL oder DirektX ausgelesen. Daraufhin werden sie komprimiert und an den Client gesendet. StreamMyGame unterstützt dazu eine Auswahl an verschiedenen Spielen. Zusätzlich bietet StreamMyGame die Möglichkeit, die Spiele aufzuzeichnen und der Community Zugang auf den eigenen Spielserver zu gewähren. In der kostenlosen Basisversion werden Auflösungen bis 640x480 Unterstützt in zwei möglich Upgrades kann die Auflösung auf die maximale Auflösung der eigenen Geräte angepasst werden.

Cloud Union. Bisher ausschließlich in China verfügbar, verfügt Cloud Union nach eigenen Aussagen über 20 Millionen aktive Kunden. Der im Jahr 2008 gestartet Service verfügt über Partnerschaften mit verschiedenen Spielentwicklern und bietet neben einem Client auch eine Flash Lösung für seine Cloud Gaming Dienste an [4].

3.2 Gaming Anywhere

Proprietäre Anwendungen wie die im Unterabschnitt 3.1 genannten, bieten wenig Möglichkeiten genauere Informationen über deren Spezifikationen und Umsetzungen zu erhalten. Gaming Anywhere als Open-Source Projekt, ermöglicht einen tieferen Einblick in die Materie des Cloud Gamings.

Gaming Anywhere ist portabel aufgebaut und verfügt aktuell über Lösungen für verschiedene Betriebssysteme. Durch austauschen der Plattform-abhängigen Komponenten wie die Audio- und Video- Erfassung kann die Software für beliebige Betriebssysteme angepasst werden. Ziel der Cloud Gaming Plattform ist es, Entwicklern und Wissenschaftlern eine Plattform für Multimedia Streaming zu bieten [9]. Dieses Ziel wird durch eine transparente Konfiguration unterstützt. Sämtliche Konfigurationen werden von der Plattform bereit gestellt und lassen sich per Textdatei auf die gewünschte Situation anpassen. Gaming Anywhere stellt alle nötigen Softwarepakete zur Verfügung um einen Cloud Gaming Server sowie einen entsprechenden Client zu konfigurieren.

Das Gaming Anywhere Prinzip beinhaltet die Komponenten Client, Portal-Server und Game-Server. Der Client kommuniziert zu Beginn mit einem Portal-Server, dessen Aufgabe neben dem Anmeldevorgang darin besteht, dem User eine Auswahl an unterstützten Spielen anzubieten und nach erfolgreicher Auswahl die URL des Game Servers zu übermitteln (vgl. Unterabschnitt 2.2). Ab diesem Zeitpunkt läuft die weitere Kommunikation zwischen dem Game-Server und dem Client ab. Auf dem Server arbeitet ein Agent, der einerseits für die Codierung und Übertragung des Audio und Video Streams zwischen Server und Client verantwortlich ist, und andererseits die Interaktionen des Clients an das Spiel auf dem Server weiterleitet. Der Agent hakt sich je nach Spiel entweder als Thread in das Spiel ein oder läuft als eigener Prozess parallel zum Spiel.

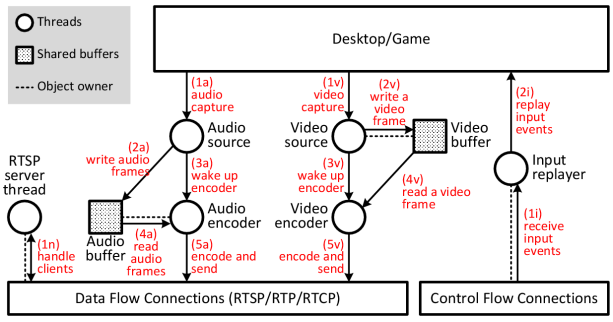

Abbildung 3: Modulaufbau des Cloud Gaming Server (Abbildung aus [10]).

doi: 10.2313/NET-2015-09-1_06

Wie Abbildung 3 zeigt, beinhaltet der Agent vier Module, einen RSTP Server, jeweils eine Audio- und Videoquelle sowie einen input replayer. Alle Module werden beim Start des Agent ebenfalls gestartet. Der RSTP Server wartet auf Befehle vom Client und dient zu Kontrolle der Echtzeitübertragung [19]. Sobald ein Client verbunden ist, werden die Audio und Videoquellen aktiviert und starten die entsprechenden Encoder. Die Audio- und Videoquellen beginnen mit der Erfassung der A/V Frames. Die Frames werden durch die Encoder konvertiert und mittels RTP und RCTP über TCP und UDP Verbindungen an den Client gesendet [10]. Auf dem Client laufen zwei Arbeitsthreads. Einer verarbeitet und sendet die Nutzereingaben. Der andere leitet den vom Server gesendeten Audio und Video-Stream an die entsprechenden Decoder weiter. Durch die modulare Struktur ist Gaming Anywhere mit geringem Aufwand erweiterbar. Dies ermöglicht es, einzelne Komponenten wie Codecs und Netzwerkprotokolle nach Vorstellungen des Nutzers zu erweitern oder zu ersetzen.

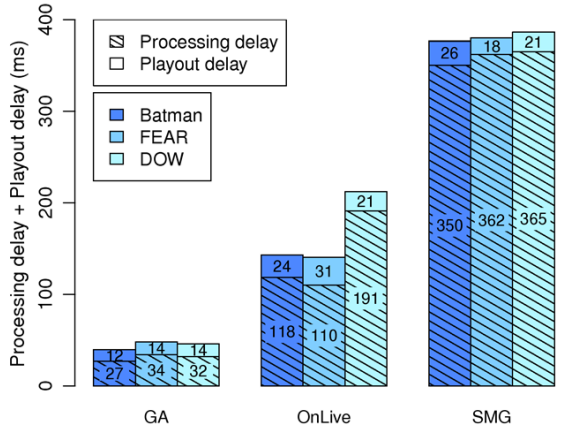

Abbildung 4: Gaming Anywhere im Vergleich (Abbildung aus [9]).

Im Vergleich zu kommerziellen Anbietern schließt Gaming Anywhere unter gleichen Voraussetzungen und Anforderungen bei Leistungstest in allen durch Huang et al. getesteten Punkten besser ab [9]. Abbildung 4 zeigt beispielsweise, dass Gaming Anywhere in dem bereits unter Punkt 2.3 angesprochen Process Delay 3 - 10 schneller als OnLive und StreamMyGame ist. Auch bei Videoqualität und benötigter Netzlast schließt Gaming Anywhere besser ab. Dies zeigt, dass Gaming Anywhere im Bereich des Encoding und Decoding sehr effizient arbeitet.

4. CLOUD GAMING VS. STATE OF THE ART

Cloud Gaming bietet im Vergleich zu konventionellem Gaming für Nutzer, Anbieter und Entwickler verschiedene Vor- aber auch Nachteile.

4.1 Vorteile

Aktuelle Video Spiele stellen hohe Anforderungen an die eingesetzten Endgeräte. Durch die stetige Entwicklung der Hardware verkürzt sich die Halbwertszeit der eigenen Geräte schnell. Cloud Gaming ermöglicht es, aktuelle Spiele auf älterer oder nicht dafür ausgelegter Hardware zu spielen. Durch die Auslagerung der Installation und Berechnung des Spiels auf einen Cloud Gaming Server, entfällt die regelmäßige Aufrüstung der eigenen Hardware. Dadurch verlängert sich der Produktlebenszyklus der eingesetzten Konsole, oder der einzelnen Hardwarekomponenten des PCs beträchtlich. So spart der Nutzer die Kosten für Neuanschaffungen und schont ebenfalls die Umwelt [13, S.152]. Durch die Auslagerung des Spiels in die Cloud, hat der Nutzer die Möglichkeit, eine entsprechende Breitbandverbindung vorausgesetzt, von überall auf seine Spiele und Spielstände zuzugreifen. Ebenfalls spielt die Wahl des dafür verwendeten Endgeräts eine untergeordnete Rolle, da sich im Grunde jedes internetfähige Endgerät, welches die vom Server gesendeten Audio und Video Streams decodieren und darstellen kann, zur Verwendung eignet. Da auf dem lokalen Geräten keine Installation mehr nötig ist, sind die Möglichkeiten in das Spiel mittels Eingriff in die Konfiguration oder durch Zusatzsoftware zu manipulieren deutlich geringer. Den Spielentwicklern bietet die Verschiebung der Spiele auf einheitliche Server die Möglichkeit Kosten in der Entwicklung zu sparen, da die Spiele nicht mehr für eine Vielzahl an Hardware kompatibel sein muss, sondern speziell auf die einzelnen Server optimiert werden können. Zusätzlich wird durch die zentrale Vermarktung und den exklusiven Zugang über die Plattformen der Cloud Gaming Anbieter die Möglichkeit der Produktpiraterie wesentlich erschwert.

4.2 Nachteile

Eine Grundvoraussetzung ist eine stabile und schnelle Internetverbindung. Diese wird von den Internetserviceprovidern jedoch nicht garantiert. Regionen die nicht über die benötigte Qualität der Breitbandverbindungen verfügen sind vom Cloud Gaming ausgeschlossen. Zusätzlich wirkt sich der Trend [22, S. 210 ff.] in der Telekommunikationsbranche zurück zu volumenabhängigen Breitbandverträgen kontraproduktiv auf das volumenintensive Cloud Gaming aus. Durch das Abonnieren von Cloud Gaming Diensten fallen monatlich Kosten für die Nutzung der Spiele und den Cloud Gaming Service an. Dadurch erlangt der Spieler jedoch nie das Eigentum an den Spielen. Darüber hinaus ist er von den Cloud Gaming Anbietern und deren Existenz abhängig. Sollte ein Cloud Gaming Anbieter seinen Service einstellen oder einzelne Spiele nicht mehr unterstützen, verliert der Spieler nicht nur den Zugang zu den Spielen sondern verliert ebenfalls sämtliche Spielstände. Zusätzlich beschränkt sich der Zugang auf die Spiele die der Anbieter unterstützt. Sollte ein bestimmter Spieltitel nicht im Angebot des Anbieters vorkommen, ist der Spieler gezwungen ein zusätzliches Abonnement bei einem weiteren Cloud Gaming Anbieter abzuschließen. Spieler von Online Multiplayer Spielen könnte so unter Umständen nur noch innerhalb eines Cloud Gaming Anbieters mit- beziehungsweise gegeneinander spielen. Mit der Auslagerung der Spiele auf Server, wird den Spielern zudem die Möglichkeit genommen, die Spiele selbst weiter zu entwickeln (Modden) und so eine treue und aktive Community aufzubauen. Eine aktive Community hält Spiele länger am Leben und bietet so auch lange nach der eigentlich Veröffentlichung noch Anreize das Spiel zu kaufen. Um ein möglichst störungsfreies und flüssiges Spielerlebnis zu gewährleisten, dürfen, wie in Unterpunkt 2.3 angesprochen,

die Entfernungen zwischen den Rechenzentren und den Spielern nicht zu großsein. Dies würde die Verzögerung zwischen Nutzereingabe und Reaktion im Spiel vergrößern. Das einloggen und spielen auf "fremden" Servern ist somit fraglich.

5. AUSBLICK

Cloud Gaming kann neben den Anbietern, den Entwicklern und Spielern auch Internetserviceprovidern und den Betreibern von Rechenzentren einen Nutzen bieten. Im Vergleich zu konventionellen Spielen bietet Cloud Gaming einen unterschiedlichen Ansatz. Durch die Auslagerung der Hardware-intensiven Komponenten eines Videospiels, eröffnen sich aus verschiedener Sicht neue Möglichkeiten. Auf der einen Seite stehen die Hersteller und Publisher der Videospiele Industrie, die zum einen die bisherigen Zielgruppen noch umfangreicher mit Inhalten versorgen und darüber hinaus zusätzlich neue Zielgruppen erreichen können. zusätzlich können Punkte wie die geringeren Entwicklungskosten durch einheitliche Hardware der Server und die exklusive online Vermarktung zu deutlichen Wettbewerbsvorteilen werden. Dazu ist der Schritt der Cloud Gaming Industrie aus der Nische erforderlich. Die dazu nötige kritische Masse an Nutzern kann erreicht werden, indem weiter an den Herausforderungen die Cloud Gaming mit sich bringt gearbeitet wird und mögliche Eintrittshürden weiter gesenkt werden. Im speziellen darf das Spielerlebnis keine signifikanten Unterschiede zu einem lokal installiertem Spiel aufweisen. Vor allem die Kategorie First Person Shooter erfordert hier eine sehr schnelle Umsetzung der Nutzereingaben um so eine minimale Verzögerung im Spielverhalten zu gewährleisten. Auf der anderen Seite stehen die Nutzer. Neben den Einsparungen die durch die geringeren Hardwareanforderungen könnte der permanente Zugriff auf die eigenen Spiele und die geringeren Manipulationsmöglichkeiten ein wesentliches Argument für Cloud Gaming Dienste werden. Trotz der Vorteile die Cloud Gaming für Anbieter und Anwender mit sich bringt, ist fraglich ob Cloud Gaming das konventionelle Gaming ablösen wird. Hierzu muss der Breitbandzugang weiter verbessert und und die Qualität der Breitbandverbindung gesichert werden. Zusätzlich muss sich die Bereitschaft der Nutzer zur monatlichen Zahlung weiter steigern. Eine Zusicherung der Nutzungsdauer der Spiele könnte hier eine Möglichkeit bieten. Sollten sich, ähnlich wie im Video on Demand Bereich, verschiedene Cloud Gaming Anbieter die exklusiven Rechte an einzelnen Spielen sichern, könnte sich dies negativ auf die Cloud Gaming Entwicklung auswirken, da potentielle Nutzer somit gezwungen wären unter Umständen mehrere Cloud Gaming Dienste zu abonnieren.

6. LITERATUR

[1] Chen et al.:*How sensitive are online gamers to network quality?*, In Communications of the ACM, 2006, 49. Jg., Nr. 11, Seite 34-38, ACM New York, NY, USA, 2006

[2] Chen, Kuan-Ta, et al.: *Measuring the latency of cloud gaming systems* In: Proceedings of the 19th ACM international conference on Multimedia, Seite 1269-1272, ACM New York, NY, USA, 2011

[3] Chen et al.: *On the Quality of Service of Cloud Gaming Systems* In Multimedia, IEEE Transactions on (Volume:16 , Issue: 2), Seite 480 - 495, IEEE, Feb. 2014

[4] *Cloud Union* `http://www.cloudunion.cn/` letzter Aufruf Mai 2015

[5] *Deutscher Gesamtmarkt für digitale Spiele (inkl. Hardware)*`http://www.biu-online.de/de/fakten/marktzahlen-2014/gesamtmarkt-digitale-spiele/gesamtmarkt-digitale-spiele-2014.html` Letzter Aufruf Mai2015

[6] D.Finkel et al.: *Assignment of games to servers in the OnLive cloud game system* In Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on. IEEE, Seite 1-3, 2014

[7] *Battlefield Hardline*`http://www.battlefield.com/de_DE/hardline`, Letzter Aufruf Mai2015

[8] Huang et al.:*Gaminganywhere: an open-source cloud gaming testbed* In Proceedings of the 21st ACM international conference on Multimedia, Seite 827-830, ACM New York, NY, USA, 2013

[9] Huang et al.: *GamingAnywhere: An Open Cloud Gaming System* In Proceedings of the 4th ACM Multimedia Systems Conference, Seite 36-47, ACM New York, NY, USA, 2013

[10] Huang et al.: *GamingAnywhere: The First Open Source Cloud Gaming System* In ACM Transactions on Multimedia Computing, Communications and Applications, Vol 10, No 1s, Seite 36-47, ACM New York, NY, USA, 2013

[11] *Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013*`http://www.gartner.com/newsroom/id/2614915`, Letzter Aufruf Mai2015

[12] *Industry Facts*`http://www.theesa.com/about-esa/industry-facts/`, Letzter Aufruf März2015

[13] F. Lampe: *Green-IT, Virtualisierung und Thin Clients : Mit neuen IT-Technologien Energieeffizienz erreichen, die Umwelt schonen und Kosten sparen* Vieweg + Teubner, Wiesbaden, 2010

[14] C. Meinel: *Virtualisierung und Cloud Computing: Konzepte, Technologiestudie, Marktübersicht* Universitätsverlag Potsdam, Potsdam, 2011

[15] *Nvidia Grid Grafikprozessoren - Technische Daten und Merkmale* `http://www.nvidia.de/object/cloud-gaming-gpu-boards-de.html` , Letzter Aufruf März 2015

[16] *Instantly Play Amazing Games* `http://shield.nvidia.com/grid-game-streaming` , letzter Aufruf Mai 2015

[17] *Nvidia's GRID cloud PC game-streaming service takes on Sony's PlayStation Now*`http://www.pcworld.com/article/2847752/nvidias-cloud-gaming-service-takes-on-sonys-playstation-now.html`, letzter Aufruf März2015

[18] *PlayStation Now* `https://www.playstation.com/en-us/explore/psnow/faq/?cid=psnow-logo-centered-threecolum-us-05jun14` letzer Aufruf Mai 2015

[19] H.Schulrinne et al.: *rfc2326. Real Time Streaming Protocol* Availab le on http://www.ietf.org/rfc/rfc2326.txt, 1998

[20] R. Shea, et al.: *Cloud gaming: architecture and performance* In Network, IEEE, 27. Jg., Nr. 4, Seite 16-21, 2013

[21] *StreamMyGame Page*, `http://goo.gl/MfDr07`, letzter Aufruf März 2015

[22] OECD: *OECD Communications Outlook 2013* OECD Publishing, 2013

[23] *OnLive Coustomer Service* `http://goo.gl/vFdLvT` Letzter Aufruf März 2015

[24] *OnLive Games on Demand* `http://www.onlive.com/` Letzter Aufruf März 2015

[25] *OnLive Games on Demand* `http://www.onlive.com/` Letzter Aufruf Mai 2015

[26] T. Wiegand et al.: *Overview of the H.264/AVC Video Coding Standard*, In Circuits and Systems for Video Technology, IEEE Transactions on (Volume:13 , Issue: 7) Seite 560 - 576, IEEE, July 2003

doi: 10.2313/NET-2015-09-1_06

Routing Caches in Software Packet Forwarding Devices

Christian Thieme
Tutor: Daniel G. Raumer und Paul Emmerich
Seminar Future Internet SS2015
Chair for Network Architectures and Services
Department of Computer Science, Technische Universität München
E-mail: christian.thieme@tum.de

ABSTRACT

Due to the rise of internet capable devices and the need to route their traffic, it is a challenge for every routing device to process the massive load in an acceptable period of time. Therefore to speed up the process of routing and to avoid the expenses for frequently upgrading Router hardware, the route cache has been invented. The route cache stores recently used routes and prevents time expensive lookups in the full Forwarding Information Base. This paper describes the packet forwarding process implemented in the Linux kernel. While the focus lies on the implemented routing cache in the Linux kernel, which was removed in version 3.7. Within the explanations of the routing cache behaviors, the Garbage Collector is also taken into account in regards of optimization and efficiency. It is a challenge to make the routing cache perform efficiently in every best- and worst-case traffic scenario. Also denial of service attacks specifically directed at the routing cache and the Garbage Collector are addressed as well as the cache hiding problem. Additionally, the open source software Open vSwitch in regards of the routing cache is discussed.

Keywords

Routing Cache, Packet Forwarding, Open vSwitch, Denial of Service, Linux Kernel, Forwarding Information Base, Cache Hiding

1. EINLEITUNG

Der Linux Kernel ist ein inzwischen weit verbreiteter Betriebssystemkernel und wurde 1991 von Linus Torvalds entwickelt [1]. Er ist als Teil von Linux Distributionen als auch in eingebetteten Systemen, wie z.B. Routern zu finden. Der Kernel bietet viele unterschiedliche grundlegende Funktionalitäten. Eine davon ist die Verarbeitung und Weiterleitung von Datenpaketen in Netzwerken. Um diesen Prozess möglichst effizient und zudem kostengünstig gestalten, ist in den Versionen vor 3.7 ein Routing Cache enthalten. Dieser wurde jedoch nach zwei Jahren andauernder Vorbereitung entfernt [17]. Dieses Paper beschäftigt sich damit, grundlegende Funktionen des Routing Prozesses im Linux Kernel zu erläutern und verständlich darzustellen. Es wird größtenteils auf IPv4 in Verbindung mit Linux Routing eingegangen. Im Folgenden wird unter Punkt 2 erklärt, wie das Verarbeiten und Weiterleiten von Paketen in Linux gehandhabt wurde, sowie die grobe Struktur dieses Prozesses. Punkt 3 geht näher auf den Routing Cache ein und erläutert auch Gründe, warum dieser entfernt wurde. Punkt 4 beschäftigt sich mit der Open Source Software Open vSwitch, welche weiterhin einen Routing Cache implementiert und diskutiert mögliche Gründe dafür. Punkt 5 gibt eine Zusammenfassung über die vorgestellten Ergebnisse.

2. PACKET FORWARDING

Packet Forwarding in Kern Internet Routern ist ein extrem herausfordernder Prozess. Wird ein IP Paket empfangen, haben Router nur ein paar Nanosekunden, das Paket zu puffern, den Longest Prefix Match, welcher das Ziel des Paketes bestimmt, zu ermitteln und das Paket an das korrespondierende aussendende Interface weiterzuleiten [10].

In der Grafik 1 ist eine Abstraktion des Ablaufes angegeben, welcher durchlaufen wird, wenn ein Paket an einem (virtuellen) Interface anliegt. Im Folgenden wird nun (approximativ) der Ablauf des IP Forwardings im Linux Kernel betrachtet.

Soll ein Paket, welches am Interface angekommen ist, verarbeitet werden, so wird im Empfangsmedium ein Interrupt ausgelöst. Das Empfangsmedium allokiert dann Speicherplatz für das anliegende Paket und übergibt dieses dem Bus, um es im allokierten Speicher abzulegen. Das Paket wird als nächstes der Link Schicht übergeben, welche das Paket in die Backlog Queue einfügt. Das Netzwerk Flag für den nächsten ‚bottom-half run[1]' wird markiert und das Empfangsmedium kehrt zu seinem vorherigen Prozess zurück. Wenn der Process Scheduler das nächste Mal läuft, bemerkt dieser, dass es Netzwerkaufgaben gibt, die bearbeitet werden sollen. Die Funktion *net_bh* entnimmt der Backlog Queue das Paket, vergleicht das Internet Protokoll und gibt das Paket weiter an die *receive* Funktion. Es wird nun auf Fehler untersucht. Das Paket wird nun entweder der Transport Schicht übergeben, falls es an diesen Host gesendet wurde, oder es bleibt auf der Netzwerk Schicht und wird geroutet [7]. Wie der Routing Prozess funktioniert und welche Besonderheiten es dort gibt, ist ausführlich in der Sektion 3 erläutert. Muss das Paket nun weitergeleitet werden, wird es überprüft und gegebenenfalls eine ICMP Fehlernachricht an den Sender übermittelt, falls ein Fehler auftritt [7]. Das Paket wird nun in einen neuen Puffer kopiert und gegebenenfalls fragmentiert, falls dies notwendig ist [7]. Dies kann z.B. der Fall sein, wenn die MTU (Maximum Transmission Unit) für das Zielnetzwerk, in welches das Paket geschickt werden soll, überschritten wird. Außerdem werden an dieser Stelle die MAC Adressen des Pakets verändert. Die ehemalige Ziel MAC Adresse wird zur neuen Quell MAC Adresse und die neue Ziel Adresse wird dem Routing Prozess entnommen. Schließlich wird es wieder an die Sicherungsschicht übergeben, welche das Paket

[1] Die Unterbrechungsroutine für ein Gerät ist oft in zwei Teile gespalten. Die 'top half' Routine ist die Routine, welche tatsächlich auf Unterbrechungen reagiert, und die anstehende Aufgabe zur Bearbeitung einplant. Die 'bottom-half' der Unterbrechungsroutine ist die Routine, welche die Aufgabe tatsächlich bearbeitet [9].

doi: 10.2313/NET-2015-09-1_07

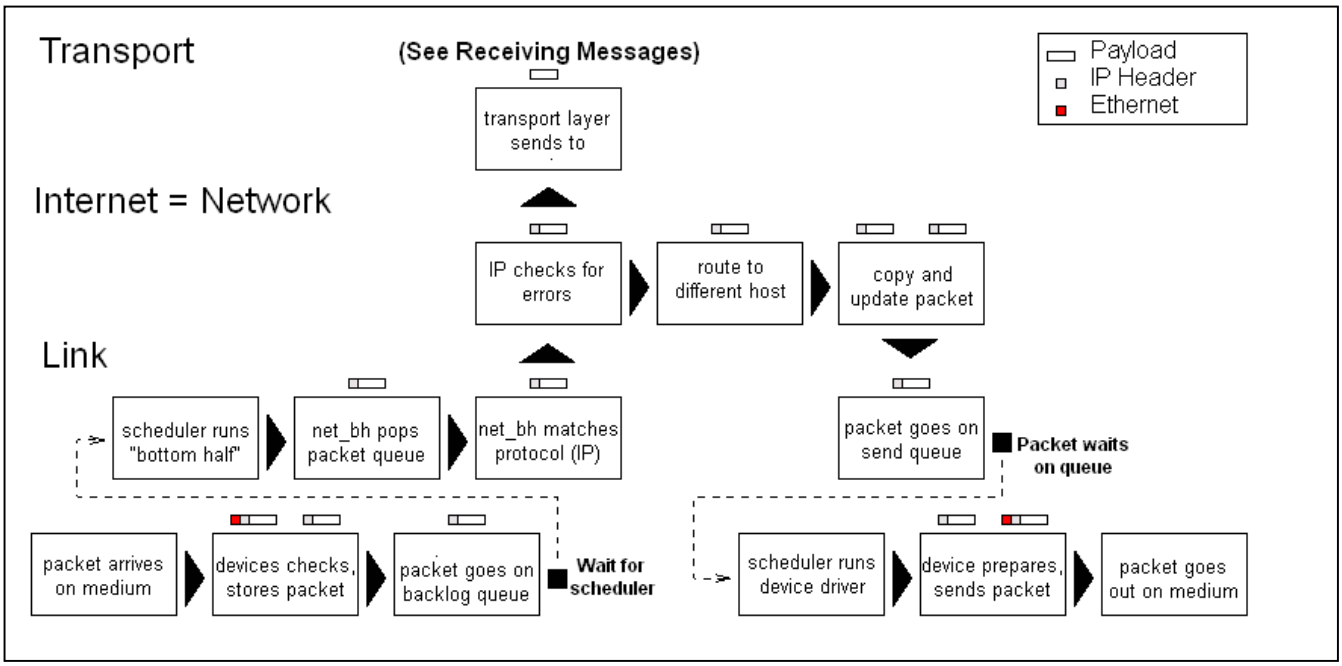

Abbildung 1: Abstraktes Diagramm vom Ablauf des IP Forwarding Prozesses im Linux Kernel. Quelle: [7]

in die Sendewarteschlange des Gerätes einreiht und sicherstellt, dass das Sendergerät weiß, dass es Verkehr zu senden hat [7]. Schlussendlich ordert das Sendegerät (wie beispielsweise eine Netzwerkkarte) seinen Bus an, das bzw. die Paket(e) zu senden [7].

Packet Forwarding kann sich auch auf andere Protokolle beziehen wie z.B. MPLS, ATM und weitere, welche hier nicht genauer behandelt werden.

3. ROUTING CACHE

3.1 Routing in Linux

Linux verwaltet(e) drei Komponenten, um Daten zu routen – eine für Computer, welche direkt mit dem Host verbunden sind (beispielsweise via LAN) und zwei für Computer, zu welchen nur indirekt eine Verbindung besteht [8].

Die Nachbartabelle beinhaltet Informationen über Computer, die physisch mit dem Host verbunden sind. Er beinhaltet Informationen darüber, über welches Gerät welcher Nachbar und welche Protokolle verwendet werden sollen, um Daten auszutauschen. Linux verwendet das Address Resolution Protocol (ARP), um diese Tabelle zu aktualisieren und zu verwalten [8].

Des Weiteren nutzt Linux zwei komplexe Komponenten von Routing Tabellen, um IP Adressen zu verwalten: eine Forwarding Information Base (FIB) mit Einträgen zu allen möglichen Adressen und einen Routing Cache gefüllt mit Daten von häufig genutzten Routen. Wenn ein IP Paket an einen weiter entfernten Host gesendet werden muss, überprüft die Netzwerkschicht zuerst den Routing Cache nach einem Eintrag mit passender Quelladresse, Zieladresse und passendem Diensttypen. Falls es dort einen solchen Eintrag gibt, wird er benutzt. Falls nicht, wird die Routing Information von der komplexeren (aber langsameren) FIB abgefragt, ein neuer Cache Eintrag mit den gefunden Daten wird generiert und der neue Eintrag dann benutzt [8].

Wie die FIB und der Routing Cache initialisiert werden, d.h. mit Einträgen bestückt werden, ist implementationsspezifisch von Anbieter zu Anbieter unterschiedlich und wird hier nicht genauer behandelt. Während die FIB Einträge semi-permanent sind (sie ändern sich nur wenn Router online oder offline gehen), verbleiben Einträge im Routing Cache nur so lange bis sie veraltet sind [8]. Liu et al. [11] stellen in ihrer Arbeit ein effizientes FIB Caching unter der Nutzung von minimalen nicht-überlappenden Präfixen vor.

3.2 Erklärung und Nutzen Routing Cache

Um mit eingehenden und ausgehenden IP Datagrammen umzugehen, muss der Kernel die Routing Tabellen abrufen [3]. Obwohl dies trivial erscheinen könnte, müssen vor der Weiterleitung einige Fragen beantwortet werden:

- Scheinen Quell- und Zieladresse valide zu sein?
- Ist die Quelladresse eine ‘Martian Address’?[2]
- Ist die Zieladresse die eigene oder soll das Paket weitergeleitet werden?
- Welche Routing Tabelle soll benutzt werden?
- Passt diese Zieladresse zu dieser Route?

[2] ‚Martian Addresses‘ sind Adressen die nicht als Quelladressen benutzt werden können, entweder weil sie für besondere Zwecke (so wie Multicast Adressen) reserviert sind oder wegen der Nutzung von ‚reverse path filtering‘, welches überprüft, ob das auf einem Interface erhaltene Paket auf demselben Interface beantwortet werden müsste, wie in RFC 3704 definiert [3].

doi: 10.2313/NET-2015-09-1_07

- Kann der zu nutzende Gateway momentan kontaktiert werden? [3]

Diese Überprüfungen können Zeit konsumierend sein. Um diese für jedes Paket zu vermeiden, verwaltet Linux einen Routing Cache, welcher abgefragt wird, bevor eine reguläre Suche gestartet wird und nach jeder dieser aktualisiert wird [3]. Wird eine Regel aus dem Routing Cache getroffen, müssen die oben genannten Fragen nicht mehr beantwortet werden, da diese Routen in der Regel erst kürzlich genutzt wurden und die benötigten Informationen im Puffer gespeichert sind.

Der Routing Cache ist die schnellste Methode, die Linux hat, um eine Route zu finden. Er behält jede Route, die momentan benutzt wird oder kürzlich benutzt wurde, in einer Hashtabelle [8]. Adressen werden mittels einer Hashfunktion auf Buckets abgebildet. Jeder Bucket enthält eine Liste, welche auch als Kette bezeichnet wird. Die Routen in den Ketten sind sortiert, am häufigsten genutzte Routen sind am Anfang, und besitzen eine Zeituhr und Zähler, welche sie von der Tabelle entfernen, wenn sie länger nicht mehr in Gebrauch waren [8].

Wird eine Routing Information benötigt, wird der passende Hashbucket durchsucht und die Kette von gepufferten Routen durchsucht bis ein Treffer gefunden wird. Dann wird das Paket auf diese Route geschickt. Falls es keinen Treffer im Routing Cache gibt, wird die FIB durchsucht. Gibt es dort einen Treffer, werden die erforderlichen Informationen ermittelt und die neue Route dem Routing Cache hinzugefügt. Falls es auch dort keine Treffer gibt, wird das Paket an eine default Route weitergeleitet (z.B. an einen Router in einer höheren Instanz). Pakete werden verworfen, falls unzulässige Quell- bzw. Zieladressen verwendet werden. So wird ein Home-Router Pakete verwerfen, die aus dem globalen Internet kommen und als Quelladresse die Broadcastadresse 255.255.255.255 haben.

3.2.1 Cache Hiding Problem

Linux nutzt das Longest Prefix Matching Verfahren, um Zieladressen mit gepufferten Adressen im Routing Cache (aber auch in der FIB) zu vergleichen. Bei dem Vergleich der Adressen wird die aufeinanderfolgende Anzahl an übereinstimmenden Bits (beginnend bei links) zweier IP Adressen gezählt und ab einer bestimmten Anzahl an Übereinstimmungen als valide gewertet.

Um den Longest Prefix Match effizient zu gestalten, nutzt Linux die *LC-Trie* (auch *Patricia Tree*) [14] Datenstruktur (für IPv6 wird ein *Radix Baum* verwendet). Beim Präfix Vergleich im Routing Cache kann möglicherweise auch das *Cache-Hiding* Problem auftreten. FIB Pufferung unterscheidet sich von traditionellen Puffer Mechanismen – selbst wenn für ein Paket ein passendes Präfix im Cache vorhanden ist, kann es trotzdem sein, dass es nicht der korrekte Eintrag ist, um das Paket weiterzuleiten, wenn es ein längeres passenderes Präfix in der FIB gibt [11]. Um das Problem anschaulicher zu gestalten, sind in den Tabellen 2 und 3 eine stark vereinfachte FIB und ein dazugehöriger Routing Cache abgebildet. Zum einfacheren Verständnis werden nur 8-bit lange Adressen betrachtet.

Tabelle 2: Beispieleinträge für eine FIB.

Kürzel	Präfix	Nächster Hop
A	11/2	1
B	101011/6	2
C	10101/5	4

Tabelle 3: Beispieleinträge für einen Routing Cache zur FIB in Tabelle 3.

Kürzel	Präfix	Nächster Hop
A	11/2	1
C	10101/5	4

Ausgehend von der FIB in Tabelle 3 und dem dazugehörigen Routing Cache in Tabelle 4 soll ein Paket mit der Zieladresse ***11010101*** geroutet werden. Zunächst wird der Routing Cache durchsucht und sogleich ein Treffer mit dem Eintrag beim Kürzel ***A*** erzielt, da ***A*** das mit ***11/2*** das längste passende Präfix zu der gesuchten Adresse besitzt. Das Paket wird nun über den nächsten ***Hop 1*** weitergeleitet. Soweit tritt kein Problem auf. Angenommen ein weiteres Paket mit der Zieladresse ***10101110*** soll weitergeleitet werden. Zunächst wird wieder der Routing Cache nach einem passenden Präfix durchsucht. Der Eintrag mit Kürzel ***C*** wird nun ausgewählt, da er mit den ersten 5 übereinstimmenden Bits der beste Eintrag im Routing Cache ist. Durch den Cache Eintrag ***C*** wird allerdings der passendere Eintrag in der FIB übergangen. Eintrag ***B*** in der FIB (Tabelle 3) hätte, statt 5, 6 übereinstimmende Bits und wäre somit die bessere Wahl gewesen. Das Paket wird also nun sozusagen fälschlicher Weise an den nächsten ***Hop 4*** statt ***2*** gesendet. Damit wird der passendere Eintrag in der FIB durch den Eintrag im Routing Cache versteckt.

3.2.2 Garbage Collector

Um den Route Cache angemessen zu verwalten und veraltete Routen, welche zu lange ungenutzt im Cache verweilen, zu löschen, nutzt Linux den Garbage Collector (im Folgenden GC genannt). Der GC spielt eine extrem wichtige Rolle in der Effizienz des Routing Caches. In durch default Einstellungen oder durch den Benutzer festgelegten Intervallen durchläuft der GC alle Einträge des Routing Caches. Zunächst evaluiert der GC die Situation, indem er die momentane Größe des Caches mit seinem konfigurierten Wert vergleicht. Ist die Anzahl der Cache Einträge kleiner oder gleich des Produkts von *rhash_entries* und *gc_elasticity*, wird der GC versuchen, ***höchstens die Hälfte der Differenz*** aus dem Routing Cache zu entfernen, auch wenn dadurch veraltete Einträge im Cache verbleiben sollten. Enthält beispielsweise der Routing Cache 1,5 Mio Einträge und im GC ist der Wert 2 Mio Einträge eingestellt (*rhash_entries* 400.000 und *gc_elasticity* 5), wird er versuchen höchstens (2 Mio – 1,5 Mio) / 2 = 250.000 Einträge zu löschen. Erreicht der GC sein Maximum an zu löschenden Einträgen bevor er den kompletten Cache durchlaufen ist, merkt er sich, wo er aufgehört hat und startet bei seinem nächsten Aufruf an der gemerkten Stelle. Welche Einträge vom GC entfernt werden, hängt von dem eingestellten *gc_timeout* Wert ab. Nur wenn das Alter des ersten Eintrages in einer Kette kleiner als *gc_timeout* wird der Eintrag beibehalten. Wenn es der zweite Eintrag ist, wird er nur behalten, wenn er halb so alt ist wie *gc_timeout*. Der dritte nur dann noch, wenn das Alter unter einem Viertel von *gc_timeout* ist (vlg. [3]). Ob ein Eintrag behalten wird, entscheidet sich also unter der Bedingung ob *„Alter des Eintrags“ < gc_timeout / (2^“Platz in der Kette“)* erfüllt ist. Wobei zu beachten ist, dass der Platz in der Kette bei 0 beginnt und aufsteigend nummeriert wird. Der GC bevorzugt kurze Ketten. Überschreitet die Größe des Routing Caches allerdings das Produkt der im GC eingestellten Werte, wechselt der GC in einen aggressiven Modus. Im aggressiven Modus wird der GC

doi: 10.2313/NET-2015-09-1_07

versuchen, mindestens *rhash_entries* zu entfernen. Der GC wird aufgerufen, wenn ein neuer Cache Eintrag hinzugefügt werden soll und die Anzahl der Cache Einträge *gc_thresh* übersteigt. Mit *gc_interval* lässt sich zudem ein reguläres Intervall einstellen, in welchem der GC aufgerufen werden soll. Der GC kann nur aufgerufen werden, wenn die Anzahl der Einträge im Routing Cache *max_size* (eine Einstellung für die Mindestgröße des Caches) übersteigt und der GC nicht innerhalb der letzten *gc_interval_ms* Millisekunden bereits aufgerufen wurde. Weiterführende und detailliertere Beschreibungen des GC und wie man diesen optimieren kann, können im Blog von Bernat [3] in Erfahrung gebracht werden.

Ist der GC unpassend für das vorliegende Netzwerk eingestellt, können unterschiedliche und möglicherweise unerwünschte Ereignisse eintreten. Beispielsweise könnte der Routing Cache sich zu langsam oder mit zu wenigen Einträgen füllen und produziert damit viele C*ache Misses*, welche die Effizienz des Routings negativ beeinflussen können, da für einen *Cache Miss* anstatt einer, beide Routing Tabellen abgefragt werden müssen und die FIB manchmal in langsamerem Speicher abgelegt ist. Das Hauptproblem beim Konfigurieren des Routing Caches bzw. des GCs ist, dass sie abhängig davon sind, welcher und wie viel Verkehr zu Routen ist. Die Effizienz des Routing Caches ist somit von äußeren Umständen abhängig und somit von außen kontrollierbar (vgl. [13]). Somit gibt es bisher im Linux Kernel keine allgemein optimale Konfiguration des Routing Caches, da der Linux Kernel in verschiedenen Umgebungen zum Einsatz kommt.

Der Routing Cache ist keine neue Erfindung. In den späten 1980er und 1990er Jahren hatten die meisten Router eine Route Caching Fähigkeit eingebaut. Allerdings waren diese Designs nicht in der Lage mit den schnell ansteigenden Packet Forwarding Raten Schritt zu halten. Wegen der hohen Kosten für *Cache Misses*, aufgrund derer es niedrigeren Durchsatz von Paketen gab, waren ein hoher Verlust von Paketen das Resultat [10].

Route Caching kann aus den folgenden Gründen (abgesehen von Effizienz) notwendig sein. Neue Protokolle mit größeren (z.B. IPv6) oder flacheren Adressräumen wurden vorgeschlagen, um das Wachstum und die Konfigurierung des Internets zu erleichtern. Jedoch, würden diese Protokolle eingesetzt werden, stiege die Größe der FIB so signifikant an, dass die Kapazitäten der meisten momentan verwendeten Ausrüstung überschritten wären. Und es ist vorhergesagt, dass mehrere Millionen FIB Einträge innerhalb einiger Jahre benötigt werden, falls der momentane Wachstumstrend weitergehen sollte [10]. Wenn FIBs sich füllen, stürzen herkömmliche Router ab oder beginnen sich inkorrekt zu verhalten [4]. Dies kann Operatoren dazu zwingen neue ‚Line Cards'[3] oder gar Router mit größerem Speicher einzusetzen [10].

Die Größe der globalen Routing Information Base (RIB) wächst in einem alarmierenden Tempo an. Dies führt direkt zu einem rapiden Wachstum der globalen FIB Größe, welche ernste Probleme für ISP[4] aufkommen lässt, da FIB Speicher in Line Cards viel teurer sind, als reguläre Speichermodule und es sehr kostspielig für ISP ist, diese Speicherkapazität regelmäßig für alle Router zu erhöhen. Eine potentielle Lösung ist es, die beliebtesten FIB Einträge in schnellem Speicher, also einen FIB Cache, zu installieren [11].

3.3 Die Entfernung des Routing Caches aus dem Linux Kernel

In der Linux Kernel Version 3.7 ist der Routing Cache nicht mehr enthalten. David S. Miller, einer der Hauptverantwortlichen für die Entwicklungen im Netzwerkbereich des Linux Kernels [2], begründet die Entfernung des Routing Caches in seinem Commit des Updates [13]. Der Routing Cache ist nicht-deterministisch, bezogen auf Effizienz, und ist Gegenstand von einigermaßen einfach zu startenden *Denial of Service* Attacken. Seiner Meinung nach, arbeite der Routing Cache großartig für ‚braven' Verkehr und die Welt sei ein viel freundlicherer Ort gewesen, als die Gründe, die zum Design des Routing Caches führten, diskutiert wurden. Selbst für ‚braven' legitimen Verkehr sehen hochfrequentiert besuchte Seiten in den Routing Caches Trefferraten von nur ~10% [13].

Denial of Service (DoS) Attacken oder auch *Distributed Denial of Service* Attacken (DDoS) sind Angriffe, die einen Service im Internet (z.B. eine Internetseite, ein Router oder ein Server) auf Dauer oder zeitweilig unerreichbar machen sollen (vgl. [16]). Dies kann auf unterschiedliche Weise erreicht werden. Beispielsweise können *Overflows* in Tabellen oder Speichern provoziert werden oder die Ressourcen des Zielgerätes werden ausgereizt (z.B. CPU Auslastung maximieren oder RAM überladen). Ein möglicher solcher Angriff, der den Routing Cache als Ziel hat, kann unterschiedliche Absichten verfolgen. Eine davon könnte sein, dass möglichst viele Einträge im Routing Cache auf einen oder einige wenige Buckets verteilt werden. Dies führt dazu, dass das Durchsuchen des Routing Caches nach einem Element dem Durchsuchen einer linearen Liste näher kommt, als der Suche nach einem Element in einer Hashtabelle. Damit würde sich die Suchzeit für Einträge stark erhöhen, und somit der Durchsatz von Paketen insgesamt deutlich verringern. Eine Voraussetzung für diesen Angriff wäre, dass der Angreifer weiß, welche Hashfunktion gerade genutzt wird. Die Ausführung dieses Angriffes wurde bereits im Jahre 2003 erschwert, indem ein Patch die Hashfunktion verschlüsselte, nicht-linear machte und zudem die Hashfunktion ca. alle 10 Minuten variierte (vgl. [21]). Eine weitere Möglichkeit, den Route Cache ineffektiv zu machen, wäre es dafür zu sorgen, dass maximal viele C*ache Misses* beim Durchsuchen des Routing Caches auftreten. Dies würde dazu führen, dass sowohl der Route Cache als auch die FIB immer wieder abgefragt werden müssen. Beschäftigt man das System damit ausreichend, kann dafür gesorgt werden, dass normale (Nicht-Angreifer) Pakete nicht mehr oder nur teilweise verarbeitet bzw. weitergeleitet werden. Obwohl eine bestimmte Menge an Paketen zur Verarbeitung in Zwischenspeichern kurzweilig eingereiht werden können, hat dies kaum eine Auswirkung auf Angriffe dieser Art. Eine Möglichkeit, diesen Angriff zu entkräften wäre es, Pakete die keinen Treffer im Routing Cache haben, an einen anderen Router zu senden, der Routing Entscheidungen nur aufgrund der FIB trifft. Ein weiterer Schutzmechanismus vor solchen Angriffen ist *Ingress-Filtering*, worauf in diesem Paper kein Bezug genommen wird.

Ein weiteres und effektives Ziel des Angreifers könnte es sein, den Routing Cache so stark aufzublähen, dass der GC die Größe des Routing Caches nicht schnell genug reduzieren kann. Dies sollte möglich sein, indem in sehr kurzer Zeit extrem viele Pakete verarbeitet werden sollen. Es könnten nun mehrere Szenarios

[3] ‚Line Card' http://www.cscprinters.com/line-card.html

[4] ‚ISP' – Internet Service Provider

doi: 10.2313/NET-2015-09-1_07

auftreten. Bei jedem *Cache Miss* wird eine Route aus der FIB in den Routing Cache geladen und dabei der GC aufgerufen. Da aber *gc_interval_ms* die Häufigkeit des Aufrufens des GC beschränkt, sollte es möglich sein, eine Überladung der Routing Tabelle zu erreichen. Alternativ könnten aber auch durch das häufige aufrufen des GCs Pakete verloren gehen, da während der GC läuft, keine Zugriffe auf die Tabelle zugelassen sind (Semaphorenproblem). Eine weitere Alternative, die zu einer Überladung der Tabelle führen könnte, ist, dass durch den GC nicht genügend Tabelleneinträge entfernt werden, da *gc_timeout* bestimmt, welche Tabelleneinträge vom GC entfernt werden.

Der Hauptgrund für die Entfernung des IPv4 Routing Caches war, dass DoS-Attacken gegen ihn einfach waren, da der IPv4 Routing Cache für jeden einzigartigen Flow einen Cache Eintrag generiert. Im Wesentlichen bedeutet das, dass es durch das Senden von Paketen an zufällige Adressen möglich ist, eine unbeschränkte Menge an Einträgen im Routing Cache zu generieren [18].

Obwohl der Routing Cache im Jahre 2012 entfernt wurde, wird erwogen, diesen wieder optional im Linux Kernel zur Verfügung zu stellen [12].

4. OPEN VSWITCH & ROUTING CACHES

Open vSwitch (OvS) ist eine Open Source Software, die einen bzw. mehrere virtuelle Switches implementiert. Sie bieten Netzwerkzugang für Virtual Machines (VMs), indem virtuelle und physische Netzwerk Schnittstellen verbunden werden [5]. Vor Open vSwitch bietet auch der Linux Kernel eine schnelle und verlässliche Bridge (die Linux Bridge) an (vgl. [6]). Die Zielgruppe von Open vSwitch ist allerdings an Multi-Server Deployments ausgerichtet, eine Umgebung, für welche die vorherigen Programme nicht gut ausgelegt waren [6]. Somit findet sich Open vSwitch beispielsweise oft in Cloud Netzwerken wieder.

Wie auch einst der Linux Kernel implementiert OvS einen Routing Cache, wenn es um Packet Forwarding geht. Wie auch bei dem Linux Kernel sind DoS-Attacken auf den Routing Cache möglich. Open vSwitch Entwickler sind sich dessen bewusst und versuchen deshalb Schutzmechanismen einzuführen, welche DoS-Attacken zumindest erschweren sollen. Beispielsweise wurde der GC für die Verwaltung des Caches abgeschafft. In Verbindung damit wurde der *Flow Mask Cache* eingeführt. Um ein beliebiges Paket zu verarbeiten muss OvS eine Flow Mask für den Flow erzeugen und diese dann im Flow Cache abfragen [20].

Wie der Linux Kernel legt auch OvS einen Trie an, um den Longest-Prefix Match möglichst schnell zu bestimmen. Um diesen Prozess weiter zu beschleunigen werden statt Microflows Megaflows bevorzugt. Also könnte man von einem Megaflow Cache sprechen. Anhand des folgenden Beispiels lässt sich diese Vorgehensweise veranschaulichen. Zum Vereinfachten Verständnis werden IPv4 Adressen, welche nur nach oktalen Präfixen unterschieden werden (z.B. /8, /16, /24 und /32), verwendet. Angenommen OvS möchte einen Megaflow für ein Packet generieren, welches an die Adresse 1.2.3.4 gesendet werden soll. Besteht nun nur ein Match für beispielsweise das erste Oktett, also 1/8, wird der Trie nur um das Präfix 1.2/16 erweitert, statt um das Präfix 1.2.3.4/32. Dieses Vorgehen verkürzt den Longest Prefix Match für folgende Pakete [15].

Um bei großem Paketdurchsatz effizient zu bleiben, verfügt der Flow Cache maximal über 200.000 Einträge. Um außerdem zu verhindern, dass mehr Flows hinzugefügt werden, als gelöscht werden, wurden zusätzliche Threads eingeführt, die Routen während der Verarbeitung von Paketen revalidieren. Dies geschieht indem Cache-Statistiken genutzt werden, welche ungefähr einmal pro Sekunde vom Kernel abgefragt werden. Der Kernel gibt dann einen Zähler (eine Variable, die zählt, wie oft ein Flow seit der letzten Abfrage genutzt wurde) für die jeweiligen Flow Einträge zurück und entscheidet basierend auf dem Zähler, ob der Flow beibehalten oder gelöscht wird. Der Cache passt sich im Betrieb so an, dass er für die ein Revalidierungsintervall weniger als eine Sekunde benötigt. Diese Vorgehensweise soll den Revalidierungsprozess auch bei großem Paketdurchsatz effizient halten [15].

Der Hauptgrund, einen Routing Cache zu verwalten, sei laut Miller hauptsächlich auf Effizienzgründe zurückzuführen (vgl. [20]). Eine Effizienz Analyse von Virtual vSwitch von Emmerich et al. [5] veranschaulicht diesen Umstand. In einer anderen Arbeit von Rotsos et. al [19] wird die Implementation von OpenFlow[5] in verschiedenen Routern bzw. Switches untersucht, darunter auch Open vSwitch, und deren Performanz verglichen.

Um den Routing Cache zu umgehen und somit auch Angriffe, empfiehlt Miller die darunterliegende Datenstrukturzugriffe schneller zu machen, um somit Abfragen effizient zu gestalten (vgl. [20]). Im Linux Kernel ist dies vermutlich durch die Implementierung des *LC-Trie* realisiert worden, welcher Longest Prefix Abfragen effizienter gestalten kann.

5. ZUSAMMENFASSUNG

In dieser Arbeit wurde die grundlegende Struktur der Paketverarbeitung im Linux Kernel erläutert. Besonderes Augenmerk liegt auf dem vom Linux Kernel verwalteten Routing Cache, welcher in neueren Versionen nicht mehr enthalten ist. Allgemein lässt sich sagen, dass der Routing Cache in erster Linie als Effizienz Werkzeug angesehen werden kann, aber auch als eine Möglichkeit Kosten bei Internet Routern einzusparen, da diese dann nicht regelmäßig mit teurem Hochleistungsspeichern wieder aufgerüstet werden müssen. Allerdings birgt ein Routing Cache nicht nur Vorteile, denn es kann sich als schwer erweisen, diesen passend für seine Netzwerkumgebung zu konfigurieren. Außerdem ist er, trotz aller Anstrengungen von Entwicklern, ein großes Sicherheitsrisiko. Es können relativ leicht ausführbare Paketmassen bzw. -abfolgen den Routing Cache ineffizient machen oder überlasten. Trotz der bekannten Risiken bezüglich der Anfälligkeit gegenüber DoS-Attacken scheinen Open vSwitch Entwickler in näherer Zukunft den Routing Cache nicht entfernen zu wollen. Andererseits haben auch Linux Entwickler bereits erwogen, den Routing Cache, diesmal zwar optional, erneut einzuführen.

6. ACKNOWLEDGEMENTS

Diese Arbeit wurde im Rahmen eines Seminars an der Technischen Universität München am Lehrstuhl für Rechnernetze und unter Betreuung von Daniel Raumer und Paul Emmerich angefertigt.

[5] http://openflow.org

7. REFERENCES

[1] Linux kernel readme. http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/README?id=f3b8436ad9a8ad36b3c9fa1fe030c7f38e5d3d0b. Accessed: 2015-03-29.

[2] List of Maintainers. http://lxr.linux.no/L3703. Accessed: 2015-03-31.

[3] Vincent Bernat. Tuning Linux IPv4 route cache. http://vincent.bernat.im/en/blog/2011-ipv4-route-cache-linux.html, 2012. Accessed: 2015-03-30.

[4] Di-Fa Chang, Ramesh Govindan, and John Heidemann. An empirical study of router response to large bgp routing table load. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 203–208, New York, NY, USA, 2002. ACM.

[5] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle. Performance characteristics of virtual switching. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pages 120–125, Oct 2014.

[6] Thomas Graf. Why Open vSwitch? https://github.com/openvswitch/ovs/blob/master/WHY-OVS.md, October 2014. Accessed: 2015-04-01.

[7] Glenn Herrin. Linux IP Networking - A Guide to the Implementation and Modification of the Linux Protocol Stack: Chapter 7. http://www.cs.unh.edu/cnrg/people/gherrin/linux-net.html#tth_chAp7, May 31 2000. Accessed: 2015-03-29.

[8] Glenn Herrin. Linux IP Networking - A Guide to the Implementation and Modification of the Linux Protocol Stack: Chapter 8. http://www.cs.unh.edu/cnrg/people/gherrin/linux-net.html#tth_chAp8, May 31 2000. Accessed: 2015-03-30.

[9] Alessandro Rubini Jonathan Corbet, Greg Kroah-Hartman. *Linux Device Drivers*. O'Reilly, 3 edition, February 2005. Ch 10.4 Interrupt Handling.

[10] Changhoon Kim, Matthew Caesar, Alexandre Gerber, and Jennifer Rexford. Revisiting route caching: The world should be flat. In SueB. Moon, Renata Teixeira, and Steve Uhlig, editors, *Passive and Active Network Measurement*, volume 5448 of *Lecture Notes in Computer Science*, pages 3–12. Springer Berlin Heidelberg, 2009.

[11] Yaoqing Liu, Syed Obaid Amin, and Lan Wang. Efficient fib caching using minimal non-overlapping prefixes. *SIGCOMM Comput. Commun. Rev.*, 43(1):14–21, January 2013.

[12] David S. Miller. Things that need to get done in the linux kernel networking. http://vger.kernel.org/ davem/net_todo.html. Accessed: 2015-03-31.

[13] David S. Miller. ipv4: Delete routing cache. http://git.kernel.org/cgit/linux/kernel/git/davem/net-next.git/commit/?id=89aef8921bfbac22f00e04f8450f6e447db13e42, July 2012. Accessed: 2015-03-31.

[14] S. Nilsson and G. Karlsson. Ip-address lookup using lc-tries. *Selected Areas in Communications, IEEE Journal on*, 17(6):1083–1092, Jun 1999.

[15] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA, May 2015. USENIX Association.

[16] Radware. *DDoS Surivial Handbook*. Radware, 2013.

[17] Éric Leblond. David miller: routing cache is dead, now what? https://home.regit.org/2013/03/david-miller-routing-cache-is-dead-now-what/. Accessed: 2015-03-29.

[18] Rami Rosen. *Linux Kernel Networking - Implementation and Theory*. Apress, 2014. p. 134.

[19] Charalampos Rotsos, Nadi Sarrar, Steve Uhlig, Rob Sherwood, and Andrew W. Moore. Oflops: An open framework for openflow switch evaluation. In *Proceedings of the 13th International Conference on Passive and Active Measurement*, PAM'12, pages 85–95, Berlin, Heidelberg, 2012. Springer-Verlag.

[20] Pravin B Shelar. [Git net-next] Open vSwitch. http://comments.gmane.org/gmane.linux.network/325250, August 2014. Accessed: 2015-04-01.

[21] Florian Weimer. Algorithmic Complexity Attacks and the Linux Networking Code. http://www.enyo.de/fw/security/notes/linux-dst-cache-dos.html, July 2003. Accessed: 2015-03-30.

Leveraging SDN for DDoS defenses

Pirmin Blanz
Betreuer: Quirin Scheitle
Innovative Internet-Technologien und Mobilkommunikation SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: p.blanz@tum.de

KURZFASSUNG

Aktuelle Beispiele [1, 2] zeigen, das noch immer kein Wundermittel gegen großflächig angelegte Angriffe gefunden wurde, welche die Verfügbarkeit verschiedenster Internetdienste bedrohen. Die Rede ist von DDoS-Angriffen (*Distributed Denial of Service*). Initiiert und gesteuert von einigen wenigen Ausgangspunkten wird der Angriff schließlich auf hunderte, tausende oder zehntausende [3] Netzwerkknoten ausgeweitet und auf dem Rücken dieser (meist unwissenden) Helfer ausgetragen. Die Beteiligten bombardieren das Opfer schließlich millionenfach mit Anfragen, was zu einem Angriffsverkehr von über 300Gbps [4] führen kann. Durch Werkzeuge der etablierten Netzwerktechnik können DDoS-Offensiven kaum zeitnah erkannt und dadurch erfolgreich verteidigt werden. Mit dem Einsatz zentraler Steuergeräte liefert das SDN-Paradigma (*Software Defined Networking*) effektive Lösungsstrategien. Nachfolgend werden diese analysiert und herkömmlichen Verteidigungsmaßnahmen gegenübergestellt. Grundlage und Raster bildet hierbei eine Taxonomie, welche bekannte Angriffsmuster abstrahiert und kategorisiert.

Schlüsselworte

DDoS, SDN, DDoS-Verteidigung

1. EINLEITUNG UND MOTIVATION

Ungeachtet der Tatsache, das DDoS-Angriffe und deren Ausmaße seit über 20 Jahren bekannt sind, verursachen diese aktuell immer noch immense Schäden. Akamai [5] zufolge hat sich die Zahl der Vorfälle im Vergleich zum entsprechenden Vorjahreszeitraum sogar mehr als verdoppelt. Generell ist eine stark steigende Tendenz zu verzeichnen.

Dieses Wachstum hat mehrere Gründe. Als Voraussetzung hierfür gilt die Durchführbarkeit von DDoS-Angriffen. So werden Schwachstellen aktiver Netzwerkknoten, wie etwa Sicherheitslücken von Webservern [7] oder Betriebssystemen benutzt, um *Bot*-Netzwerke (kurz: *Botnet*) zu züchten. Von einem Befehlsgeber entsprechend instruiert kann ein Botnet dazu verwendet werden, die für einen DDoS-Angriff notwendige Datenflut auszulösen. Mithilfe von DDoS-Angriffswerkzeugen wie *Stacheldraht* [8] ist es sogar IT-Laien möglich kleinere Webserver zu überlasten. Des Weiteren, wie [6] aufzeigt, entwickelt sich rund um DDoS-Angriffe ein komplett neuer Wirtschaftszweig. Hier können zahlende Kunden DDoS-Dienste wie bspw. den *Lizard Stresser* in Anspruch nehmen, um gezielt Webserver (bspw. von Konkurrenten) in die Knie zu zwingen.

Die leichte Durchführbarkeit ist allerdings nur ein Aspekt, welcher den DDoS-Boom plausibel erscheinen lässt. Was Angreifern darüber hinaus entscheidend in die Hände spielt, ist die Tatsache, dass sich Betroffene oft nicht effizient gegen DDoS-Angriffe verteidigen können [9]. Dies liegt zum einen daran, dass Angreifer anormal große Datenmengen in Richtung des Opfers schicken, die schließlich das durchschnittliche Verkehrsvolumen um ein vielfaches übersteigen. Zum anderen ist die Detektion von Angriffspaketen häufig kompliziert. Die Pakete unterscheiden sich kaum oder überhaupt nicht von denen, die Anfragen realer Kunden repräsentieren. Diesen soll der Dienst natürlich nicht verweigert werden.

Es wurden bereits zahlreiche Lösungsvorschläge zur DDoS-Verteidigung mit konventionellen Methoden veröffentlicht [10]. Dennoch entwickeln Angreifer immer ausgefeiltere Techniken, um bestehende Barrieren zu umgehen. Wie in vielen sicherheitskritischen Bereichen liefern sich Aggressoren und Verteidiger auch hier einen Wettkampf, bei dem Erstere meist die Nase vorn haben. Das führt zu einer raschen Entwicklung in der DDoS-Sparte mit einer scheinbar nicht zu überblickenden Vielfalt an verschiedenen Angriffsvektoren.

Das SDN-Paradigma weist für die Verteidigung von DDoS-Angriffen hervorragende Eigenschaften auf. Durch die Trennung der *Forwarding*-Ebene von der *Control*-Ebene entsteht ein Netzwerk, bei dem eine (replizierte) Kontrollinstanz die Forwarding-Logik zugeordneter Netzwerkknoten plant und verwaltet. Router werden durch simple *SDN-Switches* ersetzt.

Im zweiten Abschnitt der Arbeit wird zunächst eine Taxonomie von DDoS-Angriffen vorgestellt und im dritten Kapitel anhand von Beispielen veranschaulicht. Der vierte Abschnitt ist konventionellen Verteidigungsmechanismen gewidmet. Analog dazu werden im fünften Abschnitt Verteidigungsmechanismen, die sich des SDN-Paradigmas bedienen, erläutert. Im sechsten Kapitel diskutieren wir die Verteidigungsansätze aus beiden genannten Welten. Hier werden Ähnlichkeiten und Unterschiede herausgearbeitet. Den Abschluss bildet Kapitel sieben, welches über die Risiken beim Einsatz von SDN aufklärt.

2. DDOS-ANGRIFFE - EINE TAXONOMIE

Es existieren zahlreiche Fachartikel, die eine Kategorisierung von DDoS-Angriffen vornehmen. So unterscheiden sowohl Douligeris und Mitrokotsa [13] als auch Specht und Lee [12] zwischen zwei verschiedenen Angriffstypen: Jene, welche das Netzwerk des Opfers verstopfen und Angriffen, deren Erfolg die Überlastung wichtiger Systemressourcen (bspw. Hauptspeicher oder CPU) zur Folge haben. Mirkovic und Reiher liefern in [10] eine wesentlich detaillierte Taxonomie. Die

doi: 10.2313/NET-2015-09-1_08

Autoren berücksichtigen in ihrem Artikel bereits Faktoren, welche sich auf den Grad der Automatisierung beim Aufbau einer Botnet-Infrastruktur beziehen. Ähnlich zu [13] werden auch hier semantische- und *Brute-Force*-Angriffe genannt. Weitere Kategorien sind: Art der Quelladresse (*spoofed* oder original), Dynamik in der Masse des Angriffsverkehrs (konstant oder variabel), Auswirkungen des Angriffs auf das Opfer, Verwundbarkeit der Angriffspakete gegenüber Paketfiltern, Dynamik in der Menge der Angriffsknoten und verschiedene Opfertypen. Asosheh und Ramezani [11] erweitern obige Taxonomie noch um die Sparte Angriffsarchitektur.
Ziel dieser Arbeit ist es herkömmliche Verteidigungsmechanismen denen gegenüber zu stellen, welche auf Basis des SDN-Paradigmas arbeiten. Daher verwendet die hier vorgestellte Taxonomie nur Kategorien, welche direkt mit einem DDoS-Angriff in Verbindung gebracht werden können. Kategorien, die etwa Vor- und Nachbereitung eines DDoS-Angriffs thematisieren werden daher nicht in die hier präsentierte Taxonomie mit aufgenommen. Im Speziellen werden die in [10, 11] aufgeführten Klassen ignoriert, welche den Aufbau von Botnets und das Schadensausmaß beim Opfer kategorisieren. Dennoch orientiert sich die in dieser Arbeit vorgestellte Taxonomie (Abbildung 1) stark an den Ausführungen von Mirkovic und Reiher. Im Folgenden sollen die einzelnen Kategorien beschrieben werden.

2.1 Architektur

Diese Kategorie gliedert DDoS-Angriffe gemäß deren Infrastruktur. Angreifer, welche die *Agent-Handler*-Architektur verwenden setzen infiltrierte Netzwerkknoten entweder als *Handler* oder als *Agent* ein. Die Agents werden zumeist ohne deren Wissen missbraucht und daher auch als sekundäre Opfer bezeichnet. Aus Gründen der Sicherheit und Anonymität werden Agents indirekt über die Handler kontrolliert. Jeder Agent wird mindestens einem Handler zugeordnet und ein Handler kontrolliert mindestens einen Agent. Um nun Befehle abzusetzen (bspw. Angriff!) kommuniziert der Angreifer mit den Handlern, welche den Befehl an ihre Agents weitergeben. Die Agents führen den eigentlichen Angriff aus, indem sie die entsprechenden Pakete in Richtung des Opfers senden [12]. So setzt beispielsweise Stacheldraht die Agent-Handler-Architektur ein [8].
Aufgrund einiger negativer Eigenschaften der Agent-Handler-Variante greifen Angreifer heutzutage vermehrt zur IRC-basierten (*Internet Relay Chat*) Architektur [14]. Dort werden anstelle von Handlern IRC-Server oder ganze IRC-Netzwerke eingesetzt. Über *IRC-Channel* kommuniziert der Angreifer mit den Bots. Dieses Vorgehen hat mehrere Vorteile. Da sich Angreifer für gewöhnlich stark frequentierte IRC-Server aussuchen, gewährleisten diese automatisch Anonymität. Befehle des Angreifers unterscheiden sich kaum vom üblichen Nachrichtenverkehr auf dem Server. Agents können daher nur schwer detektiert werden [10]. Darüber hinaus liefert ein IRC-Server bereits die benötigte Infrastruktur und Protokolle welche die Kommunikation zwischen Angreifer und Agents ermöglichen [14].
Beim *Reflector*-Angriff wird eine Infrastruktur verwendet, die der Agent-Handler-Architektur sehr stark ähnelt. Auch hier hat der Angreifer die Möglichkeit, Agents indirekt über Handler zu steuern. Der entscheidende Unterschied ist, dass die Angriffspakete nicht direkt an das Opfer gesendet werden. Vielmehr schicken Agents die Pakete indirekt über Reflector-Knoten (z.B.: *DNS-Server* oder *DNS-Resolver*) an das Opfer. Um die Weiterleitung zu ermöglichen, spoofen Agenten die Quell-IP-Adresse der Angriffspakete, sprich die eigene IP-Adresse wird innerhalb der Pakete mit der des Opfers ersetzt. Die Antwortnachrichten der Reflector-Knoten werden schließlich dem Internet Protokoll entsprechend an das Opfer adressiert. Vorteil dieser Angriffsstrategie ist zum einen, dass durch das Einflechten einer weiteren Delegationsstufe die Rückverfolgbarkeit zusätzlich erschwert wird.
Eine weitere Ausprägung des Reflector-Angriffs missbraucht bestimmte Protokolleigenschaften, wodurch die Antwortnachricht wesentlich größer ausfallen kann, als die vorangegangene Anfrage. Der Quotient (die Datengröße der Antwortnachricht dividiert durch die Datengröße der Anfragenachricht) wird auch als *Amplifikationsfaktor* bezeichnet. Ein hoher Amplifikationsfaktor hat zur Folge, dass mit einer entsprechenden Anfragemenge ein wesentlich größeres Echo generiert werden kann. Während das Netzwerk des Angreifers die geringe Anfragemasse verkraftet, wird das Netzwerk des Opfers mit künstlich aufgeblasenen Antworten konfrontiert und häufig überlastet [15].

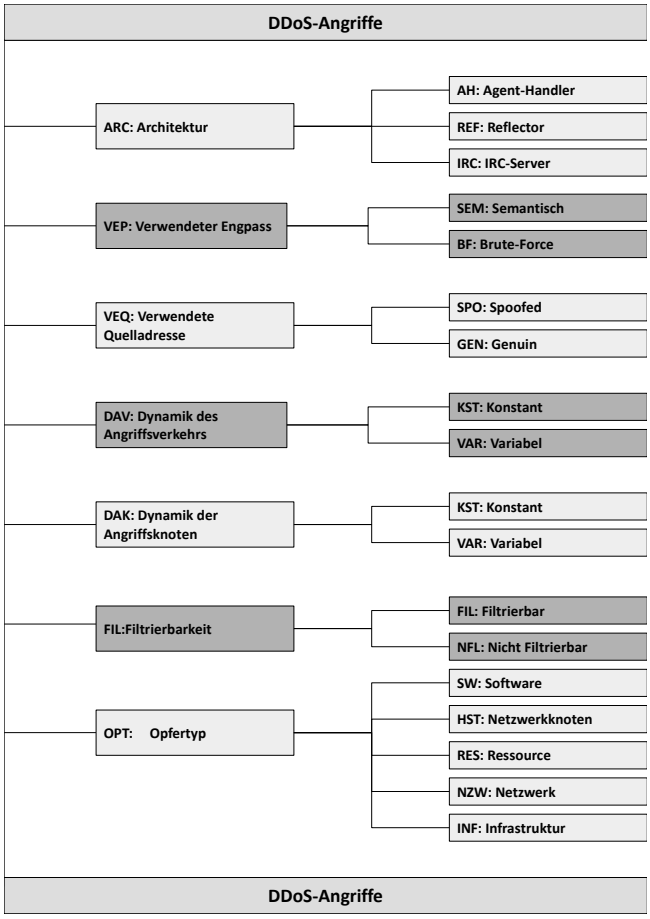

Abbildung 1: Taxonomie DDoS-Angriffe

2.2 Verwendeter Engpass

Bei *semantischen* Angriffen [10] werden gewisse Protokolleigenschaften oder Schwachstellen bzw. Fehler in der Software des Opfers missbraucht um gezielt dessen Ressourcen zu überlasten. In [12, 13] werden sie daher auch als Ressourcenerschöpfungs-Angriffe bezeichnet. Ein prominen-

doi: 10.2313/NET-2015-09-1_08

ter Vertreter dieser Klasse ist der *TCP-SYN*-Angriff. Dabei wird der drei-Wege-Handshake von TCP missbraucht. Jede TCP-Verbindungsanfrage (SYN Nachricht) verschlingt Speicherplatz. Eine große Menge von SYN Paketen führt zur Überlastung des Speichers.
Das Pendant zu semantischen Angriffen sind *Brute-Force*[10]- oder Bandbreitenerschöpfungs-Angriffe [12, 13]. Hierbei sendet der Angreifer eine große Menge unverfälschter Pakete (mit Ausnahme der Quell-IP-Adresse) in Richtung des Opfers um dessen Netzwerk zu überlasten. Zu dieser Kategorie zählt beispielsweise der oben erklärte Reflector-Angriff. Häufig werden aber auch simple UDP- oder ICMP-Pakete verwendet um das Opfernetzwerk zu fluten. Die Diskrepanz zwischen Brute-Force-Angriffen und semantischen Angriffen ist gering. Als wichtiger Unterschied ist hervorzuheben, das letztgenannte durch Protokoll- bzw. Softwarenachbesserungen abgewehrt oder zumindest geschwächt werden können. Semantische Angriffe, deren missbrauchte Lücke geschlossen wurde fallen automatisch in die Brute-Force Klasse. Abschließend ist noch festzuhalten, dass semantische Angriffe durch den Missbrauch von Schwachstellen ein wesentlich geringeres Datenvolumen benötigen als Brute-Force-Angriffe, um vergleichbare Effekte zu erzielen.

2.3 Verwendete Quelladresse

Es existieren die beiden Möglichkeiten Angriffspakete entweder mit der genuinen oder einer gespooften Quelladresse abzuschicken. Die Vorteile für gefälschte Quelladressen liegen klar auf der Hand. Zum einen kann ein Agent vom Opfer nicht durch die Quell-IP-Adresse in den Angriffs-Paketen identifiziert werden. Außerdem kann ein einziger Agent verschiedene Quell-Adressen für den Angriff verwenden, was Paketfiltern die Arbeit erheblich erschwert. Einige Angriffsmuster, wie etwa der Reflector-Angriff, würden ohne IP-Adress-Spoofing gar nicht funktionieren. Originale IP-Quelladressen werden nur eingesetzt, wenn das Spoofen derselben nicht möglich ist. Gründe hierfür sind etwa die Notwendigkeit von Administratorrechten bzw. mangelhafte Unterstützung durch die Betriebssystemebene [10].

2.4 Dynamik des Angriffsverkehrs

Die meisten DDoS-Angriffe erzeugen konstante Verkehrsraten. Der aktuelle Trend geht in Richtung länger andauernder Angriffe mit geringerer Bandbreite [5]. Die starke und plötzliche Abweichung vom regulären Datenvolumen und die Konstanz im Datenfluss ermöglicht es Verteidigern schnell zu reagieren, sprich Angreifer zu detektieren und Paketfilter entsprechend anzupassen. Daher werden viele Angriffe mit einer variablen Datenrate ausgeführt. Ansteigende oder schwankende Volumina sind mögliche Verfahrensweisen [10]. Ansari und Shevtekar [16] beschreiben den (aus ihrer Sicht) perfekten Angriff, welcher konstante und variable Datenraten kombiniert.

2.5 Dynamik der Angriffsknoten

Bei DDoS-Angriffen werden Botnets verwendet, die aus einer großen Menge verschiedener Netzwerkknoten bestehen. Zum Angriffszeitpunkt kann eine Teilmenge von Bots zeitgleich oder verschiedene Teilmengen zeitversetzt aktiviert oder deaktiviert werden. Ständig wechselnde Angreifergruppen erschweren die Rückverfolgung und Detektion der Angreifer zusätzlich [10].

2.6 Filtrierbarkeit

Viele Angriffstypen verwenden Pakete deren äußeres Erscheinungsbild von regulären Paketen abweicht (s. semantische Angriffe). Wenn eine solche Abnormität detektiert wird, können Filterregeln erstellt werden, die speziell auf die entsprechenden Pakete zugeschnitten sind. Diese Angriffe sind demzufolge durch Paketfilter abwehrbar. Im Gegensatz dazu existieren DDoS-Angriffe, die unverfälschte Datenpakete (mit Ausnahme der Quell-IP-Adresse) benutzen (s. Brute-Force-Angriffe). Obwohl diese Teil eines Angriffs sind, können sie von regulären Anfragen legitimer Servicenutzer kaum unterschieden werden und sind damit nur schwer aussiebbar [10].

2.7 Opfertypen

Wie bereits beschrieben existieren Angriffe, die speziell gegen bestimmte Softwareanwendungen gerichtet sind. Ziele könnten etwa ein Webserver oder eine DNS-Anwendung sein. Je nach Ressourcenzuteilung durch das Betriebssystems wird das angegriffene Gerät vollständig überlastet. Obwohl die Auswirkungen in diesem Fall zwar weitreichender sind, kann der Angriff gleichzeitig schneller detektiert werden. Sind der angegriffenen Anwendung lediglich Anteile der gesamten Systemressourcen zugewiesen, können parallel laufende Anwendung ungestört weiter arbeiten.
Eine weitere Klasse von DDoS-Angriffen haben die Überlastung des gesamten Netzwerkknotens zum Ziel. Auch hier ist der TCP-SYN-Angriff ein repräsentatives Beispiel. Ressourcenangriffe gehen gegen Knoten, die einen für das Netzwerk essentiellen Dienst anbieten. Wird bspw. der *Gateway Router* oder lokale DNS-Server außer Gefecht gesetzt, so wird das gesamte Netzwerk gelähmt oder sogar handlungsunfähig gemacht. Da solche Netzwerkknoten in der Regel gut geschützt (z. B. durch Replikation) sind oder über leistungsstarke Systemkomponenten verfügen, benötigt der Angreifer hier einen starken DDoS-Angriff. Diese lassen sich meist einfacher detektieren. Netzwerkangriffe sind ebenfalls gegen ein gesamtes Netzwerk gerichtet. Der dabei erzeugte Datenverkehr überlastet das Netzwerk (bzw. die Bandbreite der Internetleitung), was den Transport legitimer Nachrichten verhindert. Der dafür erforderliche Angriffsverkehr kann auf Grund des großen Datenvolumens einfacher detektiert werden.
Die letzte Unterkategorie der Opfertypen betrifft die gesamte Infrastruktur des Internets. Ähnlich wie bei Ressourcenangriffen im kleinen Rahmen attackiert der Angreifer essentielle Dienste und Systeme des Internets. Dazu gehören unter anderem *Backbone-Router* oder Root-DNS-Server [10].

3. AKTUELLE DDOS-ANGRIFFE UND DEREN EINORDNUNG IN DIE TAXONOMIE

Wie bereits in der Einleitung beschrieben, existiert die Problematik von DDoS-Angriffen nach wie vor. Aktuelle Beispiele belegen dies. Doch auch in der Vergangenheit wurden bereits zahlreiche DDoS-Angriffe ausgeführt. Eine Zuordnung der Angriffe zu der Taxonomie wird in Tabelle 1 gezeigt.

DNS: So waren Oktober 2002 die Root-DNS-Server Ziel eines solchen Attentats (Herkunft unbekannt). Mit einer *ICMP-Ping*-Flut wurden 9 der 13 DNS-Root-Server kurzzeitig lahm gelegt. Eine Messung ergab eine Datenrate von circa 80 Mb-

doi: 10.2313/NET-2015-09-1_08

Tabelle 1: Zuordnung der Angriffe in die DDoS-Taxonomie

Ziel	ARC	VEP	VEQ	DAV	DAK	FIL	OPT
DNS	-	BF	-	KST	-	FIL	INF
SCI	-	BF	GEN	VAR	KST	FIL	HST
SPM	REF	BF	SPO	VAR	KST	FIL	NZW
CLF	REF	BF	SPO	VAR	KST	FIL	NZW
PSN	IRC	SEM	SPO	-	-	NFL	HST
GIT	-	SEM	GEN	KST	VAR	NFL	SW

ARC:Architektur **VEP**:Engpass **VEQ**:Quelladresse
DAV:Angriffsverkehr **DAK**:Angriffsknoten
FIL:Filtrierbarkeit **OPT**:Opfertyp

ps. Der Angriff dauerte in etwa eine Stunde, wobei der normale Internetnutzer davon kaum tangiert wurde. Um den Angriff abzuschwächen wurden vorgeschaltete Paketfilter instruiert IMCP-Pakete zu blocken [17, 31].
SCI: Im Januar 2008 war *Scientology's* Internetauftritt Ziel eines DDoS-Angriffs. Zu der Tat bekannten sich *Anonymous*-Aktivisten. Ausgeführt wurde die Tat mittels mehrerer Angriffswellen. Hierfür wurde das DDoS-Tool LOIC (*Low Orbit Ion Cannon*) verwendet, welches UDP- und TCP-Pakete generiert. Mit Spitzenwerten von 220 Mbps hat es Anonymous geschafft den Dienst der Webseite zu blockieren. Seitens Scientology sind keine nennenswerten Abwehrmaßnahmen bekannt [32, 33, 34]. Da die Datenrate selbst für einen DDoS-Angriff dieser Zeit nicht überdurchschnittlich hoch war, und LOIC-Angriffe mittels einfacher Firewall-Regeln abzuwehren sind [35], ist anzunehmen, dass Scientology keine Abwehrmaßnahmen bereitstellte. Als Reaktion auf den Angriff nahm Scientology die DDoS-Abwehrdienste von *Prolexic Technologies* in Anspruch.
SPM: Der nächste vorgestellte DDoS Angriff spielt größentechnisch in einer weit höheren Liga. Im März 2013 wurde die Webseite der anti-spam Organisation *Spamhaus* von einem DDoS-Angriff (Herkunft unbekannt) erschüttert und die Webseite unerreichbar gemacht. Die Angreifer benutzten mehrere verschiedene Angriffsvektoren. Der mit Abstand größte Datenverkehr wurde mit einem DNS-Reflector-Angriff erzeugt. Hierfür missbrauchten die Täter über 30000 offene *DNS Resolver*. Ein weiterer Bestandteil der Angriffsmasse wurde mit *ACK-Reflector*-Angriffen generiert. Auch hier sind keine Verteidigungsmaßnahmen seitens Spamhaus bekannt. Diese wären jedoch ohnehin unwirksam gewesen, da der Angriffsverkehr mit Spitzenwerten bis zu 75 Gbps die Internetleitung der Webseite komplett überlastete. Einen Tag später kontaktierten die Spamhausbetreiber den CDN-Anbieter (*Content Delivery Network*) *Cloudflare*. Dieses Unternehmen verfügt über 23 Datenzentren die weltweit verteilt sind. Mithilfe der Adressierungsart *Anycast* kann der Datenverkehr auf die verschiedene Datenzentren verteilt werden, die wiederum Duplikate der Webseite speichern. So wird der Angriff zwar nicht abgeschwächt, aber die Last auf verschiedene Netzwerkknoten verteilt. Durch die Unterstützung von Cloudflare war der Internetauftritt von Spamhaus kurz darauf auch wieder erreichbar [36]. Eine detaillierte Analyse des Angriffs kann in [37] gesichtet werden.
CLF: Fast genau ein Jahr später berichteten diverse Quellen von einem DDoS-Angriff der alle zuvor erlebten Angriffe in den Schatten stellte. Ziel des Angriffs war ein Cloudflare Kunde. Die Täter verwendeten einen NTP-Reflector-Angriff (*Network Time Protocol*). Hierbei wird in der Anfrage der *MONLIST* Befehl verwendet. Der NTP-Server Antwortet mit einem Paket, das die IP-Adressen der letzten 600 Anfragen enthält. Der Amplifikationsfaktor beträgt dabei 206. Mit dieser Vorgehensweise schafften es die Angreifer einen Datenverkehr von über 400 Gbps zu generieren. Dafür wurden 4529 verschiedene NTP-Server benutzt. Cloudflare konnte den Angriff mit den oben beschriebenen Methoden erfolgreich verteidigen [38].
PSN: Im Dezember des selben Jahres gelang es der Gruppe *Lizard Squad* das PSN (*PlayStation Network*) mehrere Stunden lahmzulegen [40]. Über den genauen Tathergang ist nach bestem Wissen des Autors nichts bekannt. Aller Wahrscheinlichkeit nach wurde ein *TCP-Flag-DDoS*-Angriff ausgeführt [39]. Quellen zu Datenraten oder zu etwaigen Verteidigungsmaßnahmen Seitens Sony's konnten nicht gefunden werden. Dennoch ist grob bekannt, wie Lizard Squad generell operiert. Die Gruppe verwendet ein Botnet, das aus mit *Malware* infizierten Routern besteht. Diese können schließlich für einen DDoS-Angriff instrumentalisiert werden [41].
GIT: Der letzte hier vorgestellte DDoS-Angriff betraf den Hosting-Dienst *GitHub*. Der Angriff stammte aus China. Dabei wurde Internetnutzern, die bestimmte Webseiten aufriefen, bösartiger Javascript Code gesendet. Das Skript fordert den Browser in einer Endlosschleife auf, zwei spezielle Internetseiten der GitHub-Domäne aufzurufen. Quellen zu Datenraten oder zu etwaigen Verteidigungsmaßnahmen sind auch hier nicht bekannt [42].

4. KONVENTIONELLE VERTEIDIGUNGSMAßNAHMEN

Bei DDoS-Verteidigungsmaßnahmen verhält es sich ähnlich wie bei DDoS-Angriffsvektoren: Es existiert eine kaum zu überblickende Vielfalt verschiedener Ansätzen, Ideen und tatsächlichen Implementierungen. Um dieses Kapitel übersichtlich zu gestalten werden die Verteidigungsansätze in drei verschiedene Kategorien unterteilt. Genauer gesagt wird hierbei nur die Standortzugehörigkeit berücksichtigt. Es existieren drei Möglichkeiten eine DDoS-Verteidigung zu platzieren:

1) Auf der Netzwerkseite des Opfers
2) Im Kernnetzwerk, welches Angreifer und Opfer verbindet
3) Im Quellnetzwerk des DDoS-Angriffs

Im Folgenden sollen für jede Kategorie Verteidigungsmaßnahmen vorgestellt werden.

4.1 Verteidigungsmaßnahmen im Netzwerk des Opfers

Hier angesetzte Verteidigungsmechanismen sind der geballten Wucht von DDoS-Angriffen vollständig ausgeliefert. Anstelle von präventiven Maßnahmen bleibt nur die Reaktion. Wie das Beispiel von Spamhaus [36] zeigt, ist die Verteidigung zu diesem Zeitpunkt oft bereits zu spät. Dennoch existieren Ansätze, die DDoS-Angriffe detektieren und verteidigen können.
Wang et al. [43] stellen in ihrer Arbeit eine Methode vor, die DDoS-Angriffe mit gefälschten IP-Adressen erkennt: *Hop Count Filtering* (HCF). Der Ansatz verwendet das *TTL*-Feld im Header von IP-Paketen. Dabei wird die Tatsache benutzt, dass zwischen Angreifer und Opfer und gespoofter IP-Adresse und Opfer meist unterschiedlich viele Netzwerk-

knoten (*Hops*) liegen. Sprich, viele gespoofte IP-Packete besitzen einen TTL-Wert der nicht zur gefälschten Adresse passt (Betriebssysteme verwenden Stan-dard-Werte für den initialen TTL-Wert). HCF erstellt während einer Lernphase eine *IP-zu-Hop-Count* (IP2HC) Tabelle, in der Netzwerkpräfixe bereits eingegangener IP-Pakete mit den korrespondierenden *HopCounts* gespeichert werden. Wenn ein Angriff detektiert wird (etwa über die stark angestiegene Datenrate) wechselt HTF in den Filtermodus und verwirft mit Hilfe der IP2HC-Tabelle gespoofte Pakete. Experimente zeigen, dass der HCF-Algorithmus gespoofte Pakete mit einer Wahrscheinlichkeit von ca. 90% erkennt. Voraussetzung ist eine ausgiebige Lernphase. Etwa 10% der Pakete werden als *False-Positives* erkannt.
Eine weitere Strategie ist die Verwendung von *SYN-Cookies*. Diese Methodik wird im RFC 4987 präsentiert [44] und bezieht sich auf einen Artikel von Bernstein [45]. SYN-Cookies können als Verteidigungsmaßnahme gegen SYN-Angriffe eingesetzt werden. Die Cookies werden in das *Sequenznummernfeld* von TCP-Paketen gespeichert. Da (bis auf die Länge von 32 Bit) keine konkreten Vorgaben für die initiale Sequenznummer existieren, kann dieses Feld auch dazu verwendet werden Verbindungsinformationen wie etwa Quell--IP-Adresse, Quell-Port-Nummer, Ziel-IP-Adresse und Ziel-Port-Nummer zu speichern. Da diese Informationen zu groß sind, werden Sie mittels einer kryptographischen *Hash*-Funktion auf die gewünschte Länge abgebildet. Diese Verbindungsinformationen müssten sonst in einer Tabelle verwaltet werden. Der Server antwortet(SYNACK) mit dieser speziell gewählten Sequenznummer. Da die Sequenznummer der folgenden Nachricht nur um 1 größer ist (Ausnahme: *Piggyback ACKs*), kann der Server das SYN-Cookie verifizieren. Das Wegfallen der Verbindungstabelle verhindert Speicherüberläufe und schwächt daher die Auswirkungen von SYN-Angriffen stark ab.

4.2 Verteidigung im Kernnetzwerk

DefCom ist eine Verteidigungsstrategie die von Oikonomou et al. [47] entwickelt wurde. Dabei verwenden sie bereits etablierte Mechanismen der DDoS-Detektion und -Verteidigung. Das Verfahren operiert nicht ausschließlich mit Knoten des Kernnetzwerks (aber größtenteils), sondern setzt auch Entitäten des Quell- und Opfernetzwerks ein. Alle beteiligten Netzwerkknoten bilden ein dynamisches *Overlay*-Netzwerk. Den Knoten werden verschiedene Rollen zugeteilt: 1) *Classifier*-Knoten werden dazu eingesetzt Angriffspakete zu detektieren und zu markieren. 2) *Rate-Limiter*-Knoten die Angriffe abschwächen (z.B.: Verwerfen von Paketen) 3) *Alarmgeber*-Knoten detektieren DDoS-Angriffe und propagieren diese im DefCom-Netzwerk.
Wobei 1) und 2) ausschließlich im Kernnetzwerk implementiert werden können, ist die Installation von Alarmgebern auch im Opfer- bzw. Quellnetzwerk möglich. Wenn nun ein DDoS-Angriff detektiert wird, aktiviert der Alarmgeber die beiden anderen Knotentypen (1 und 2). Aktive Classifier markieren Angriffspakete, die dann von Rate-Limitern entsprechend behandelt (gemäß der eingesetzten Policy) werden.

4.3 Verteidigung im Quellnetzwerk

Das Quellnetzwerk ist theoretisch der effektivste Ansatzpunkt für eine DDoS-Verteidigung, da Angriffspakete schon gefiltert werden können, bevor sie überhaupt ins Netz gelangen.
RFC 2827[48] spezifiziert Richtlinien für den Einsatz von Filterregeln um Angriffe mit gespooften IP-Adressen zu verhindern. Die wichtigste Maßnahme ist das Blockieren von ausgehenden Paketen, deren Quell-IP-Adresse nicht aus dem Quell-Subnetz stammt.
D-WARD [49] ist ein Paketfilter, der auf dem Gateway-Router des Quellnetzwerks installiert wird. Dieser überwacht eingehenden und ausgehenden Verkehr und speichert Statistiken (Anzahl gesendeter Pakete, Anzahl gesendeter Bytes, etc.) zu den *Flows*. Anomalien (z.B.: Zu hohe Senderate von Paketen mit einer IP-Adresse als Ziel) in den Flows werden detektiert und eingestuft. Flows die zu einer als Angriff eingestuften Anomalie gehören werden blockiert (bzw. die Verkehrsrate wird limitiert). Mit dieser Vorgehensweise ist D-WARD in der Lage ausgehende TCP-, UDP- und ICMP-Angriffe zu blockieren. Sowohl in durchgeführten Experimenten als auch unter realen Bedingungen erzielt D-WARD sehr gute Ergebnisse.

5. VERTEIDIGUNGSMECHANISMEN AUS DER WELT DES SDN

Heutzutage etablierte Netzwerkgeräte werden mit einem vorinstallierten Software-Satz ausgeliefert, der wenig Konfigurationsspielraum für den Endkunden lässt. Von außen betrachtet sind diese Geräte autonome Blackboxen, die Forwarding- und Kontrolllogik in sich vereinigen und mithilfe weniger Stellschrauben an die eigenen Bedürfnisse angepasst werden können. Diese Herangehensweise ist in vielerlei Hinsicht problematisch. Zum einen ist die Schnittstellenlandschaft sehr vielseitig. Jeder Hersteller verwendet proprietäre Software für seine Netzwerkgeräte. Daher existiert keine einheitliche Schnittstellensprache, mit deren Hilfe die Geräte konfiguriert werden können.
Des weiteren, um die Netzwerkgeräte zu konfigurieren steht Administratoren nur ein vorgefertigter Befehlssatz zur Verfügung. Inzwischen gibt es Lösungen die das Ausführen von Scripten erlauben [18]. Nichtsdestotrotz müssen auch diese in einer systemnahen Programmiersprache angefertigt werden. Das Umsetzen abstrakter Aufgabenstellungen (wie bspw. Netzwerk-Policies) ist daher ein schwieriger und fehleranfälliger Prozess [20]. Die Vereinigung von Kontroll- und Forwardinglogik in einem geschlossenen System bringt weitere Herausforderungen mit sich. Obwohl Netzwerkgeräte mithilfe von Routingprotokollen Informationen austauschen, entscheidet die Kontrollschicht jedes Geräts individuell über die Verfahrensweise mit eingehenden Paketen. Dadurch gestaltet sich das Umsetzen von Änderungen auf globaler Ebene schwierig. Man spricht daher auch von der *Verknöcherung* [19] des Internets.
Wie bereits erwähnt tauschen Netzwerkgeräte Routinginformationen um möglichst optimale Routingentscheidungen auf Kontrollebene zu treffen. Diese lassen sich allerdings nur fällen, wenn jeder Router eine globale Netzwerksicht besitzt. Netzwerke sind allerdings hochdynamische Systeme deren Stellschrauben sich ständig verändern. Solche Änderungen werden durch Routingprotokolle propagiert. Auf globaler Ebene skaliert diese Verfahrensweise allerdings nur langsam.
SDN löst viele dieser Probleme durch die Trennung von Forwarding- und Kontroll-Ebene. SDN-Switches werden lediglich für die Weiterleitung von Paketen eingesetzt und sind über wohldefinierte Schnittstellen zugreifbar. Die Kon-

trolleinheit befindet sich physisch getrennt von den Switches auf einem anderen Netzwerkgerät. Über diese zentrale Instanz werden alle Switches gesteuert. Genauer gesagt bietet eine programmierbare Schnittstelle der Kontrolleinheit Nutzern die Möglichkeit, das konkrete Verhalten von Switches zu manipulieren. Die Schnittstelle wird von einem Netzwerk-Betriebssystem bereitgestellt. Für dieses Betriebssystem können Anwendungen erstellt werden, welche die Steuerung des gesamten Netzwerks auf einem sehr abstraktem Niveau erlauben [20]. Es existieren bereits konkrete Implementierungen für Netzwerk-Betriebssysteme. Die bekannteste Ausprägung ist *NOX* [21]. *OpenFlow* [19] ist der de facto Standard für das Protokoll das die Kommunikation zwischen Kontrollgerät und Switch reglementiert.

5.1 DDoS-Detektion im Kernnetzwerk mithilfe von Self Organizing Maps (SOM)

Rodrigo, Edjard und Passito [22] präsentieren eine Methode zur effizienten DDoS-Detektion im Kernnetzwerk. Dabei verwenden sie eine NOX-Kontrolleinheit und OpenFlow-Switches. Letztere verwalten Statistiken jedes aktiven Flows: 1) Die Anzahl der empfangenen Pakete 2) Die Menge der empfangenen Daten (in Bytes) 3) Die Zeitdauer die sich der Flow bereits in der Flow-Tabelle befindet. NOX fordert in regelmäßigen Abständen die Tabellendaten aller Switches an. Somit befinden sich die Flow-Information des gesamten Netzwerks an einer zentralen Stelle. Ein *Feature Extractor* erkennt schließlich Merkmale, welche auf einen DDoS-Angriff hindeuten können. Diese Merkmale werden mithilfe von SOM [23] ausgewertet und kategorisiert. Durchgeführte Experimente zeigen mit ca. 99% eine hohe Zuverlässigkeit bei der Detektion von DDoS-Angriffen (TCP/SYN-Flood, UDP-Flood, ICMP-Flood). Dabei liegt die Rate von False-Positives unter 1%.

5.2 DDoS-Detektion im Quellnetzwerk mithilfe von Frequent Sets Analyzern (FSA)

Mehdi et al. [24] beleuchten den Einfluss des SDN-Paradigmas auf die Netzwerksicherheit im Allgemeinen. In ihrer Arbeit beschreiben sie zudem ein DDoS-Detection System. Anders als beim vorangehenden Ansatz dient die hier vorgestellte SDN-Anwendung zur Detektion von DDoS-Angriffen im Quellnetzwerk. Dafür wird die Idee von FSA herangezogen. FSA ist eine Methode des Datamining zur Detektion von Anomalien. In der Arbeit wird FSA zur Detektion von verdächtigen Verkehrsflüssen verwendet. Essentielle Voraussetzung ist die Möglichkeit des Zugriffs auf Flow-Informationen des gesamten Netzwerks. Genau das leistet SDN. Switches senden ihre Flow-Tabellen an die Kontrollinstanz. Diese analysiert die empfangenen Informationen und filtert *Frequent-Data-Sets* heraus. Das sind Teilmengen der Flow-Tabellen die auf DDoS-Verkehrsflüsse hindeuten. In der Arbeit werden keine konkreten DDoS-Angriffe genannt oder Experimente durchgeführt, welche die Effizienz der vorgestellten Methode belegen.

5.3 DDoS-Detektion und Verteidigung in Echtzeit im Kernnetzwerk

Krishnan und Durrani präsentieren in [25, 26] ein System, das DDoS-Angriffe in Echtzeit erkennt und verteidigt. Dabei konzentrieren sie sich auf lang andauernde Angriffe mit großen Datenraten (*Long-Lived Large Flows* bzw. LLL-Flows), die Protokolle der Transport- und Netzwerkschicht missbrauchen. Unter Verwendung der *sflow*-Technologie [27] senden SDN-Switches Flow-Informationen an *inMon sFlow-RT*-Kontrolleinheiten [28]. Aus den Flow-Informationen werden Metriken generiert, die eine Einstufung der Flows zulassen. Eine Anwendung auf dem SDN-Steuergerät bestimmt Regeln für den Umgang mit verdächtigen Flows (z.B.: *Drop, RateLimit*) und schreibt die Regeln mittels OpenFlow in die Flow-Tabellen der SDN-Switches. Das System wird mit einem NTP-Reflector-Angriff getestet und ist in der Lage, den Angriff binnen weniger Sekunden zu detektieren. Die Angriffs-Pakete werden von den SDN-Switches im Kernnetzwerk verworfen und gelangen gar nicht erst bis zum Netzwerk des Opfers.

5.4 DDoS-Detektion im Netzwerk des Opfers und Verteidigung im Kernnetzwerk

Saray et al. beschreiben in [29] ein System bei dem ein ISP (*Internet Service Provider*)und dessen Endkunden eng zusammen arbeiten. Beide Netzwerke setzen SDN-Steuergeräte ein. Die Switches des ISP-Netzwerks, die eine direkte Verbindung mit dem Kunden-Netzwerks besitzen (Egress-Switches) füttern die Kontrolleinheit des Kunden ständig mit neuen Informationen aus deren Flow-Tabellen. Mit diesen Informationen kann das Steuergerät des Kunden verdächtige Informationen detektieren (etwa unter Verwendung von [30]). Entdeckungen dieser Art berichtet die Kontrolleinheit des Kunden and das Steuergerät des ISP. Dadurch, dass die Angriffs-Pakete über das ISP-Netzwerk zu den Kunden gelangen, existieren in den Flow-Tabellen aller involvierter Switches Tabelleneinträge, welche die Angriffs-Flows repräsentieren. Als Konsequenz auf die Meldung des Kunden markiert das Steuergerät des ISP die verdächtigen Flows. Pakete die zu einem so markierten Flow gehören werden vom ISP genauer untersucht. Ein entsprechendes Modul siebt Pakete die zu einem DDoS-Angriff gehören aus und leitet legitime Pakete zurück in das Kernnetzwerk.

6. EIN VERGLEICH HERKÖMMLICHER UND SDN-BASIERTER VERTEIDIGUNG

Die Diskussion soll gemäß der im 4. Kapitel vorgenommenen Kategorisierung unterteilt werden.

6.1 Verteidigung im Quellnetzwerk

Generell gilt: Je näher das Verteidigungssystem an der Quelle des Angriffs sitzt, desto besser kann ein DDoS-Angriff auch verteidigt werden. Im Extremfall sind bereits Verteidigungsanwendungen auf dem Netzwerkgerät installiert, das die Pakete entsendet. Wahrscheinlicher ist allerdings, dass solche Anwendungen im Knotenpunkt angebracht sind, der das interne Netzwerk mit dem Internet verbindet. Dazu wurden im Punkt 4.3 bereits etablierte Verteidigungsstrategien vorgestellt, namentlich Egress-Filter und D-WARD. Die in Kapitel 5.2 vorgestellte SDN-Variante verfährt ähnlich. SDN-Switches im Quellnetzwerk verwalten Flow-Tabellen die sie in regelmäßigen Abständen Nachrichten an die Kontrolleinheit senden. Dadurch besitzt das Steuergerät eine globale Sicht über das Quellnetzwerkes. Allerdings sitzt der bei D-WARD verwendete Router an der Schnittstelle zum Internet und leitet alle Pakete des internen Netzwerks weiter. Damit ist die globale Sicht auch hier gegeben. Die Detek-

doi: 10.2313/NET-2015-09-1_08

tionsalgorithmen und Verteidigungsmaßnahmen sind dann rein theoretisch austauschbar. Sprich D-WARD könnte auch FSAs zur Detektion von Anomalien verwenden während der SDN-Controller Egress-Filtering einsetzen könnte.
Anhand dieses Beispiels kann gefolgert werden, dass das SDN-Paradigma bei einer Verteidigung auf der Opferseite seinen größten Vorteil nicht ausspielen kann. Herkömmliche Verteidigungsstrategien verfügen ebenfalls über eine globale Netzwerksicht. Daher bringt das SDN-Paradigma hier keine bahnbrechenden Neuerungen. Ein Vorteil von SDN könnte allerdings die einfache Portierbarkeit der Verteidigungsanwendungen sein. Aufgrund standardisierter Protokolle (OpenFlow) und Schnittstellen (NOX) könnte eine solche Anwendung unabhängig von Geräteherstellern auf verschiedenen Systemen installiert werden.

6.2 Verteidigung im Opfernetzwerk

Werden Angriffe gegen die Bandbreite gefahren, ist die Verteidigung auf der Opferseite oft schon zu spät (s. Spamhaus [36]). Lässt man diesen Angriffsvektor außer Acht, existiert noch eine weitere große Gefahr: IP-Spoofing. Bei Angriffen dieser Art steht das Opfer vor einem Dilemma. Zum einen möchte ein Serviceanbieter seine Kunden bedienen, zum anderen sollen Pakete mit gespoofter IP-Adresse geblockt werden. Die Detektion von Angriffen mit gespoofter IP-Adressen ist normalerweise nicht die Schwierigkeit. Diese kündigen sich in Form eines plötzlichen, massiven Anstiegs der Datenrate von selbst an. Die Problematik ist viel mehr: Wie können Pakete mit gespooften IP-Adressen von legitimen Paketen realer Kunden unterschieden werden? HCF (in 4.1) stellt hierfür einen Lösungsansatz parat. Der Algorithmus verwendet Unstimmigkeiten bei den HopCounts eingehender Pakete zur Detektion von Spoofing-Angriffen. Der Ansatz weist allerdings einige Schwachstellen auf. Beispielsweise könnten Angreifer IP-Adressen innerhalb des selben Subnetzes verwenden. Der HopCount wäre bei solchen Paketen stimmig. Mit SYN-Cookies werden gespoofte Pakete zwar nicht detektiert, aber die negativen Auswirkung von SYN-Angriffen (welche oft im Zusammenhang mit gespooften IP-Adressen verwendet werden) auf das Opfer abgeschwächt.
Im Opfernetzwerk allein kann das SDN-Paradigma nicht viel zur Verbesserung der momentanen Situation beitragen. Ähnlich wie bereits im Quellnetzwerk liefert das SDN-Paradigma auch hier nicht den Vorteil der globalen Netzwerksicht. Gegen Brute-Force-Angriffe bzw. IP-Spoofing kann die neue Technologie auf der Opferseite nichts ausrichten. Diese Angriffe müssen bereits vorher (im Kernnetzwerk oder bei der Quelle) vereitelt werden.
In Kapitel 5.5 wird ein hybrider Ansatz (mit Verteidigungskomponenten im Opfer- und Kernnetzwerk) beschrieben. Egress Switches füttern das Steuergerät des Kunden kontinuierlich mit Flow-Informationen. Mit diesen Informationen kann der Kunde verdächtige Flows detektieren und dem ISP melden. Der ISP filtert die verdächtigen Pakete und schwächt den Angriff damit ab. Dieser Ansatz funktioniert bei Angriffen mit "wenigen"(z.B.: 10000) Quellen, die jeweils große Datenmengen generieren. Ein NTP-Reflection-Angriff passt in das Schema. Damit lassen sich also Angriffe gegen die Bandbreite des Opfers verhindern. Allerdings ist nur schwer vorstellbar, wie dieses Verteidigungssystem SYN-Angriffe mit gespooften IP-Adressen verhindern soll. Schließlich erzeugen diese Millionen verschiedene Flows.

6.3 Verteidigung im Kernnetzwerk

In 4.2 wird DefCom beschrieben, welches ein Overlay-Verteidigungs-Netzwerk aus Knoten des Quell-, Kern- und Opfernetzwerks erstellt. Dies erfordert zum einen die Zusammenarbeit von Kunden mit ihrem ISP, was nicht vorausgesetzt werden kann. Zweitens müssten Netzwerkrouter mit neuer Funktionalität versehen werden. Zu guter Letzt werden neue Protokolle benötigt, welche die Interaktion der beteiligten Parteien definieren. Soviele Neuerungen zu etablieren ist in einem Netzwerk mit größtenteils vertikal integrierten Routern schwierig.
Die Gründe hierfür werden weitestgehend im 5. Kapitel erläutert: Für jede Routersoftware werden individuelle Lösungen benötigt. Neue Sicherheitsprotokolle müssen entwickelt und eingebettet werden. Durch die hochverteilte Infrastruktur sind Änderungen auf globaler Ebene nur mühsam umsetzbar. Selbst wenn DefCom oder ähnliche Strategien in die Realität umgesetzt werden könnten, enstehen dadurch weitere Probleme. Neue Technologien sind für gewöhnlich verletzlich (z.B.: DNSSEC [50]). Um deren Widerstandsfähigkeit zu stärken müssen Wartungs- und Nachbesserungsarbeiten durchgeführt werden. Dieser Entwicklungsprozess findet im heutigen, starren Internet nur sehr langsam statt. Sind die Verteidigungsstrategien veraltet, müssten unter größten Anstrengungen neue Konzepte implementiert werden. Diese Beschreibung lässt erahnen, weshalb eine grundlegende Änderung der konventionellen Architektur vorteilhaft sein könnte.
Im 5. Kapitel werden SDN-Verteidigungssysteme vorgestellt, die im Kernnetzwerk operieren. Lässt man deren Feinheiten außer Acht und betrachtet stattdessen die grundlegende Architektur und Funktionsweise, so erkennt man die Ähnlichkeit der Ansätze. SDN-Switches im Kernnetzwerk senden Statusberichte an einen zentralen Controler. Dieser detektiert mithilfe installierter Softwareanwendungen Anomalien. Entsprechend der zugrundeliegenden Policies sendet die Kontrolleinheit Befehle an die Switches, wonach verdächtige Flows blockiert oder eingeschränkt werden können. So eine Architektur kann durch Alarmgeber auf der Seite des Opfernetzwerks unterstützt werden.
Unter einem noch abstrakteren Blickwinkel existiert also ein zentraler Befehlsgeber, welcher eine globale Netzwerksicht besitzt und mithilfe von Softwareanwendungen den Netzwerkfluss analysieren und steuern kann. Diese Architektur ist sehr flexibel, da Detektions- und Steuerungsalgorithmen durch einfache Softwareupdates verbessert und gewartet werden können. Auch ist die Installation neuer Verteidigungsstrategien ist wesentlich unkomplizierter. Die entsprechende Anwendung kann in einer hohen Programmiersprache entwickelt und auf dem betreffenden Controler installiert werden. Hier werden keine Individuallösungen benötigt, welche auf jedem Router installiert werden müssen.

7. SDN-CONTROLER - SINGLE POINT OF FAILURE

Neben vielen bereits beschriebenen Vorteilen des SDN-Paradigma existieren natürlich auch Nachteile. Das größte Sicherheitsrisiko, das Experten bei der Einführung von SDN sehen, ist das Problem des *Single Point of Failure*. Bisher waren Aufgaben (wie etwa das Routing) über viele Netzwerkknoten hinweg verteilt. Redundanz und Vielseitigkeit machen Netzwerke wie das Internet zwar schwerer Kontrol-

lierbar aber auch robuster gegenüber Angriffen. Schließlich hat es noch kein Angreifer seit bestehen des Internets geschafft, dieses auch nur ansatzweise außer Kraft zu setzen. SDN-Controller auf der anderen Seite steuern Netzwerke von einem einzigen Punkt aus. Dies wäre ein ideales Angriffsziel um großen Schaden anzurichten. Denkbar wären etwa DDoS-Angriffe, welche die Handlungsunfähigkeit des Controllers bewirken. Da SDNs bisher noch nicht in der Öffentlichkeit eingesetzt werden, konnte die konkrete Verwundbarkeit des Controlers nicht festgestellt werden. Die Gefahr eines SSingle Point of Failureßollte bei der Planung und Entwicklung von SDNs stets berücksichtigt werden. Die Installation von Redundanz durch Controler-Replikate könnte beispielsweise die Widerstandsfähigkeit erhöhen.

8. FAZIT

In dieser Arbeit wurden SDN-basierte Verteidigungsansätze und konventionelle Lösungen untersucht und miteinander verglichen. Dabei zeigen sich die Stärken und Schwächen beider Verteidigungswelten. Sowohl auf der Opferseite, als auch im Quellnetzwerk bietet das SDN-Paradigma keine maßgeblichen Vorteile gegenüber der konventionellen Verteidigungsvariante. Anders verhält es sich im Kernnetzwerk. Dort besticht das SDN-Paradigma durch seine Flexibilität. Die heutige, starre Struktur des Internets lässt Erweiterungen kaum zu. Natürlich können auch SDNs nicht von dem einen Tag auf den anderen installiert werden. Der flächendeckende Einsatz dieses neuen Netzwerkparadigmas stellt vor allem für ISPs eine gewaltige Aufgabe dar. Dennoch zeigen aktuelle Beispiele ([51, 52]) die Umsetzbarkeit und deren Vorteile. Der ISP AT&T plant bis zum Jahr 2020 für 75% seiner Netzwerke das SDN-Paradigma einzusetzen [53]. Es bleibt abzuwarten wie sich SDN-basierte DDoS-Verteidigung in einer realen Umgebung schlägt. Erste Experimente zeigen bereits vielversprechende Resultate.

9. LITERATUR

[1] *PlayStation Network, Xbox Live hit by DDOS attacks*, Abgerufen: 21.05.2015, `http://www.gamesindustry.biz/articles/2014-12-29-playstation-network-xbox-live-hit-by-ddos-attacks`,2015

[2] *Hackerattacke auf Merkel und Regierung*, Abgerufen: 21.05.2015, `https://www.tagesschau.de/inland/kanzlerin-bundestag-101.html`,2015

[3] *Large DDoS Botnet Powered by Routers Infected With "Spike" Malware*, Abgerufen: 21.05.2015, `http://www.securityweek.com/large-ddos-botnet-powered-routers-infected-%E2%80%9Cspike%E2%80%9D-malware`,2015

[4] *Arbor Networks Detects Largest Ever DDoS Attack in Q1 2015 DDoS Report*, Abgerufen: 21.05.2015, `http://www.arbornetworks.com/news-and-events/press-releases/recent-press-releases/5405-arbor-networks-records-largest-ever-ddos-attack-in-q1-2015-ddos-report`,2015

[5] Akamai: *Q1 2015 State of the Internet — Security Report*, 2015

[6] Akamai: *Q4 2014 State of the Internet — Security Report*, 2014

[7] *DDoS-Malware auf Linux-Servern entdeckt*, Abgerufen: 26.05.2015, `http://www.golem.de/news/botnetze-ddos-malware-auf-linux-servern-entdeckt-1409-109028.html`,2014

[8] David Dittrich: *The "stacheldraht"distributed denial of service attack tool*,1999

[9] Saman Taghavi Zargar et al.: *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, Communications Surveys & Tutorials, IEEE 15.4 (2013): 2046-2069.

[10] Jelena Mirkovic, Peter Reiher: *A Taxonomy of DDoS Attack and DDoS Defense Mechanisms*, ACM SIGCOMM Computer Communication Review 34.2 (2004): 39-53.

[11] Dr. Abbass Asosheh, Naghmeh Ramezani: *A Comprehensive Taxonomy of DDoS Attacks and Defense Mechanism Applying in a Smart Classification*, WSEAS Transactions on Computers 7.7 (2008): 281-290.

[12] Stephen M. Specht, Ruby B. Lee: *Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures*, ISCA PDCS. 2004.

[13] Christos Douligeris, Aikaterini Mitrokotsa : *DDoS Attacks and Defense Mechanisms: A Classification*, 3rd IEEE International Symposium on Signal Processing & Information Technology,2003

[14] David Dittrich, Sven Dietrech : *command and control structures in malware*, Usenix magazine 32.6,2007

[15] Christian Rossow : *Amplification Hell: Revisiting Network Protocols for DDoS Abuse*, Symposium on Network and Distributed System Security (NDSS),2014

[16] Nirwan Ansari, Amey Shevtekar : *On the new Breed of Denial of Service (DOS) Attacks in the Internet* ,2004

[17] Susan Branowski: *How Secure are the Root DNS Servers?*, Version 1,2003

[18] Cisco Systems, Inc.: *Cisco IOS Scripting with TCL Configuration Guide*,2014

[19] Nick McKeown et al. : *OpenFlow: Enabling Innovation in Campus Networks*, ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[20] Bruno Nunes et al.: *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*, Communications Surveys & Tutorials, IEEE 16.3 (2014): 1617-1634.

[21] Natasha Gude et al.: *NOX: Towards an Operating System for Networks*, ACM SIGCOMM Computer Communication Review 38.3 (2008): 105-110.

[22] Rodrigo Braga et al.: *Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow*, Local Computer Networks (LCN), 2010 IEEE 35th Conference on. IEEE, 2010.

[23] T. Kohonen: *The self-organizing map*, Proceedings of the IEEE 78.9 (1990): 1464-1480.

[24] Syed Akbar Mehdi et al.: *SDN Architecture Impact on Network Security*,2011

[25] Ramki Krishnan, Muhammad Durrani: *Real Time SDN and NFV Analytics for DDoS Mitigation*,2014

[26] Ramki Krishnan: *Real Time SDN and NFV Analytics for DDoS Mitigation*, Abgerufen: 11.06.2015, `https://youtu.be/8JBU88dsyks`, 2014

[27] sflow.org : *sflow*, Abgerufen: 11.06.2015, http://www.sflow.org,2015

[28] InMon Corp. sFlow: *sFlow-RT*, Abgerufen: 11.06.2015, http://www.inmon.com/products/sFlow-RT.php,2015

[29] Rishikesh Sahay et al.: *Towards Autonomic DDoS Mitigation using Software Defined Networking*,2015

[30] K. Giotis et al.: *Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments*, Computer Networks 62 (2014): 122-136.

[31] Ryan Naraine: *Massive DDoS Attack Hit DNS Root Servers*, Abgerufen: 11.06.2015, http://www.internetnews.com/dev-news/article.php/1486981/, 2002

[32] Jose Nazario: *Church of Scientology DDoS Statistics*, Abgerufen: 11.06.2015, https://asert.arbornetworks.com/church-of-scientology-ddos-statistics/, 2008

[33] Quinn Norton: *Church of Scientology DDoS Statistics*, Abgerufen: 11.06.2015, http://www.wired.com/2011/12/anonymous-101-part-deux/3/, 2011

[34] PC World: *Hackers Hit Scientology With Online Attack*, Abgerufen: 11.06.2015, http://www.washingtonpost.com/wp-dyn/content/article/2008/01/25/AR2008012503399.html, 2008

[35] Ryan Barnett: *LOIC DDoS Analysis and Detection*, Abgerufen: 11.06.2015, https://www.trustwave.com/Resources/SpiderLabs-Blog/LOIC-DDoS-Analysis-and-Detection/,2011

[36] Matthew Prince: *The DDoS That Knocked Spamhaus Offline (And How We Mitigated It)*, Abgerufen: 11.06.2015, https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/,2013

[37] NSFOCUS: *Analysis of DDoS Attacks on Spamhaus and recommended solution*, 2013

[38] Matthew Prince: *Technical Details Behind a 400Gbps NTP Amplification DDoS Attack*, Abgerufen: 11.06.2015, https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/,2014

[39] Bill Brenner: *TCP Flag DDoS Attack by Lizard Squad Indicates DDoS Tool Development*, Abgerufen: 11.06.2015, https://blogs.akamai.com/2015/01/tcp-flag-ddos-attack-by-lizard-squad-indicates-ddos-tool-development.html,2015

[40] Brian Krebs: *Cowards Attack Sony PlayStation, Microsoft xBox Networks*, Abgerufen: 11.06.2015, http://krebsonsecurity.com/2014/12/cowards-attack-sony-playstation-microsoft-xbox-networks/comment-page-1/,2014

[41] Brian Krebs: *Lizard Stresser Runs on Hacked Home Routers*, Abgerufen: 11.06.2015, http://krebsonsecurity.com/2015/01/lizard-stresser-runs-on-hacked-home-routers/,2015

[42] Erik Hjelmvik: *China's Man-on-the-Side Attack on GitHub*, Abgerufen: 11.06.2015, http://www.netresec.com/?page=Blog&month=2015-03&post=China%27s-Man-on-the-Side-Attack-on-GitHub,2015

[43] Haining Wang et al.: *Defense Against Spoofed IP Traffic Using Hop-Count Filtering*, IEEE/ACM Transactions on Networking (ToN) 15.1 (2007): 40-53.

[44] W. Eddy: *TCP SYN Flooding Attacks and Common Mitigations*, RFC 4987,2007

[45] D. J. Bernstein: *SYN cookies*, Abgerufen 11.06.2015, http://cr.yp.to/syncookies.html,1997

[46] Information Sciences Institute: *TRANSMISSION CONTROL PROTOCOL*, RFC 793,1981

[47] George Oikonomou et al.: *A Framework for A Collaborative DDoS Defense*, Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual. IEEE,2006.

[48] P. Ferguson: *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, RFC 2827,2000

[49] J. Mirkovic, P. Reiher: *D-WARD: a source-end defense against flooding denial-of-service attacks*, Dependable and Secure Computing, IEEE Transactions on 2.3 (2005): 216-232.

[50] Roland van Rijswijk-Deij et al.: *DNSSEC and Its Potential for DDoS Attacks*, Proceedings of the 2014 Conference on Internet Measurement Conference. ACM,2014

[51] Marcia Savage: *Google's Infrastructure Chief Talks SDN*, Abgerufen: 08.07.2015, http://www.networkcomputing.com/data-centers/googles-infrastructure-chief-talks-sdn/d/d-id/1320352,2015

[52] Jonathan Vanian: *Facebook shows the promise of SDN with new networking tech*, Abgerufen: 08.07.2015, https://gigaom.com/2014/11/14/facebook-shows-the-promise-of-sdn-with-new-networking-tech/,2014

[53] J Dan Meyer: *Software, NFV, SDN focus bolsters workforce opportunities*, Abgerufen: 08.07.2015,http://www.rcrwireless.com/20141216/telecom-software/att-targets-75-virtualization-software-control-of-network-by-2020-tag2,2014

What is "Privacy"? - Information theory

Samuel Hall
Betreuer: Marcel von Maltitz
Seminar Innovative Internet-Technologien und Mobilkommunikation SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: sammy.hall@tum.de

KURZFASSUNG

Wenn man versucht eine Definition für Datenschutz zu finden, stößt man sehr schnell auf das Problem, dass es sehr viele verschiedene Definition gibt, die sich zum einen stark voneinander unterscheiden und zum anderen oft sehr weich formuliert sind. In dieser Seminararbeit werden deshalb sowohl verschieden Metriken vorgestellt, die sich an verschiedene Definitionen von Datenschutz und seinen Teilaspekte anlehnen, als auch auf die Probleme eingegangen, die dabei entstehen. Ziel soll sein Datenschutz verschiedener Systeme mit Hilfe von diesen Metriken abzuprüfen. Dabei soll vor allem der informations-theoretische Ansatz im Vordergrund stehen.

Schlüsselworte

Privacy, Privatsphäre, Datenschutz, Metrik, Informationstheorie, anonymity, unlinkability

1. EINLEITUNG

Um *"What is privacy?"* beantworten zu können, muss zunächst für den Begriff *privacy* eine korrekte Übersetzung ins Deutsche gefunden werden. Vergleicht man verschiedene Übersetzungen [1][2], stellt man fest, dass sich *privacy*, sowohl zu Privatsphäre, als auch zu Datenschutz übersetzen lässt. Die zwei Begriffe sind aber keinesfalls als Synonyme zu verstehen. Datenschutz lässt sich auf vielfältige Weise definieren. Der Duden definiert Datenschutz folgendermaßen:

> Schutz des Bürgers vor Beeinträchtigungen seiner Privatsphäre durch unbefugte Erhebung, Speicherung und Weitergabe von Daten, die seine Person betreffen. [3]

Datenschutz ist hier also als Schutz der Privatsphäre definiert. Privatsphäre selbst lässt sich aus Artikel 12 der Allgemeinen Erklärung der Menschenrechte [4] ableiten und ist deshalb auch im deutschem Recht in Form verschiedener Gesetze, wie dem Recht auf Post- und Fernmeldegeheimnis oder das Recht auf die Unverletzlichkeit der Wohnung im Grundgesetz [5], verankert. Durch diese Gesetze und ihrer tiefen Verankerung wird deutlich wie wichtig Privatsphäre für die Gesellschaft ist. Datenschutz trägt maßgeblich zum Erhalt der Privatsphäre bei. Datenschutz selbst wird im Grundgesetz allerdings nicht explizit erwähnt. Wenn man sich die Duden Definition anschaut, könnte man auch sagen, dass Datenschutz den technischen Aspekt des Schutzes der Privatsphäre abdeckt. Juristisch wird Datenschutz, als der Schutz personenbezogener Daten vor missbräuchlicher Verwendung bezeichnet [6]. Um Datenschutz zu quantifizieren, messbar und vergleichbar zu machen und letztendlich dann um Fragen wie "Wie gut hält System XY Datenschutz ein?" beantworten zu können, reicht diese Definition allerdings nicht aus. Um das trotzdem zu können, werden jeweils einzelne Aspekte von *privacy* betrachtet und Metriken darauf angewendet. Metriken bilden mit Hilfe mathematischer Funktionen, Eigenschaften eines Systems auf einen Zahlenwert ab. In dieser Arbeit werden bestehende Metriken vorgestellt, die genau das möglichen machen. Dabei wird zwischen Metriken unterschieden, die sich allgemein anwenden lassen und Metriken, die nur für bestimmten Anwendungen funktionieren. Zunächst wird die rechtliche Situation anhand des Bundesdatenschutzgesetzes in Kapitel 2 erläutert. Besonderes soll gezeigt werden welche Vorgaben zu Datenschutz das Gesetz vorgibt, um dann vergleichen zu können, inwiefern die vorgestellten Metriken dazu beitragen diese Vorgaben zu erfüllen. In Kapitel 3.1 wird eine Metrik vorgestellt, die misst wie gut Informationen über Standorte in *vehicle-to-infrastructure (V2X)* Systemen geschützt werden. In Kapitel 3.2 werden drei verschieden Metriken gezeigt, die dazu dienen den Grad von Anonymität von zu veröffentlichenden Mikrodaten, also Daten, die zu statistischen Zwecken über Individuen erhoben wurden, zu bestimmen. Dabei liegt der Fokus, darauf welche Angriffe die Datensätze, die mit Hilfe der Metriken anonymisiert wurden, verhindern und welche potentielle Probleme nach der Anonymisierung bestehen. In Kapitel 3.2.4 wird die Definition der zuvor vorgestellten Metriken um einen informationstheoretischen Ansatz erweitert und ein Zusammenhang zwischen den Metriken in Kapitel 3.2 hergestellt. In Kapitel 4 geht es dann darum Metriken zu finden, die möglichst allgemein anwendbar sind.

2. RECHTLICHE SITUATION

Der Datenschutz ist in Deutschland seit 1977 gesetzlich im Bundesdatenschutzgesetz (BDSG) [12] geregelt. Unabhängig von der Anwendung schreibt das BDSG in § 3a Datenvermeidung und Datensparsamkeit vor. Dies bedeutet, dass nur so wenig personenbezogenen Daten wie es für die Anwendung nötig ist, zu erheben sind. Ist es trotzdem nötig personenbezogene Daten zu nutzen, so sind diese zu anonymisieren. Dies wird auch nochmals explizit in § 30, §30a und § 40, die die "[g]eschäftsmäßige Datenerhebung und -speicherung zum Zweck der Übermittlung in anonymisierter Form" , "Markt- oder Meinungsforschung" und "Verarbeitung und Nutzung personenbezogener Daten durch Forschungseinrichtungen"

doi: 10.2313/NET-2015-09-1_09

regeln erwähnt. In allen drei Paragraphen wird vom Gesetz verlangt, dass die Datenbestände anonymisiert werden, sobald es "nach dem Zweck des Forschungsvorhabens bzw. nach dem Forschungszweck möglich ist". Der Begriff *Anonymisierung* wird in § 3 (6) definiert:

> Anonymisieren ist das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können.

Interessant hierbei ist, dass die einzige Aussage, wie weit die Daten anonymisiert werden müssen, ist, dass sie nur mit unverhältnismäßig großem Aufwand Personen zugeordnet werden können. Das kann unter Umständen relativ frei interpretiert werden. Es liegt also in der Hand der Gerichte, basierend auf aktuelle technische Gegebenheiten, zu entscheiden ob die Anonymisierung ausreichend ist. In den Kommentaren und Erläuterungen zu § 3 Absatz 6 wird allerdings näher erläutert wie die Anonymisierung zu erfolgen hat [13]. Hierbei sind durchaus Parallelen zu den Methoden erkennbar, die im folgenden Kapitel vorgestellt werden. So wird verlangt, dass *identifier* gelöscht werden. Es werden auch explizit *identity disclosure* und *attribute disclosure* erwähnt und dass diese durch verallgemeinerte *quasi identifiers* verhindert werden können. *Quasi identifier* sind Attribute die in Kombination ein Individuum identifizieren können. Auf die Begrifflichkeiten wird in Kapitel 3.2 näher eingegangen. Die genauen Hintergründe hierzu werden ebenfalls in diesem Kapitel erläutert. Zusätzlich zu den dort vorgestellten Methoden wird in dem Gesetzkommentar auch empfohlen Zufallsfehler, quasi ein Rauschen, den Daten hinzuzufügen. Aber auch hier fehlen Angaben, wie wie weit die Verallgemeinerung gehen muss.

3. ANWENDUNGSPEZIFISCHE METRIKEN

3.1 Location privacy metric in V2X

Vehicle-to-infrastructure (V2X) bezeichnet man ein Kommunikationssystem, bei dem Fahrzeuge Informationen mit verkehrstechnischer Infrastruktur austauschen. Je nach Anwendung kann es sein, dass der aktuelle Aufenthaltsort, der als besonders schützenswert gilt, beispielsweise für Funktionen, wie der Kollisionsvermeidung, ziemlich detailliert preisgegeben werden muss. Dieses preisgeben des Standortes kann als Verletzung der Privatsphäre gesehen werden. Wie detailliert die Informationen über den Standort einer Person preisgegeben werden, kann in dieser Metrik festgehalten werden. Dazu sind einige Modellierungen nötig. Wir betrachten Fahrten, die aus einer Serie von Aufenthaltsorten bestehen. Gemessen wird die Möglichkeit eines Angreifers eine Fahrt einer bestimmten Person zuzuordnen. Zum einen modellieren wir einen Graphen, der drei verschiedene Arten von Knoten besitzt. Individuen I, Ursprung O und Ziel D. Wir nehmen an, dass es gleich viele Ursprünge und Ziele gibt. Die Kanten sind mit einer Wahrscheinlichkeit p gewichtet. Zusätzlich zu dem Graphen lässt sich das Modell auch mit drei Adjazenzmatrizen IO, OD und DI repräsentieren. Die Einträge in IO stellen die Wahrscheinlichkeiten dar ein Individuum einem Ursprung zuzuordnen, in OD sind die Einträge die Wahrscheinlichkeiten für Fahrten zwischen O und D. In DI sind entsprechend die Wahrscheinlichkeiten ein Individuum einem Ziel zuzuordnen. Die Summen der Zeilen in OD und DI müssen genau 1 ergeben. In IO darf die Summe einer Zeile auch kleiner 1 sein. Die verbleibende Differenz zu 1 ist die Wahrscheinlichkeit, dass das in der entsprechenden Zeile repräsentierte Individuum keine Fahrt antritt und daheim bleibt. Diese Wahrscheinlichkeit wird als p^c bezeichnet. Abbildung 1 (a) zeigt beispielhaft den Graph der auf ein Individuum zentriert wurde. In Abbildung (b) wurde der Graph dahingehend vereinfacht, dass nur noch das Produkt aus den Einzelwahrscheinlichkeiten angezeigt wird.

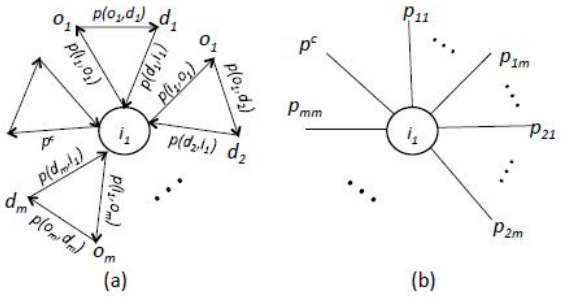

Abbildung 1: Beispielgraph [7]

Um die Informationen die in den Wahrscheinlichkeiten stecken messbar zu machen, bedient man sich der Entropie. Entropie ist ein Maß um anzugeben wie viel Unsicherheit eine Verteilung enthält. Je höher die Unsicherheit ist, desto höher ist die Entropie. Eine hohe Entropie ist im Sinne des Datenschutzes erstrebenswert. Die Entropie H ist folgendermaßen definiert:

$$H = -\sum p_i log_b p_i \tag{1}$$

Wird der Logarithmus zur Basis b=2 genommen, ist die Einheit der Entropie *bit*. Die Entropie $H(i_s)$ für ein Individuum i_s wird wie folgt definiert:

$$H(i_s) = -(\sum_{j=1}^{m}\sum_{k=1}^{m} \hat{p}_{jk} log(\hat{p}_{jk}) + \hat{p}^c log(\hat{p}^c)) \tag{2}$$

Dabei sind $\hat{p}_{jk}$ und $\hat{p}^c$ die normalisierten Wahrscheinlichkeiten, dass i_s eine Fahrt macht bzw. keine Fahrt macht.

$$\hat{p}_{jk} = \frac{p(i_s, o_j)p(o_j, d_k)p(d_k, i_s)}{\sum_{j=1}^{m}\sum_{k=1}^{m} p(i_s, o_j)p(o_j, d_k)p(d_k, i_s) + \hat{p}^c} \tag{3}$$

$$\hat{p}^c = 1 - \sum_{j=1}^{m} p(i_s, o_j) \tag{4}$$

Um den Wert der hier berechnet wird besser einordnen zu können, wird zusätzlich die maximal mögliche Entropie $MaxH$ berechnet und in Relation zu der für ein Individuum berechneten Entropie gesetzt:

$$H_{\%} = \frac{H(i_s)}{MaxH(i_s)} 100\% \tag{5}$$

$$MaxH(i_s) = -log(\frac{1}{m^2+1}) \tag{6}$$

doi: 10.2313/NET-2015-09-1_09

$H_{\%}$ gibt somit an wie weit der Datenschutz für ein Individuum vom maximal möglichen Datenschutz entfernt ist. Diese Definition besitzt einige Parallelen zu den Definition von *unlinkability* bzw. *anonymity*, die im Kapitel 4 vorgestellt werden. Dies liegt allerdings weniger daran, dass man diese Metrik auf einen allgemeinen Fall zurückführen kann, sondern dass die Metrik in diesem Kapitel auf diesem Prinzip aufbaut. Man kann eher davon sprechen, dass die *location privacy metric* ein Spezialfall der *unlinkability* Metrik ist, der entsprechend den Eigenheiten, die $V2X$ mit sich bringt, angepasst wurde.

3.2 Metriken zur Anonymisierung von Mikrodaten

Der Anwendungsfall, dass Datensätze anonym veröffentlicht werden sollen, die sensible Daten enthalten, ist der vermutlich am besten erforschte Bereich, was *privacy*-Metriken angeht. So gibt es in der Konsequenz bereits einige verschiedene Metriken, die ausdrücken wie stark die Daten anonymisiert sind und die Anhaltspunkte geben, ob die Daten weiter anonymisiert werden müssen, um die Privatsphäre der einzelnen Personen zu gewährleisten und wie hoch das Risiko einer Deanonymisierung ist.

Nach [10] können die Daten aus den zu veröffentlichenden Datensätzen in drei Kategorien eingeteilt werden. Zum einen gibt es Attribute, die die Individuen eindeutig identifizieren können. Beispielsweise die Passnummer, die Sozialversicherungsnummer oder auch der vollständige Name. Diese Attribute werden *Identifiers* genannt. Attribute die in Kombination, aber nicht alleine, ein Individuum identifizieren können werden *quasi-identifiers (QI)* genannt. Das können z.B. Alter, Geschlecht, usw... sein. Attribute die sensitive Daten über Individuen, z.B. Krankheitsbefunde, enthalten, werden *sensitive attributes* genannt. Ein Datensatz kann mehrere verschiedene *sensitive attribute* enthalten. Der Einfachheit halber, gehen wir in dieser Arbeit davon aus, dass der Datensatz nur eines enthält. Betrachten wir nun als Beispiel *Tabelle 1*, welches aus [10] entnommen ist und für unsere Bedürfnisse angepasst wurde. Dann ist dort die Diagnose W das *sensitive attribute* und Namen und Höhe X sind jeweils *QI*. Dieser Datensatz soll nun veröffentlicht werden. Dazu muss der Datensatz anonymisiert werden, um die Privatsphäre der Patienten zu schützen. Hierfür werden in Kapitel 3.2.1, Kapitel 3.2.2 und Kapitel 3.2.3 jeweils verschiedene Metriken herangezogen.

3.2.1 k-anonymity

Die k-anonymity Metrik ist die am wenigsten strenge Metrik. Ziel dieser Metrik ist es so genannte *linking attacks*, bei denen der Angreifer versucht durch die Kombination der veröffentlichten *quasi-identifiers* Rückschlüsse auf die Identität der Personen zu erlangen. Die Metrik besagt, dass für jede Kombination von *quasi-identifier* es mindestens k Einträge in der Tabelle gibt. Jeder Eintrag ist dann einer Person mit einer Wahrscheinlichkeit von $1/k$ zuordenbar. In Bezug auf Tabelle 1 würde diese Vorgabe, der k-Anonymität, bedeuten, dass es für jede Kombination aus Namen und Größe X mindestens k Einträge geben muss. Um dies zu erreichen wird in der anonymisierten Tabelle mit $k = 2$ der Name komplett zensiert und statt der Größe X werden nur noch Bereiche die jeweils 10cm umfassen angegeben. Die Daten wurden also

Original Dataset {*X, W*}		
Name	Height X	Diagnosis W
Timothy	166	N
Alice	163	N
Perry	161	N
Tom	167	N
Ron	175	N
Omer	170	Y
Bob	170	N
Amber	171	N
Sonya	181	N
Leslie	183	N
Erin	195	Y
John	191	Y

Tabelle 1: Original Datensatz [10]

generalisiert. Tabelleneinträge, die dieselben Werte für die *quasi-identifier* besitzen, gehören einer Äquivalenzklasse an. Die Äquivalenzklassen werden auch einfach Blöcke genannt. Wäre der Datensatz größer, hätte man eventuell auch den Anfangsbuchstaben oder noch mehr von den Namen nicht zensieren müssen oder auch die Bereiche der Größe kleiner fassen können. Der Umfang der Anonymisierungen hängt also nicht nur vom gewählten k ab, sondern auch von der Menge der Daten im Datensatz, sowie ihrer konkreten Ausprägung.

Anonymized Dataset {X, W}		
Name	Height $\tilde{X}$	Diagnosis W
******	[160-170]	N
******		N
******		N
******		N
******	[170-180]	N
******		Y
******		N
******		N
******	[180-190]	N
******		N
******	[190-200]	Y
******		Y

Tabelle 2: Anonymisierter Datensatz [10]

Man erkennt schnell, dass der Informationsgehalt mit steigendem k verloren geht. So kann es sein, dass unter Umständen die Daten in ihrem ursprünglichen Verwendungszweck gar nicht mehr verwertbar sind.

Grundsätzlich gibt es zwei verschiedene Risiken der Identifizierung, zum einen *identity disclosure*, welches auftritt, wenn ein Eintrag in der Tabelle einer bestimmten Person zugeordnet werden kann. Dieses Risiko kann mit *k-anonymity* erfolgreich ausgeschlossen werden. Für das zweite Risiko, der *attribute disclosure*, bei dem Attribute einer Person zugeordnet werden können, bedarf es weitere Metriken. Diese werden in den folgenden Kapiteln erläutert. [8]

3.2.2 l-diversity

Bei Betrachtung des letzten Blocks von Tabelle 2 fällt auf, dass das *sensitive attribute* bei allen Einträgen gleich ist.

doi: 10.2313/NET-2015-09-1_09

Dies ermöglicht *homogeneity attacks* und *background attacks*. Der Angreifer kann durch Hintergrundwissen das *sensitive attribute* einer bestimmten Person eindeutig bestimmen. In unserem Beispiel genügt es zu wissen, dass eine Person in dem veröffentlichten Datensatz erfasst ist und dass die Person größer als 190 cm ist, um zur der Erkenntnis zu gelangen, dass diese Person über einen positiven Befund verfügt. In [9] wird eine neues Prinzip, *l-diversity*, eingeführt, um dieser Art von Angriffen entgegenzuwirken. Es gibt verschiedene Möglichkeiten *l-diversity* zu definieren. grundsätzlich schreibt diese Metrik vor, dass in jedem Block die l häufigsten Werte der *sensitive attribute*, mindestens einmal, idealerweise jedoch möglichst gleich verteilt, repräsentiert werden. Ein Block erfüllt *l-diversity*, wenn mindestens l Werte vorkommen. Ein Datensatz erfüllt *l-diversity*, wenn alle Blöcke *l-diversity* erfüllen. Unser Beispiel kann maximal *2-diversity* erfüllen, da das *sensitive attribute* nur 2 Werte annehmen kann. Betrachtet man die Tabelle 2, stellt man fest, dass nicht mal *2-diversity* erfüllt ist. Nur im 2. Block kann man von *2-diversity* sprechen.

3.2.3 t-closeness

Dass *l-diversity* nicht immer zielführend ist, kann man ebenfalls wieder in unserem Beispiel erkennen. In dem Beispiel kann das *sensitive attribute* nur zwei Werte annehmen. Je nach Befund, positiv oder negativ. Zum anderen kommt der positive Befund gewöhnlich viel seltener vor, als der negative. So auch in diesem Beispiel. In Blöcke in denen kein positiver Wert vorliegt, führt das zu keinen Problemen, da man davon ausgehen kann, mit einem negativen Befund in Verbindung gebracht zu werden nicht die Privatsphäre eines Einzelne verletzt. Es kann allerdings per Ausschlussprinzip zu Verletzung der Privatsphäre eines dritten kommen. Angenommen in einem Block befinden sich gleich viele positive und negative Befunde, dann erfüllt dieser zwar *2-diversity*, aber jede Person, deren Daten in diesem Block repräsentiert werden, wird plötzlich mit einer Wahrscheinlichkeit von $p = 50\%$ mit einem positiven Befund in Verbindung gebracht, obwohl die Wahrscheinlichkeit eines solchen Befundes sehr viel geringer ist. Dieses Problem ermöglicht sogenannte *skewness attacks*. Wenn das *sensitive attribute*, anderes als in unserem Beispiel, mehrere Werte annehmen kann, so kann dies zu *similarity attacks* führen. *l-diversity* berücksichtigt nämlich nicht, ob *sensitive attributes* sich semantisch ähnlich sind. In [11] wird eine dritte Metrik, *t-closeness*, beschrieben, die sich diesen Problemen widmet. Diese Metrik basiert bereits auf einem Prinzip, auf dem auch allgemeinere Metriken basieren, wie sie in Kapitel 4 beschrieben werden. Die a posteriori Wahrscheinlichkeit ein *sensitive attribute* einer Person zuzuordnen, nachdem der Datensatz veröffentlicht wurde, darf nicht größer sein, als die a priori Wahrscheinlichkeit vor der Veröffentlichung. Das wird laut *t-closeness* erreicht, wenn sich die Verteilung des *sensitive attribute* in einem Block maximal durch einen Schwellwert t von der Verteilung des ganzen Datensatzes unterscheidet. Um diesen Unterschied in den Verteilungen zu messen hat sich die *Earth Mover's distance* als am brauchbarsten herausgestellt [11]. Diese misst den Aufwand der nötig wäre, eine Verteilung in die andere zu überführen. Bei der Festlegung von t gilt es abzuwägen, wie viel Informationsverlust man in Kauf nimmt. Denn der Grad an Datenschutz den einem t zusichert steht im direkten Zusammenhang mit dem Grad an Informationen, die verloren gehen. Das lässt sich damit begründen, dass die Informationen der veröffentlichten Datensätze genau in den Unterschieden der Verteilungen liegen. In dem Beispiel könnten Forscher beispielsweise versuchen einen Zusammenhang zwischen der Größe X und der Diagnose W herzustellen. Dieser Zusammenhang geht allerdings mit zunehmender Anonymisierung verloren.

3.2.4 One-symbol information

Da die in diesem Kapitel gezeigten Metriken auf den Schutz vor verschieden Arten von Angriffen abzielen sind sie untereinander nur schwer vergleichbar. Deshalb werden in [10] die Metriken aufgegriffen und auf eine *one-symbol information* Einheit zurückgeführt. Dazu werden *k-anonymity*, *l-diversity* und *t-closeness* mit Hilfe von Entropie und Transinformation neu definiert. Transinformation $I(X;Y)$ gibt die Stärke des statistischen Zusammenhangs zwischen den Zufallsvariablen X und Y an.

$$\begin{aligned} I(X;Y) = & \sum_{x\in X, y\in Y} p(x,y) log_2 \left[\frac{p(x,y)}{p(x)p(y)}\right] \\ = & \sum_{x\in X, y\in Y} p(y)p(x|y) log_2 \left[\frac{p(x|y)}{p(x)}\right] \\ = & H(Y) - H(Y|X) = \sum_{x\in X} p(x)\left[H(Y) - H(Y|x)\right] \end{aligned} \tag{7}$$

Bei *one-symbol information* wird zunächst nicht die ganze Tabelle betrachtet, sondern jeder Eintrag einzeln. Sei x ein Eintrag in einem Datensatz X und $\tilde{x}$ ein Eintrag in einem anonymisierten Datensatz $\tilde{X}$. Die Wahrscheinlichkeit einen Eintrag x mit einem gegebenem $\tilde{x}$ zu identifizieren beträgt $p(x|\tilde{x}) = 1/N_{\tilde{x}}$, wobei $N_{\tilde{x}}$ die Anzahl der Eintrage x ist, die das gegebene $\tilde{x}$ in X abdeckt. N ist dagegen die Anzahl der unterscheidbaren Einträge in X.

k-anonymity

Somit lässt sich *k-anonymity* folgendermaßen mit Hilfe der bedingten Entropie ausdrücken:

$$H(X|\tilde{x}) \geq log_2 k \tag{8}$$

Für *One-symbol information* gibt es vier verschiedene Definitionen I_1, I_2, I_3 und I_4, wobei hier nur die ersten zwei Definitionen verwendet werden. Als *one-symbol information* wird *k-anonymity* dann so definiert:

$$I_2(X,\tilde{x}) \equiv H(X) - H(X|\tilde{x}) \leq log_2 \frac{N}{k} \tag{9}$$

Will man nicht die einzelnen Einträge $\tilde{x}$ sondern den ganzen Datensatz $\tilde{X}$ betrachten muss man den Durchschnitt betrachten, der wie folgt definiert wird.

$$I(X,\tilde{X}) \leq log_2 \frac{N}{k} \tag{10}$$

l-diversity

l-diversity kann ebenfalls mittels Entropie ausgedrückt werden:

$$H(W|\tilde{x}) \geq log_2 l \tag{11}$$

W ist dabei das *sensitive attribute*. Als *one-symbol information* ausgedrückt lautet die Formel:

$$I_2(W,\tilde{x}) \equiv H(W) - H(W|\tilde{x}) \leq H(W) - log_2 l \tag{12}$$

doi: 10.2313/NET-2015-09-1_09

Als Durchschnitt über $\tilde{X}$:

$$I(W, \tilde{X}) \equiv H(W) - H(W|\tilde{X}) \leq H(W) - log_2 l \quad (13)$$

t-closeness

Auch *t-closeness* lässt sich als *one-symbol information* ausdrücken:

$$I_1(W, \tilde{x}) \equiv \sum_{w \in W} p(w|\tilde{x}) log_2 \frac{p(w|\tilde{x})}{p(w)} \leq t \quad (14)$$

Als Durchschnitt über $\tilde{X}$:

$$I_1(W, \tilde{X}) \equiv \sum_{\tilde{x} \in \tilde{X}} p(\tilde{x}) \sum_{w \in W} p(w|\tilde{x}) log_2 \frac{p(w|\tilde{x})}{p(w)} \leq t \quad (15)$$

Aus diesen Definition lassen sich nun die oberen und unteren Schranken für l herleiten, als auch ein Zusammenhang zwischen l und t herstellen:

$$1 \leq l \leq l_{max} \equiv 2^{H(W)} \quad (16)$$

$$l_t = 2^{H(W)-t} \quad (17)$$

So lässt sich mit l_t ein l für ein gegebenes t berechnen.

4. ALLGEMEINE METRIKEN

Da sich die Metriken aus Kapitel 3 nur jeweils für einen speziellen Anwendungsfall nutzen lassen, wird in diesem Kapitel versucht Metriken aufzustellen, die sich unabhängig davon, welcher Anwendungsfall vorliegt, immer anwenden lassen sollen. In [14] werden zwei Metriken, *degree of anonymity* und darauf basierend *degree of unlinkability* eingeführt, die genau das zum Ziel haben. In diesem Kapitel wird deshalb näher auf diese Metriken eingegangen.

4.1 Degree of anonymity

In 3.2.3 hat man a priori und a posteriori Wahrscheinlichkeiten verglichen, um zu sehen an wie viel Informationen ein Angreifer gelangen kann. Auf einem ähnlichen Prinzip basiert der *degree of anonymity*. Während bei *t-closeness*, wie bereits in Kapitel 3.2.3 erwähnt, die *Earth Mover's Distance* benutzt, welche den Aufwand beschreibt, eine Verteilung in eine andere zu überführen, werden beim *degree of anonymity* die aus der Informationstheorie bekannten Entropien verglichen. Dafür wird zunächst $A = \{a_1, \ldots, a_n\}$ als eine nicht-leere, endliche Menge von Aktionen und $U = \{u_1, \ldots, u_n\}$ als eine Menge von Benutzer definiert. Jedes $u_i \in U$ mit $i \in \{1, \ldots, n\}$ führt a mit einer Wahrscheinlichkeit $p_i > 0$ aus. Die a priori Wahrscheinlichkeit, dass u_i die Aktion a ausgeführt hat, beträgt idealerweise $1/n$. Durch Beobachtung kann der Angreifer Rückschlüsse ziehen und auf eine a posteriori Wahrscheinlichkeit, die sich von der a priori Wahrscheinlichkeit unterscheidet, schließen. Die a posteriori Wahrscheinlichkeit wird mit Hilfe der Zufallsvariable X definiert, wobei $p_i = P_a(X = u_i)$ gilt. Der *degree of anonymity* lässt sich berechnen wenn man den Unterschied zwischen der maximal möglichen Entropie $max(H(X)) = log_2(n)$ und der a posteriori Entropie $H(X) = -\sum_{i=1}^{n} p_i log_2(p_i)$ betrachtet. Weil man nicht die Größe der Menge U, sondern nur die Verteilung messen will, ist es nötig den Unterschied zusätzlich noch zu normalisieren. Der *degree of anonymity* $d(U)$ wird folglich folgendermaßen definiert:

$$d(U) := 1 - \frac{max(H(X)) - H(X)}{max(H(X))} = \frac{H(X)}{max(H(X))} \quad (18)$$

Durch die Normalisierung nimmt $d(U)$ nur Werte im Bereich $[0, 1]$ an. $d(U) = 0$ bedeutet, dass es ein Subjekt gibt, welches einer Aktion mit der Wahrscheinlichkeit $p_i = 1$ zuzuordenbar ist. $d(U) = 1$ dagegen würde bedeuten, dass jede Aktion a einem Subjekt nur mit einer Wahrscheinlichkeit von $1/n$ zuordenbar ist [14].

4.2 Degree of unlinkability

Das Konzept der Anonymität ist nicht immer ausreichend, weil es nur auf Personen anwendbar ist. Deshalb ist es sinnvoll zusätzlich den *Degree of unlinkability* zu definieren. *Unlinkability* wird folgendermaßen definiert: Zwei Elemente sind, nachdem man ein System beobachtet hat, nicht einander mehr oder auch weniger zuordenbar, als zuvor. Unterscheidet sich die a priori Wahrscheinlichkeit, Elemente einander zuordnen zu können, von der a posteriori Wahrscheinlichkeit, spricht man von einem *existential break*. Die Elemente können dabei alles mögliche sein. Zum Beispiel Personen, Nachrichten, Aktionen oder vieles mehr. Für die formale Definition wird wieder eine Menge $A = \{a_1, \ldots, a_l\}$ mit den Elementen aus dem zu betrachteten System definiert. Innerhalb dieser Menge werden die Äquivalenzklassen $A_1, \ldots, A_n$ gebildet, die jeweils Elemente enthalten, die einander verwandt sind, beispielsweise Nachrichten die vom gleichen User versendet wurden. Außerdem wird die Äquivalenzrelation $\sim_{r(A)}$ gebildet, welche bedeutet 'ist verwandt mit'. Zunächst betrachten wir den Fall, dass der *degree of unlinkability* von zwei Elementen a_i und a_j innerhalb einer Menge berechnet wird. Der *degree of unlinkability d(i,j)* zweier Elemente a_i und a_j wird wie folgt definiert:

$$\begin{aligned} d(i,j) := \quad & H(i,j) \\ = \quad & -P(a_i \sim_{r(A)} a_j) \cdot log_2(P(a_i \sim_{r(A)} a_j)) \\ & -P(a_i \not\sim_{r(A)} a_j) \cdot log_2(P(a_i \not\sim_{r(A)} a_j)) \end{aligned} \quad (19)$$

$d(i,j) = 0$ bedeutet, dass der Angreifer zu 100% a_i und a_j derselben Äquivalenzklasse zuordnen kann oder ausschließen kann, dass sie in derselben sind. $d(i,j) = 1$ dagegen bedeutet, dass der Angreifer a_i und a_j mit einer Wahrscheinlichkeit von $\frac{1}{2}$ derselben Äquivalenzklasse zuordnen kann oder eben nicht. Im nächsten Schritt wird die Definition dahingegen erweitert, dass nicht die Zuordenbarkeit von zwei, sondern beliebig vielen Elementen bestimmt werden kann. Sei $\{a_{i_1}, \ldots, a_{i_k}\}$ eine Teilmenge von A mit $2 < k \leq n$ Elementen. $P((\sim_{r_j(A)} |\{a_{i_1}, \ldots, a_{i_k}\}) = (\sim_{r(A)}))$ ist dann die Wahrscheinlichkeit, dass die betrachteten Elemente $\{a_{i_1}, \ldots, a_{i_k}\}$ den richtigen Äquivalenzklassen aus A zugeordnet wurden. Die Definition des *degree of unlinkability* $d(i_1, \ldots, i_k)$ lautet dann folglich:

$$\begin{aligned} d(i_1, \ldots, i_k) := \quad & H(i_1, \ldots, i_k) \\ = \quad & -\sum_{j \in I_k} \frac{1}{|I_k|} \\ & \cdot[P((\sim_{r_j(A)} |\{a_{i_1}, \ldots, a_{i_k}\}) = (\sim_{r(A)})) \\ & \cdot log_2(P((\sim_{r_j(A)} |\{a_{i_1}, \ldots, a_{i_k}\}) = (\sim_{r(A)})))] \end{aligned} \quad (20)$$

doi: 10.2313/NET-2015-09-1_09

Dabei ist I_k ein Index, der alle möglichen Äquivalenzklassen zählt und $|I_k| = 2^{k-1}$.

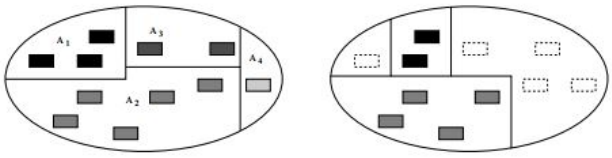

Abbildung 2: Beispiel: Teilmenge [14]

Abbildung 2 ist ein Beispiel für die zuvor beschriebene Situation. Links befindet sich die Menge A, die sich in vier Äquivalenzklassen, A_1 bis A_4, aufteilen lässt. Rechts wurde eine Teilmenge von A den Äquivalenzklassen zugeordnet.

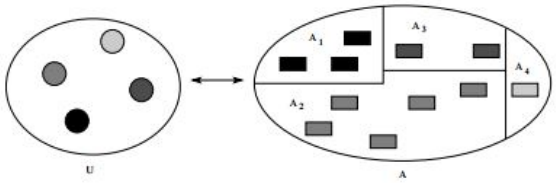

Abbildung 3: Beispiel: Mehrere Mengen [14]

Je nach Szenario, wie z.B. in Kapitel 4.1, kann es nötig sein mehr als eine Menge zu modellieren. Beispielsweise können die Benutzer in einer Menge U zusammengefasst werden und die Aktionen in einer anderen Menge A zusammengefasst werden. Abbildung 3 illustriert dieses Szenario. Die Definitionen lassen sich dahingehend problemlos anpassen, um auch diesen Fall abzudecken [14].

Die zwei Metriken lassen sich grundsätzlich bei allen Systemen anwenden. Die Herausforderung besteht dabei die zu untersuchenden Systeme korrekt zu modellieren. Da die Metriken sehr allgemein gehalten sind, können die Ergebnisse unter Umständen nicht zielführend genug sein. In diesem Fall bieten sich die Metriken aber an, um auf ihnen basierend eine spezifischere Metrik zu definieren, wie es beispielsweise auch in Kapitel 3.1 getan wurde.

5. FAZIT

Anhand von verschiedenen Metriken konnte gezeigt werden, dass gewisse Aspekte von *privacy* durch Metriken durchaus messbar sind. Die meisten Metriken zeigen auf, ob der Datenschutz in den untersuchten Systemen gewährleistet wird. Der Großteil bietet allerdings keinen Lösungsansatz, wie die Situation zu verbessern wäre. Sie dienen deshalb hauptsächlich als Indikatoren, ob es zu Problemen mit Datenschutz geben kann. Nur die in Kapitel 3.2 vorgestellten Metriken können auch dazu verwendet werden den Datenschutz, in diesem Fall speziell die Anonymität, zu verbessern. Viele der verfügbaren Metriken sind für konkrete Anwendungsfälle konzipiert. Deshalb sind die Ergebnisse aus den Metriken nur eingeschränkt untereinander vergleichbar. Es konnte aber auch gezeigt werden, dass es Metriken gibt, die grundsätzlich für alle Anwendungen anwendbar sind. Es wäre allerdings wünschenswert, dass es in diesem Bereich weitere Metriken gibt, die größere Teile von *privacy* abdecken. In den bestehenden Metriken wurde hauptsächlich der Aspekt der Anonymität und *unlinkability* beleuchtet. Gerade wenn man das Bundesdatenschutzgesetz [12] betrachtet, fällt auf, dass dort bei der Verarbeitung und Veröffentlichung der persönlichen Daten die Zustimmung hierzu eine große Rolle spielt. Dies wurde in den gezeigten Metriken gar nicht berücksichtigt. Deshalb wäre es abschließend auch wünschenswert, dass Metriken entwickelt werden, die diesen Aspekt mit aufnehmen.

6. LITERATUR

[1] *dict.cc, privacy*, `https://www.dict.cc/?s=privacy`, (Abgerufen am 17. 07. 2015)

[2] *leo.org, privacy*, `http://dict.leo.org/ende/index_en.html#/search=privacy`, (Abgerufen am 17. 07. 2015)

[3] *Duden, Datenschutz*, `http://www.duden.de/rechtschreibung/Datenschutz`, (Abgerufen am 17. 07. 2015)

[4] *Allgemeine Erklärung der Menschenrechte*, `http://www.un.org/depts/german/menschenrechte/aemr.pdf`, (Abgerufen am 17. 07. 2015)

[5] *Grundgesetz*, `http://www.gesetze-im-internet.de/gg/BJNR000010949.html#BJNR000010949BJNG0001` (Abgerufen am 17. 07. 2015)

[6] *Datenschutz: Definition, Begriff und Erklärung*, `http://www.juraforum.de/lexikon/datenschutz`, (Abgerufen am 17. 07. 2015)

[7] Ma, Z., Kargl, F. and Weber, M.: *A location privacy metric for V2X communication systems* (2009)

[8] Samarati, P. and Sweeney, L.: *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.*, Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory (1998)

[9] Machanavajjhala A., Kifer, D., Gehrke, J. and Venkitasubramaniam, M. : *l-Diversity: Privacy Beyond k-Anonymity*, ACM Trans. Knowl. Discov. Data 1, 1, Article 3 (2007)

[10] Bezzi, Michele: *An information theoretic approach for privacy metrics*, TRANSACTIONS ON DATA PRIVACY 3 (2010)

[11] Li, N., Li, T. and Venkatasubramanian, S.: *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity*, ICDE. Vol. 7. (2007)

[12] *Bundesdatenschutzgesetz*, `http://http://www.gesetze-im-internet.de/bdsg_1990/`, (Abgerufen am 17. 07. 2015)

[13] *Kommentare und Erläuterungen zu § 3 Weitere Begriffsbestimmungen*, `http://www.bfdi.bund.de/bfdi_wiki/index.php/3_BDSG_Kommentar_Absatz_6`, (Abgerufen am 17. 07. 2015)

[14] Steinbrecher S. and Köpsell, S.: *Modelling Unlinkability* (2003)

doi: 10.2313/NET-2015-09-1_09

Methods of privacy preservation

David Otter
Betreuer: Marcel von Maltitz
Innovative Internet-Technologien und Mobilkommunikation SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: otter@in.tum.de

KURZFASSUNG

Die Einhaltung von gesetzlichen wie ethischen Vorgaben bei der Gewährleistung von Privatsphäre der Nutzer stellt Softwareentwickler vor große Herausforderungen. Diese Arbeit beschäftigt sich mit der Frage, wie Privatsphäre von Seiten der Entwickler einer Software schon bei deren entstehen systematisch berücksichtigt werden kann. Dafür werden Software Pattern gesammelt, klassifiziert und erläutert wie diese, verschiedene Aspekte der Privatsphäre positiv beeinflussen.

Schlüsselworte

Software Pattern, Privacy Pattern, Privacy Enhancement

1. EINFÜHRUNG

1.1 Motivation

Der Schutz der Privatsphäre ist ein wichtiges Thema, welches überall dort von besonderer Bedeutung ist, wo große Mengen an sensitiven Daten entstehen. Das Thema ist auch in letzter Zeit wieder vermehrt in den Fokus der Öffentlichkeit gerückt und außerdem gibt es bereits sehr umfangreiche Literaturbestände zur Bedeutung von Privacy. Vor einem Problem hingegen stehen solche Softwareingenieure, welche darauf achten möchten oder müssen, dass sie Privacy by Design ermöglichen. Der Begriff Privacy by Design wurde dabei erstmals in einem Report [22] 1995 über Privacy Enhancing Technologies erwähnt und beschreibt einen Ansatz, bei dem Privatsphäre über den gesamten Lebenszyklus einer Software, mit der Softwarearchitektur beginnend und über alle Organisationsstufen hinweg, berücksichtigt wird.
Typischerweise wird in der Informatik versucht, wiederkehrende Probleme durch erprobte Muster zu lösen und Privacy ist mittlerweile zu solch einer wiederkehrenden Problemstellung geworden. Solche Design Patterns haben in der Informatik lange Tradition, weshalb es auch hierzu viele Sammlungen an solchen und Literatur dazu im Allgemeinen gibt. Bei der Suche nach Privacy Design Pattern fällt auf, dass insbesondere übersichtliche Sammlungen solcher, fehlen. Außerdem verwendet jede Arbeit ihr eigenes Schema, nach welchem die Pattern beschrieben werden und es gibt keine Arbeit bei der die Pattern nach den Design Strategies von Hoepman [1] eingeteilt werden. Dabei hat Hoepmam seine Design Strategies nach bestehendem ISO 29100 Privacy framework [28] ausgewählt, welche es erleichtern das richtige Muster für den richtigen Anwendungszweck zu finden. Deshalb sollen in dieser Arbeit verschiedene Quellen, welche sich bereits mit einzelnen Pattern beschäftigt haben gesammelt, die Pattern nach einem einheitlichen Schema beschrieben und nach angewendeter Strategie kategorisiert werden.

1.2 Privacy

Eine allgemein anerkannte Definition für Privacy zu finden ist auf Grund der Vielzahl an Aspekten, welches von diesem Konzept betroffen ist, schwierig. David H. Flaherty [12] nennt 13 Aspekte von Privacy, welche so in seiner Arbeit angeführt und auch häufig in der Literatur zitiert werden:

1. The right to individual autonomy
2. The right to be left alone
3. The right to a private life
4. **The right to control information about oneself**
5. **The right to limit accessibility**
6. The right of exclusive control of access to private realms
7. The right to minimize intrusiveness
8. **The right to expect confidentiality**
9. The right to enjoy solitude
10. The right to enjoy intimacy
11. **The right to enjoy anonymity**
12. The right to enjoy reserve
13. **The right to secrecy**

Einige dieser Aspekte, welche hier hervorgehoben wurden, können direkt auf Software angewendet werden. Wichtig an dieser Stelle ist, dass Sicherheit, gerne auch als Synonym für Privacy verwendet wird. Informationssicherheit kann über das weit verbreitete **C**onfidentiality, **I**ntegrity, and **A**vailability Triaden Modell beschrieben werden. Nun kann die Sicherheit der eigenen Daten aber nicht verhindern, dass diese Daten absichtlich an dritte Parteien weiterverkauft oder für Marketingzwecke ein komplettes, detailliertes Profil von einem selbst erstellt wird. Auf der anderen Seite kann eine hochgradig unsichere Anwendung, welche keine persönlichen Daten sammelt, nicht die Privatsphäre verletzen. Deshalb wird Sicherheit nicht im Fokus dieser Arbeit stehen.
Jaap-Henk Hoepman beschäftigt sich in seiner Forschung insbesondere mit Privacy by Design und hat auch eine Vielzahl an paper darüber verfasst. Seine Arbeit, welche auch als Grundlage der Struktur der restlichen Arbeit dient, hat es geschafft eine fundierte Ordnung in das Thema zu bringen.

1.3 Nomenklatur

Der Begriff des Pattern wurde ursprünglich von Christopher Alexander (1979) geprägt: „A pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution“. In der Softwaretechnik werden solche Entwurfsmuster für wiederkehrende Probleme in ver-

doi: 10.2313/NET-2015-09-1_10

schiedene Abstraktionsebenen unterteilt. Das Hauptwerk zu Design Patterns in der Softwaretechnik wurde dabei von der Gang of Four verfasst [29]. Das Namensschema und die Grenzen zwischen den Kategorien wird nach Hoepman [1] übernommen.
Die höchste Abstraktionsstufe Design Strategies, auch bekannt als Architeture Patterns, sind für die fundamentale strukturelle Organisation von Softwaresystemen verantwortlich. Design Patterns, wie sie nachfolgend gesammelt sind, sind Muster welche ein Subsystem, also einzelne Komponenten beschreiben. Privacy Enhancing Technologies (PETs) sind hingegen konkrete Ausprägungen eines Design Patterns, welches einen positiven Einfluss auf die Privatsphäre besitzt. Um den Unterschied zwischen einem Design Pattern und einer PET klarer zu machen, eignet sich besonders ein Beispiel von Hoepman: Idemix, ein kryptografisches Protokoll für die Privatsphäre erhaltenden Authentifizierung und den Transfer von Zertifikatattributen [23], ist die Ausarbeitung einer konkreten Technologie auf Basis des Pattern Anonymous Credentials. Aber nicht immer ist eine so klare Abgrenzung möglich.

2. DESIGN-PATTERN-VERZEICHNIS

Für die Strukturierung der Design Patterns werden die 8 Privacy Design Strategies, welche von Hoepman definiert wurden, verwendet [1]. Die Privacy Pattern wiederum werden nach folgendem Schema beschrieben:

1. Name mit Quelle
2. Kontext
3. Problem
4. Lösung
5. Beispiele
6. Ähnliche Ansätze und Querverweise

2.1 Minimise

Die einfachste Art, Privatsphäre zu gewährleisten, ist die Anzahl an gesammelten Daten zu minimieren. Ziel der Strategie ist, nur noch die für die Qualität des angebotenen Diensts notwendigen Daten zu sammeln.

2.1.1 Strip Invisible Metadata [3]

Bei der Verwendung von nutzer-generierten Dateien werden oft unsichtbare Metadaten mitgeliefert, welche vom Service nicht benötigt werden.
Die meisten Nutzer sind sich dieser Informationen, da versteckt, nicht bewusst und haben auch kein Wissen darüber, wie sie diese Metadaten entfernen können.
Solange dem Nutzer Metadaten nicht sichtbar gemacht werden können, oder diese für den Service überhaupt nicht benötigt werden, sollen sie gelöscht werden, womit die Anforderung an *Minimise* erfüllt wird.
Als Beispiel für diesen Ansatz darf Twitter gesehen werden. Beim Hochladen von Fotos auf deren Plattform werden alle Exif-Daten, ein Standard für Metainformationen in der Fotografie, von den Fotos entfernt. [4]

2.1.2 Location Granularity [13]

Viele Dienste benötigen Standortdaten, um sie selbst zu bearbeiten oder an andere Dienste weiterzuleiten.
Solche Daten werden oft durchgehend und in maximaler zur Verfügung stehender Genauigkeit gesammelt und beeinträchtigen dadurch die Privatsphäre des Nutzers.
Die Auflösung, in der Standortdaten gesammelt werden, ist oftmals für den angebotenen Service gar nicht nötig und kann deshalb heruntergesetzt werden. Damit werden die gesammelten Daten auf ein Minimum reduziert, ohne die Qualität des angebotenen Dienstes zu beeinträchtigen.
Ein Beispiel, wo dieses Pattern angewendet werden kann, ist ein Wetterdienst. Auch mit einem genauen GPS-Signal kann keine genauere Wettervorhersage geliefert werden, als mit der Postleitzahl.
Wenn die Wahl über den gewählten Detailgrad der Standortdaten von dem Nutzer selbst getroffen werden kann, wird damit auch das Pattern der *Blurred Personal Data* (2.6.4) und damit die *Control*-Strategie erfüllt.

2.2 Hide

Keine gesammelten oder übertragenen Daten sollen als Klartext vorliegen. Je nach Kontext sollen die Daten vor allen versteckt werden, um keine Musteranalyse zuzulassen, oder vor Dritten, um Vertraulichkeit gewährleisten zu können.

2.2.1 Constant Length padding [2]

In Netzwerken wird Verschlüsselung eingesetzt, um die darüber verbreiteten Daten zu schützen. Trotzdem können immer noch Metadaten, wie die Länge der Nachricht eingesehen werden.
Diese Metadaten können einerseits die Wirksamkeit der Verschlüsselung kompromittieren. Gerade in Netzwerken, wo symmetrische und asymmetrische Verschlüsselung kombiniert werden, ist dies ein Problem, da Nachrichten an jedem Knotenpunkt entschlüsselt werden und sich bei jeder Entschlüsselung die Nachrichtenlänge verringert. Andererseits gehört die Länge auch zu den sensitive Daten, welche geschützt werden sollten.
Deshalb werden alle Paketlängen uniformiert und falls notwendig, um die vorgegebene Länge zu erreichen, die Pakete mit einem Padding aufgefüllt. Die Länge als Teil der Metadaten ist also nicht mehr einfach von außen einsehbar.
Ein Beispiel für die Verwendung von *Constant Length padding* findet sich in Kombination mit der *Layered encryption*, auch bekannt als *Onion Routing* und noch anderen Techniken in Mix Networks. Das anonyme Netzwerk Tor baut auf diesen Prinzipien auf.
Eine weiteres Pattern welches sich mit Metadaten beschäftigt ist *Strip Invisible Metadata*, welches bei impliziten wie der Länge aber nicht angewendet werden kann.

2.2.2 Cover Traffic [2]

In Anonymen Netzwerken wird versucht, sich gegen alle mögliche Arten von Angriffen und Traffic Analysen zu schützen. Insbesondere Netzwerke mit geringer Latenz können auf Grund dieser Echtzeitanforderungen leichter analysiert werden, aber auch andere Mix Networks sind bestrebt die Sicherheit weiter zu erhöhen. Eine Möglichkeit, solch eine Attacke auf ein low-latency Netzwerk durchzuführen ist, indem Bandbreiteschwankungen induziert werden und dann beobachtet wird, wie diese durch das Netzwerk sickern. [24]
Durch Dummy Pakete wird der Versuch, einzelne Pakete oder Paketströme nachzuvollziehen, erschwert, oder die Art des verwendeten Protokolls maskiert. Unter anderem können Dummy Nachrichten in einem Mix Network verschickt

werden, welche bei zufälligen Knotenpunkten enden und damit die Identifizierung des Empfängers erschweren.
Ein ähnliches Pattern kann auch für Datenbanken verwendet werden. Dabei wird in Ergebnissen der Antwort von (No-)SQL Anfragen die Reihenfolge der erhaltenen Daten vertauscht, oder die Ergebnisse werden verfälscht, indem einzelne Dummy Datensätze eingefügt werden.
Cover Traffic wird unter anderem von Tarzan [14] verwendet, einem peer-to-peer-Anonymisierungsdienst.

2.2.3 Layered Encryption [2]

In Mix Networks soll Verschlüsselung eingesetzt werden, um die Daten, die darin versendet werden zu schützen.
Bei einem einfachen Verschlüsselungsansatz reicht es aus, einen einzigen Knotenpunkt zu kompromittieren, um den gesamten Verbreitungsweg nachvollziehen zu können und damit Sender und Empfänger zu identifizieren.
Die Lösung besteht in einer schichtweisen Verschlüsselung, bei der jeder Knotenpunkt nur den unmittelbar nächsten Nachbarn kennt. Als Vorbereitung auf das Versenden eines Pakets wird also zuerst eine Route festgelegt. Neben der End-to-End-Verschlüsselung haben alle Knotenpunkte und deren Nachbar ihren separaten symmetrischen Schlüssel und die Nachricht wird auf dem Weg schichtweise entschlüsselt. Die Daten sind also von außen und bis auf die nötigen Informationen über den nächsten Knotenpunkt auch innerhalb des Netzwerks geschützt.
Synonym zur *Layered Encryption* wird auch von *Onion Routing* gesprochen. Die populärste Anwendung davon, welche auch auf *Constant Length Padding* zurückgreift, ist das Netzwerk Tor.
Das Pattern erfüllt auch die Voraussetzungen der Strategie *Separate*, da die Informationen über den zurückgelegten Weg im Netzwerk in Einzelschritte zerlegt werden.

2.3 Separate

Speicherung, Transport und Verarbeitung von Daten soll möglichst verteilt sein. Besonders die Verknüpfung verschiedener Daten soll damit erschwert werden.

2.3.1 CP-ABE [25]

In verteilten Systemen sollen Daten nur von bestimmten Personen einsehbar sein. Die Gruppe der Personen kann über bestimmte Anmeldedaten oder andere Attribute festgelegt werden. Solch eine Richtlinie kann über einen vertrauenswürdigen Server kontrolliert werden, welcher die Daten speichert und den Zugang regelt.
Das Problem besteht nun darin, die Vertrauenswürdigkeit der Daten zu gewährleisten, selbst wenn ein Server, welcher die Daten gespeichert hat, nicht vertrauenswürdig ist. Eine bewährte Lösung besteht in Verschlüsselung, wobei die meisten eingesetzten public key Verfahren keine Möglichkeit bieten, einzelne Nutzerrollen umzusetzen und zu verwalten.
Bei Ciphertext-Policy Attribute-Based Encryption wird jeder private key eines Nutzers mit einer bestimmten Anzahl an Attributen verknüpft und die Zugriffsstruktur in der verschlüsselten Datei hinterlegt. Die Zugriffsstruktur wird dabei als eine Baumstruktur umgesetzt, in welcher ein Knoten, eine UND Bedingung, ODER Bedingung oder ein Attribut ist. Nur wenn dieser Zugriffsbaum durch die mit dem private key verknüpften Attribute erfüllt wird, ist dem Nutzer die Entschlüsselung der Daten möglich.
Von einem der Autoren von [25] wurde ein Toolkit zur Umsetzung von *CP-ABE* erstellt, welches jedoch vom Autor selbst ausdrücklich nicht für den Produktiveinsatz in sicheren Systemen empfohlen wird. [26]
Neben der sicheren verteilten Speicherung von Daten ermöglicht das Pattern auch eine umfangreiche Zugriffskontrolle. Da diese mathematisch im Pattern verankert ist und nicht einer Instanz vertraut werden muss, ist damit auch die *Enforce*-Strategie erfüllt wird.

2.3.2 Domain Specific Pseudonyms

Anbieter vieler verschiedener Dienste wie Google oder aber auch die Öffentliche Hand besitzen die Möglichkeit, Daten aus verschiedensten Lebensbereichen der Nutzer zu sammeln.
Durch Verknüpfung der erworbenen Daten bietet sich ein umfangreiches Bild des Nutzers, welches weit über jenes hinausgeht, welches sich bei der Nutzung nur eines Dienstes ergeben würde. Selbst bei der Verwendung eines Pseudonyms kann die Eindeutigkeit der Daten dazu ausreichen, ein einzelnes Individuum zu identifizieren.
Eine Lösung besteht darin, für jeden angebotenen Dienst ein eigenes Pseudonym zu erstellen, um dadurch die Verlinkung der Informationen zu erschweren. Dieses Vorgehen stellt eine logische Separation der Daten dar.
Dieser Ansatz wurde auch für die Deutsche Identitätskarte gewählt [10], wo jedem Dienstleister, welche diese Art der Authentifizierung verwendet, ein eigenes Pseudonym zugewiesen wird. Damit können selbst bei Verlust der Daten keine Informationen über mehrere Dienstleister hinweg kombiniert werden.
Pseudonyme im Allgemeinen fallen unter die Strategie des Information *Hiding*, welche auch bei *Domain Specific Pseudonyms* erfüllt wird.

2.4 Aggregation

Informationen sollen maximal aggregiert, also der Detailgrad minimiert werden, so dass die Qualität des Service immer noch erfüllt werden kann. Dadurch kann die Anzahl und der Detailgrad von Daten noch nach dem Sammeln verringert werden, auch wenn sie ursprünglich in dieser Auflösung gebraucht wurden. Zur Aggregation können dabei beliebige Funktionen verwendet werden, nach deren Anwendung der Detailgrad der gespeicherten Daten geringer ist als vorher.

2.4.1 Aggregation over time or space

Daten werden oft sehr detailliert erhoben und in diesem Detailgrad unter anderem für Echtzeitanwendungen verwendet und abgespeichert.
Mit diesen Daten lassen sich zum Teil Tagesabläufe und andere höchst persönliche Einsichten in das Leben von anderen Personen generieren.
Bei der Aggregation over time können alte Daten oft ohne Einfluss auf die Qualität des angebotenen Service aggregiert werden, da zum Beispiel bloß die Summe für statistische Analysen und Auswertungen benötigt wird. Bei der Aggregation over space werden mehrere Datenquellen, wie Sensoren in Clustern zusammengefasst. Die gesammelten Daten eines Clusters werden dann aggregiert, bevor sie zur Verarbeitung weitergeschickt werden. Damit können auch in Echtzeit benötigte Daten geschützt werden.

doi: 10.2313/NET-2015-09-1_10

Ein Beispiel für die Anwendung von Aggregation findet sich in [6], wo die Möglichkeiten analysiert wurden, ein intelligentes Stromnetz unter Berücksichtigung der Privatsphäre zu installieren. Als weitere Beispiele werden in [5] der Gesundheitssektor, Cloud Services, allgemeine Sensornetzwerke oder die Überwachung von Menschen beschrieben.

2.5 Inform

Nutzer sollen maximale Transparenz über Sammlung, Verarbeitung und Speicherung von Daten erhalten. Auch Informationen über Maßnahmen, welche zum Schutz der Daten getroffen wurden, fallen unter die *Inform*-Strategie.

2.5.1 Asynchronous notice [17]

Viele mobile Geräte zeichnen Standortdaten oder andere sensorbasierte Werte auf und zwar auf eine für den Nutzer nicht transparente Art und Weise.
Selbst wenn die Genehmigung dafür vom Nutzer erteilt wurde, kann oft nach langer Zeit nicht mehr garantiert werden, dass dieser sich dessen noch bewusst ist. Eine andere Möglichkeit ist, dass diese Genehmigung von einem vorherigen Besitzer des Gerätes erteilt wurde.
Um größtmögliche Transparenz zu erreichen, sollten nach einem vorher definierten Muster asynchrone Erinnerungsnachrichten mit der Aufklärung über aktuell erhobene Daten verschickt werden. Asynchron beschreibt in diesem Zusammenhang die Eigenschaft, dass die Nachricht nicht nur bei der erstmaligen Verwendung des Sensors geschickt wird.
Der mittlerweile eingestellte Dienst von Google Latitude bot eine Möglichkeit sich mit einer Email informieren zu lassen, wann immer der Standort mit anderen externen oder internen Anwendungen geteilt wurde.

2.5.2 Privacy dashboard [27]

Die Menge an Daten, welche von mobilen Anwendungen oder anderen Services gesammelt werden, sind sehr umfangreich und unübersichtlich. Eine Datenschutzerklärung, in welcher über alle relevanten Eingriffe in die Privatsphäre aufgeklärt wird, hat für Nutzer in der Praxis wenig Relevanz, da diese mit juristischen Floskeln gefüllt wird und nicht mehr die Aufgabe erfüllen die Nutzer bei der Abwägung von Vor- und Nachteilen eines Dienstes zu unterstützen.
Gesucht ist also eine Lösung, mit der über alle gesammelten Daten übersichtlich und leicht auffindbar informiert wird, damit der Nutzer nicht erst im Nachhinein herausfindet, welche Daten von ihm gesammelt und verarbeitet wurden.
Die Lösung dafür ist, alle relevanten Daten auf einer Seite zu sammeln und soweit zu komprimieren, dass der Nutzer die Übersicht behält, ihm aber trotzdem noch klar ist, welche Daten von ihm in welchen Bereichen erhoben werden. Dieses *Privacy Dashboard* muss auch noch für den Nutzer leicht auffindbar sein, wenn er danach sucht.
Beispiel für den Einsatz des Patterns ist bei Android die Übersicht über die von Apps eingeforderten Berechtigungen, welche vor der Installation gezeigt werden, aber auch noch im Nachhinein jederzeit eingesehen werden können.
Wenn über das *Privacy Dashboard* dem Nutzer auch noch Kontrollmöglichkeiten in die Hand gegeben werden, wird damit auch die *Control*-Strategie erfüllt. Beispiel dafür ist die mittlerweile bestätigte Möglichkeit in Android M, ähnlich dem bereits existierenden System von Cyanogen Mod, Apps einzelne Berechtigungen auch wieder entziehen zu können.

2.5.3 P3P [18]

P3P war der Versuch, einen maschinell auswertbaren Standard zu schaffen, mit dem Nutzer über von ihnen gesammelte Daten informiert werden können.
Das Programm war Gegenstand einiger wissenschaftlicher Arbeiten, wurde aber mit Version 1.1 eingestellt. Neben der mangelnden Akzeptanz, besonders von Seiten der großen Browser-Hersteller, gab es Probleme die Einhaltung der Versprochenen Datenschutzversprechen zu kontrollieren. Da die P3P kompatiblen Datenschutzerklärungen von den Services selbst erstellt wurden, gab es keine Möglichkeit sicherzustellen, dass diese auch eingehalten wurden.

2.6 Control

Als Gegenstück zur *Inform*-Strategie müssen dem Nutzer auch Möglichkeiten gegeben werden, um Kontrolle über die eigenen Daten zu erhalten. Mit eingeschlossen ist auch die Einfachheit und Zugänglichkeit der Werkzeuge, die dafür zur Verfügung gestellt werden.

2.6.1 Encryption with user controlled key [20]

Einerseits wollen Nutzer den Komfort von online-basierten Angeboten nutzen, andererseits aber trotzdem die Kontrolle über ihre Daten behalten. Eine Möglichkeit der Kontrolle ist symmetrische Verschlüsselung der abgespeicherten Daten mithilfe eines kryptografischen Keys.
Solange der Anbieter, welcher die Daten speichert, auch im Besitz des Keys ist, kann keine vollständige nutzer-basierte Kontrolle garantiert werden. Entweder der Anbieter selbst, oder dritte Parteien, welche Serviceleistungen für den Anbieter durchführen, können diese Dateien einsehen. Auch der Zugriff von Regierungsorganisationen kann im Allgemeinen nicht ausgeschlossen werden.
Zur Lösung dieses Problems können Dateien vor dem Hochladen verschlüsselt und der Schlüssel lokal gespeichert werden. Auch eine Lösung, bei dem der Schlüssel online gespeichert wird, aber die Passphrase lokal einzugeben ist, ist als Kompromiss zwischen Benutzerfreundlichkeit und nutzerbasierter Kontrolle möglich.
Durch die Trennung von Daten und dem zugehören Schlüssel wird auch die *Separate*-Strategie erfüllt. Eine Beispielanwendung, welche zwar den Schlüssel physikalisch trennt, aber nicht ganz die Anforderung des user-controlled Keys erfüllt, ist KeyNexus [21]. Dabei handelt es sich um einen cloud-Service ausschließlich für Verschlüsselungs-Keys.

2.6.2 Token based access

Viele Anwendungen bieten den Zugriff durch Dritte über eine vordefinierte API, genauso wie viele Anwendungen Daten für den Nutzer speichern.
Sowohl für die API als auch für persönliche Daten soll es eine für den Nutzer möglichst einfache, aber dennoch sichere Möglichkeit geben, festzulegen, welchen Anwendungen oder andere Personen er Zugriff gibt.
Solch ein Zugriff wird üblicherweise mithilfe von Token realisiert, welche die zugreifende Partei gegenüber dem Dienst autorisiert. Wenn individuelle Token vergeben werden, gibt es für den Nutzer auch eine einfache Möglichkeit, Zugriffe zu identifizieren und einzelnen Token die Zugriffsrechte wieder

doi: 10.2313/NET-2015-09-1_10

zu entziehen.
Ein Beispiel für die Verwendung von individuellen Token ist tum online, wo bereits erteilte Genehmigungen auch wieder zurückgenommen werden können.
Bei google drive hingegen können Daten mithilfe eines privaten links geteilt werden. Auch hier können erteilte Genehmigungen wieder zurückgenommen werden, allerdings nur indem der private link, für alle die ihn besitzen, deaktiviert wird.

2.6.3 Broadcasting with client side selecting

Um ortssensitive Services, wie zum Beispiel Wetterberichte oder Gastronomieempfehlungen anbieten zu können, müssen vorher entsprechende GPS-Daten gesammelt und an den Server gesendet werden.
Jede Sammlung von Daten und das Versenden dieser hat Einfluss auf die Privatsphäre, insbesondere gibt der Nutzer in diesem Moment die Kontrolle über seine eigenen Daten ab.
Eine Möglichkeit ist die Rasterung in große Gebiete, welche zum Beispiel auf Bundesländerebene geschehen könnte. Daraufhin werden an alle in diesem Gebiet befindlichen Empfänger ein Broadcast mit allen für dieses Raster relevanten Daten gesendet. Der Nutzer hat dann die Möglichkeit aus den empfangenen Informationen, die für ihn relevanten, auszuwählen.
Für das Wetterberichtbeispiel würde das bedeuten, dass der Nutzer alle Daten von dem Bundesland in dem er sich gerade befindet, empfängt. Aus diesen Daten kann der Nutzer dann lokal das Wetter, für den Ort an dem er sich gerade befindet, auswählen.
Neben der zusätzlichen Kontrolle wird auch die Menge an gesammelten Daten verringert, weshalb damit auch die *Minimise* Strategie erfüllt wird.

2.6.4 Blurred Personal Data [19]

Viele Websites und andere Dienste benötigen Informationen wie den Standort des Nutzers.
Nutzer sollten die Kontrolle über den Detailgrad der von ihnen übermittelten Daten erhalten.
Durch eine Auswahlliste über die der Nutzer selbstständig den Detailgrad der übermittelten Daten bestimmen kann, wird die Kontrolle über die eigenen Daten erhöht. Eine solche Lösung könnte entweder individuell eingepflegt werden oder als Standard für eine komplette Plattform realisiert werden. Bei zweiterem hätte der Nutzer dann an jeder Stelle, an der er aufgefordert wird Standortdaten preiszugeben, die Möglichkeit, den Detailgrad selbst zu bestimmen.
In der zitierten Literatur wird der Mockup eines Browser-Addons gezeigt, über den dann der Nutzer die Auswahlmöglichkeit zwischen den Standort der Stadt, der Postleitzahl oder des genauen Standort hätte. Auch eine absolut festgelegte Unschärfe ist vorstellbar, hier muss aber bedacht werden, dass bei mehrfacher Standortabfrage die Streuung mathematisch stark reduziert werden kann.
Ein ähnlicher Ansatz bei dem sich der Entwickler aber auf eine einzige sinnvolle Auflösung im Vorhinein festlegen muss, ohne dem Nutzer eine Wahl zu bieten, wird durch *Location Granularity* als Pattern der *Minimise*-Strategie verfolgt.

2.7 Enforce

Das Vorhandensein einer Datenschutzerklärung, welche auch alle gesetzlichen Vorgaben erfüllt, ist eine Strategie um die Privatsphäre zu erhöhen. Allerdings nur dann wenn die Einhaltung der Datenschutzerklärung auch durchgesetzt wird.

2.7.1 Sticky Policies [8]

Gesammelte Daten werden oft zwischen verschiedenen internen und externen Parteien ausgetauscht. Um dabei die Privatsphäre der Nutzer zu wahren, gibt es eine Vielzahl an Verträgen, service-level agreements, spezielle frameworks und Auditing-Verfahren.
Dabei hat der Nutzer allerdings keine individuelle Kontrolle mehr, wie mit seinen Daten umgegangen werden soll.
Sogenannte Sticky Policies sind Metadaten, welche den persönlichen Daten angehängt und dann auch mit den Daten so an andere Parteien weitergegeben werden. Sie bestimmen die Voraussetzungen und Grenzen, in welchen die Daten verarbeitet werden dürfen.
Im Zuge des EnCoRe-Projekts [9] wurde eine Architektur geschaffen bei der eine Trust Authority die Einhaltung der Policies der Parteien prüft und erst dann den Zugang zu den verschlüsselten Daten gewährt.

2.7.2 k-Anonymity [15]

Das Konzept der k-Anonymity wurde ursprünglich für den akademischen Einsatz erdacht. Als Kontext dienen dabei Datenbanken von Instituten, wie öffentliche Einrichtungen oder Banken, welche ihre Daten für Forschungszwecke teilen möchten.
Das Problem besteht darin, eine Version dieser Daten zu veröffentlichen, von denen garantiert ist, dass sie die Anonymität der Individuen gewährleisten. Es darf nicht möglich sein, dass einzelne Personen sich identifizieren lassen. Über relinking könnten kritische Daten wie die Gesundheitsakte mit öffentlich zugänglichen Daten verknüpft werden, um dadurch ein Gesamtprofil zu erhalten. Schon einige wenige Daten können dabei in Kombination eindeutig sein.
Bei der Verwendung von k-Anonymity werden die Daten in Äquivalenzklassen eingeteilt, wobei jede dieser Klassen mindestens k Einträge beinhaltet. Von diesen Daten werden dann alle Identifikatoren entfernt. Als Quasi-Identifikatoren bezeichnet man dabei Tupel an Attributen, welche ein Individuum eindeutig identifizieren können. Deshalb werden die Äquivalenzklassen so gewählt, dass alle Datensätze einer Klasse die gleichen Quasi-Identifikatoren besitzen. Darunter fallen auch alle Attribute, welche mithilfe von anderen öffentlich zugänglichen Datenbanken, zu Quasi-Identifikatoren werden. Zur Bildung der Äquivalenzklassen muss teilweise die Auflösung der Daten verringert werden. Ein Beispiel für Identifikatoren ist der Name und für Quasi-Identifikatoren die Attribute des Tupels (Postleitzahl, Geburtsdatum, Alter).
Das Verfahren wurde im Jahr 2002 patentiert [16].
Zur eigentlichen Anonymisierung, also der Erstellung der Äquivalenzklassen, können verschiedene Methoden verwendet werden, welche sich typischerweise der *Aggregate*-Strategie bedienen.

2.8 Demonstrate

Die *Demonstrate*-Strategie ist der nächste logische Schritt nach *Enforce*. Die Strategie legt fest, dass die Einhaltung der Datenschutzerklärung jederzeit und nachvollziehbar demonstriert werden kann. Wenn davon ausgegangen wird, dass die Umsetzung der *Inform*-Strategie ehrlich erfolgt und die

doi: 10.2313/NET-2015-09-1_10

Sammlung und Verarbeitung rechtskonform ist, kann auch jegliches Pattern, welches über die gesammelten Daten informiert, die *Demonstrate*-Strategie erfüllen.

2.8.1 Accounting

Es kommt vor, dass Datenlecks in Firmen geleugnet werden bis es nicht mehr widerlegbare Beweise dafür gibt. Oder die Lecks werden erst entdeckt, wenn die Daten bereits im Umlauf sind.
Neben der Zugriffskontrolle gibt es kaum Möglichkeiten, im Nachhinein zu zeigen, wann von wem auf welche Daten zugegriffen wurde, oder von welchem Dienst die Daten verarbeitet wurden. Das führt zu einer Grundskepsis von Nutzern gegenüber online basierten Diensten.
Um die Einhaltung der Datenschutzrichtlinien jederzeit nachweisen zu können, sollte ein System auf Servern, welche persönliche Daten speichern, eingerichtet werden, welche alle relevanten Zugriffe auf die Daten und darauf ausgeführte Funktionen, lückenlos aufzeichnet. Das System muss zuverlässig und fehlerresistent sein und sollte einen Output ähnlich einem Log erzeugen, welcher in geeigneter Art und Weise einsehbar sein muss. Damit sollen alle externen und internen Zugriffe aufgezeigt werden.

2.8.2 Auditing

Viele Firmen versuchen zu beteuern, dass ihre Software sicher ist und auch alle Datenschutzrichtlinien eingehalten werden. Diese Aussagen sind nicht nachprüfbar, weil viel Code closed source ist. Aber auch bei open source Projekten gehen die Lines of Code schnell in die hunderttausend. Vielen Nutzern fehlt die Expertise oder aber auch die Zeit, diesen Code zu prüfen.
Eine mögliche Lösung ist Software-*Auditing* von unabhängigen Experten, welche den Code auf Backdoors oder unbeabsichtigte Schwachstellen prüfen.
Ein Beispiel, wo mit Hilfe von *Auditing* [7] versucht wurde, das Vertrauen der Nutzerbasis wiederherzustellen ist TrueCrypt, eine Verschlüsselungssoftware deren Entwicklung Mai 2014 eingestellt wurde, weil es angeblich schwerwiegende Sicherheitslücken darin gebe.
Im Sinne der *Inform*-Strategie könnte *Auditing* standardisiert und nach erfolgreicher Prüfung der Software eine Art Qualitätssiegel, ähnlich der *trusted shops* Auszeichnung im online shopping Bereich, vergeben werden.

3. CONCLUSIO UND AUSBLICK

Die Arbeit zeigt, dass es bereits zu allen Strategien, welche die Privatsphäre erhöhen, Design Pattern gibt mit welchen Privacy by Design realisiert werden kann. Viele Pattern setzen gleich auf mehreren Ebenen an und werden in einzelnen Projekten schon erfolgreich eingesetzt. Indem das Bewusstsein für und die Verbreitung von Privacy Pattern erhöht wird, kann ein wichtiger Beitrag zum Schutz der Privatsphäre und damit das Vertrauen der Nutzer in solche Software geschaffen werden.
Nichtsdestotrotz benötigt der Ansatz, den Schutz der Privatsphäre bereits auf der Systemdesignebene zu gewährleisten, sicherlich noch weitere Bemühungen in der Forschung. Projekte wie PRIPARE [11] könnten dafür eine entscheidende Rolle spielen.
Als Ausblick sollte die erfolgte Sammlung an Pattern in Zukunft noch erweitert und mit weiteren Detailinformationen wie zur Implementation bereichert werden.

4. LITERATUR

[1] HOEPMAN, Jaap-Henk. Privacy Design Strategies. 2012.

[2] HAFIZ, Munawar. A pattern language for developing privacy enhancing technologies. Software: Practice and Experience, 2013, 43. Jg., Nr. 7, S. 769-787.

[3] privacypatterns.org - Strip invisible Metadata, http://privacypatterns.org/patterns/Strip-invisible-metadata, accessed: 28-05-2015

[4] twitter - faq, https://support.twitter.com/articles/20169321-posten-von-fotos-auf-twitter, accessed: 28-05-2015

[5] SHI, Elaine, et al. Privacy-Preserving Aggregation of Time-Series Data. In: NDSS. 2011. S. 3.

[6] KURSAWE, Klaus; DANEZIS, George; KOHLWEISS, Markulf. Privacy-friendly aggregation for the smart-grid. In: Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2011. S. 175-191.

[7] Open Crypto Audit Project, https://opencryptoaudit.org/, accessed: 14-05-2015

[8] PEARSON, Siani; MONT, Marco Casassa. Sticky policies: an approach for managing privacy across multiple parties. Computer, 2011, 44. Jg., Nr. 9, S. 60-68.

[9] MONT, Marco Casassa; SHARMA, Vaibhav; PEARSON, Siani. EnCoRe: dynamic consent, policy enforcement and accountable information sharing within and across organisations. HP Laboratories technical Report: HPL-2012-36, 2012.

[10] BENDER, Jens, et al. Domain-specific pseudonymous signatures for the german identity card. In: Information Security. Springer Berlin Heidelberg, 2012. S. 104-119.

[11] PReparing Industry to Privacy-by-design by supporting its Application in REsearch, http://pripareproject.eu/, accessed: 10-05-2015

[12] David H. Flaherty. Protecting Privacy in Surveillance Societies: The Federal Republic of Germany, Sweden, France, Canada, and the United States. University of North Carolina Press, Chapel Hill, NC, USA, 1989.

[13] privacypatterns.org - Location Granularity, http://privacypatterns.org/patterns/Location-granularity, accessed: 27-05-2015

[14] FREEDMAN, Michael J.; MORRIS, Robert. Tarzan: A peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002. S. 193-206.

[15] SWEENEY, Latanya. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, 10. Jg., Nr. 05, S. 557-570.

[16] Systems and methods for deidentifying entries in a data source - https://www.google.com/patents/US7269578, accessed: 01-06-2015

[17] privacypatterns.org - Asynchronous Notice, http://privacypatterns.org/patterns/Asynchronous-notice, accessed:

07-05-2015
[18] P3P, http://www.w3.org/P3P/, accessed: 01-06-2015
[19] CHUNG, Eric S., et al. Development and evaluation of emerging design patterns for ubiquitous computing. In: Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, 2004. S. 233-242.
[20] privacypatterns.org - Encryption with user managed keys, http://privacypatterns.org/patterns/Encryption-user-managed-keys, accessed: 01-06-2015
[21] Key Nexus, https://keynexus.net/, accessed: 01-06-2015
[22] Privacy-Enhancing Technologies: The Path to Anonymity. Office of the Information & Privacy Commissioner of Ontario, Registratiekamer, 1995.
[23] Idemix - IBM research zurich, http://www.zurich.ibm.com/idemix/whatitdoes.html, accessed: 06-06-2015
[24] CHAKRAVARTY, Sambuddho; STAVROU, Angelos; KEROMYTIS, Angelos D. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In: Computer Security–ESORICS 2010. Springer Berlin Heidelberg, 2010. S. 249-267.
[25] BETHENCOURT, John; SAHAI, Amit; WATERS, Brent. Ciphertext-policy attribute-based encryption. In: Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007. S. 321-334.
[26] http://hms.isi.jhu.edu/acsc/cpabe/
[27] privacypatterns.org - Privacy Dashboard, http://privacypatterns.org/patterns/Privacy-dashboard, accessed: 06-06-2015
[28] SO/IEC 29100, Information technology – Security techniques – Privacy framework, Technical report, ISO JTC 1/SC 27
[29] GAMMA, Erich, et al. Design patterns: elements of reusable object-oriented software. Pearson Education, 1994.

doi: 10.2313/NET-2015-09-1_10

Who Is Scanning the Internet?

Roman Trapickin
Betreuer: Oliver Gasser, Johannes Naab
Innovative Internet-Technologien und Mobilkommunikation SS2015
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: roman.trapickin@tum.de

ABSTRACT

Today's port scanning software is able to perform massive port scans up to the complete IPv4 space within minutes. Once a device is connected to the Internet, it will be almost immediately scanned for open ports and services. Most of these scans are done by anonymous individuals, in particular targeting at finding and exploiting system vulnerabilities. However, there is also a variety of individuals and organizations openly practicing massive port scanning and pursuing different objectives. In this paper we will introduce several scanning entities, while emphasizing their motives. In addition we will propose an entity classification model. In order to endorse further understanding of the topic we will also discuss port scanning as a measurement discipline as well as introduce the contemporary port scanning software used for Internet-wide scans.

Keywords

port scan, internet-wide scan, measurement, entities, nmap, zmap, masscan, classification, caida, sonar, sipscan, conficker, shodan, hacienda, legality, ethics

1. INTRODUCTION

IPv4 address space consists of 2^{32} or almost 4.3 billion possible IP addresses, which is within the *giga* order of magnitude. Today's computers have CPUs with gigahertz clock rates and gigabytes of RAM and storage. Today's networks allow bandwidths exceeding 1 gigabit per second. This makes iterations over the entire IPv4 space possible within a comparatively short time period. By comparison, IPv6 address space consists of 2^{128} or nearly 340 undecillion (10^{36}) addresses, which is rather unlikely to iterate over entirely in the nearest future.

Since the massive transition to IPv6 has not yet begun and IPv4 is still by far the dominant Internet protocol,[18] we now have a unique opportunity to reach every IPv4 address on the Internet in reasonable time. The year 2013 has seen ZMap and masscan emerging, two port scanning tools, which promised to port scan the entire Internet in under an hour on consumer hardware. Particularly ZMap has received an extensive IT media coverage,[2][10][31] which once again raised the discussion about port scanning and associated threats.

In this paper we will attempt to answer the question "Who is scanning the Internet?" by giving an insight into the current trends of massive port scanning. First, we will briefly introduce the technical terms used for describing port scanning activities in section 2. Popular software tools will be discussed in section 3. In section 4 we will talk about the individuals and organizations, which openly perform Internet-wide scans. We will focus on understanding their intentions, used tools and techniques. Finally, we will endeavor a classification model for these entities in section 5. As a short supplement we will revitalize the discussion about the legal and ethical aspects of (massive) port scanning in section 6.

2. SCANNING BASICS

The term "Internet(-wide) scan(ning)" describes a *port scanning* procedure performed by a single or multiple scanning entities on a considerable amount of hosts. There is no clear requirement or specification at which point scanning multiple hosts may be dubbed an "Internet scan", however scanning the complete IP spaces of regional Internet registries or even countries definitely falls into this term. In this section we will shortly describe the main reason behind such massive port scanning initiatives and also port scans in general as well as introduce main concepts and terms used throughout this paper. Although rudimentarily explained in this section, the real motives will be introduced in sections 4 and 5.

The main purpose of port scanning is *gathering information* about offered services from hosts connected to a network. This is done via sending probe messages to targeted hosts and prompting a response later on. As the name suggests, port scanning is centered around Transport layer ports and hence mostly based on the Transmission Control Protocol (TCP). Nevertheless both adjoining Network and Application layers often provide additional information about the targeted system.

2.1 Port Status

First, we explain what kind of knowledge a scanning entity can receive from a TCP segment. The essential step in (Internet-wide) port scanning is to determine the port status of a remote host:[24]

open port indicates that an application is accepting connections on this port. Finding an open port is the main goal for a scanning entity, since it discloses the most knowledge about targeted host.

closed port indicates that there is no active application on the other end. Bearing significantly less information about the host, a closed port could still provide some clues about e. g. the operating system by fingerprinting the TCP/IP stack. Various operating systems exhibit char-

doi: 10.2313/NET-2015-09-1_11

acteristic behavior while generating the freely selectable TCP/IP fields, so that collecting multiple responses and matching them against a database with known fingerprints makes OS detection possible.

undetermined port status is the least desirable message for a scanning entity. The port scanning software does not receive any response. In most cases the port is **filtered** – protected by a firewall. Some port scanning software, most prominently nmap, which offers various scanning techniques that – under favorable circumstances – may return more informative results. As a matter of consequence, the entire scanning process slows down drastically. The scanning techniques will be introduced in section 3.1.

In addition to Transport and Application layer knowledge, a scanning entity can acquire relevant information from the Network layer. Querying a WHOIS database with targeted IP address will provide the scanning entity with approximate host geolocation and ISP data among other things. A variety of WHOIS services is publicly available on the Internet. In general, retrieving a short description of running services is called **banner grabbing**.

2.2 Defining the Scope of a Scan

The next step is to define the scope of Internet scanning. This scope is defined by two dimensions, namely ports and hosts. A naïve calculation for IPv4 address space for every port results in $2^{32} \cdot 2^{16} = 2^{48}$ communication endpoints to be scanned. However, subtracting private IP ranges as well as blacklisting some IP addresses will not change the order of magnitude. By comparison, brute-forcing a 48-bit cryptographic key space is considered feasible today,[6] but still only on dedicated hardware.[29] Even though iterating over 2^{48} combinations alone can take significant amount of time on consumer-grade machines, actually it is the *bandwidth* that creates the major bottleneck. While scanning a remote host on the Internet, the scanning performance depends on bandwidth of every path segment a packet has to traverse. In fact, there are various factors, such as congestion control, which negatively affect the overall bandwidth. As a result, such scans are indeed possible, albeit rather impractical. Instead, scanning entities concentrate their effort on relevant ports, so that time to completion can be thoroughly improved. There exist three major approaches to reduce the scanning scope:[22]

Horizontal scan describes a port scan performed for the same port on multiple hosts. An extreme example of a horizontal scan is a **/0 scan** (entire IPv4 space) on a single port. Horizontal scans are often used by attackers to detect open ports on a large number of machines in order to exploit vulnerabilities of the listening application. In turn, such scans can be used for massive security audits, i. e. measuring global distribution of a vulnerability.

Vertical scan describes scanning multiple ports on a single host. Scanning every port out of 2^{16} of a host is called **vanilla scan**, while scanning a small subset is dubbed **strobe scan**. Such scans are mostly done for vulnerability detection on single systems. Obviously, such limited scope is not suited for Internet-wide scans.

Block scan is a combination of both horizontal and vertical scan, therefore a scan of multiple ports on several hosts. An extreme example is a /0 vanilla scan. As mentioned before, large block scans are poorly scalable, since including a single host to the target domain results in up to 2^{16} additional ports. In practice only a small number of ports per host is scanned. Hence an Internet-wide block scan may rather be considered as a series of few horizontal scans or alternatively as a /0 strobe scan. Such scans are as well interesting for attackers aiming at multiple vulnerabilities and security experts doing global research, but also useful for various service discovery tasks.

Figure 1 visualizes the three types of scanning in terms of targeted scope.

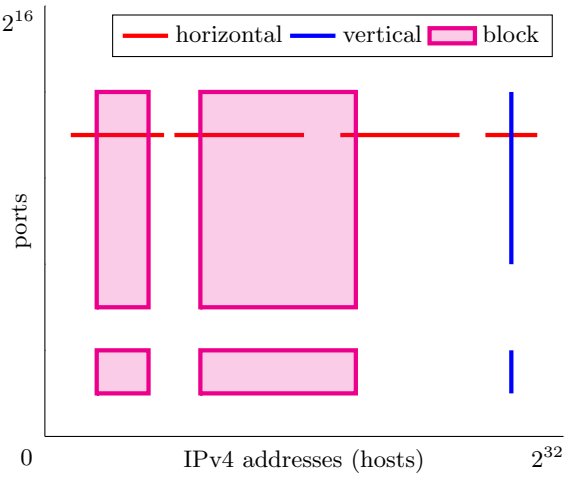

Figure 1: A simplified visualization of scan types

2.3 Measurement Standards

Finally, we will discuss port scanning from a scientific, or more precisely, experimental point of view. Regardless of the real motivation of a scanning entity, a (massive) port scanning initiative is a *quantitative research*, and, as such, it is subject to several measurement and quality standards. Vern Paxson from the International Computer Science Institute in Berkeley, California has proposed the following aspects for sound Internet measurement:[25]

Precision describes a degree of relative proximity between individual measured values. A telling example is the sampling rate of a continuous signal, whereas Paxson cites an example of clocks with 1 µs and 1 ms precision each as well as *filtering* responses. He also states, that, concerning the Internet measurement, precision is "readily apparent" and hence is rather of secondary importance. However, according to Paxson, a *sound* measurement study should at least respond to precision concerns.

Metadata is the data that accompanies the actually measured data. A prominent example of metadata *preservation* are (human-readable) logs which save additional information during the measurement. Metadata is an important aspect for Internet-wide scans, since it is possible to extract crucial information about port status and listening application from accompanying data.

Accuracy is a degree of relative proximity of measured values to the reference value. Acquiring such value can be

doi: 10.2313/NET-2015-09-1_11

done by comparison with values from other sources or previous measurements. When talking about Internet-wide scans, accuracy is often used to describe an effort spent for checking port status. An accurate scan therefore exhausts every possibility to get the best result when a simple check returns an undetermined status. As already mentioned before, elaborate scanning techniques may drastically decrease performance. Most purpose-built scanning tools (e. g. ZMap, masscan) often sacrifice accuracy for the sake of performance.

Misconception refers to false interpretation of results based on faulty measurement execution. A misconception commonly occurs as a consequence of leaving a critical detail out of account. For example, considering host as powered off, though it is actually hidden behind a firewall is a misconception.

Calibration is process of applying techniques for efficiently exposing inaccuracy and misconception issues. This includes i. a dealing with outliers and spikes or comparing multiple measurements. These techniques will not be further discussed in this paper.

Aside from the above aspects, which are supposed to be considered in order to avoid observational errors during the measurement, Paxson also introduces the best practices for working with (read: analyzing) measurement data:

Large volumes of data may lead to unexpected behavior of analysis tools, in turn slowing down the research process drastically. Paxson states, that extracting multiple data samples and analyzing them individually as well as comparing them for common properties and differences, will help to gain an overview about the data before proceeding with analyzing the whole dataset.

Reproducibility of analysis is a fundamental principle, found in every experimental scientific field. Reproducibility describes the ability of a third party to re-enact the experiment for the sake of process validation and verification of results. It is on behalf of the researcher not only to describe the experiment in detail, but also to provide a comprehensible "master script" for compiling the whole analysis chain, thus making its result quickly reproducible.

Public availability of data supplements the reproducibility of analysis. However, publicly available research data not only adds value to comprehensibility of the research, but also contributes to a "common framework", which may be used by different researchers to confirm their results. Paxson endorses the publication of detailed datasets (including metadata), but also addresses the problem of disclosing sensitive information.

2.4 Scanned Entities

A port scan is a bilateral activity, and, as such it affects both scanning and scanned entities. As the last part of the Scanning Basics, we will shortly introduce the possibilities of detecting scans from third parties.

Because of the nature of port scanning, one cannot tell with 100 % certainty, whether an incoming connection is a port scan without confirmation from the scanning entity. Several "friendly" entities, while scanning HTTP services, tend to put contact information into metadata, e. g. short description and a web link in the HTTP User-Agent field that can be quickly seen by a system administrator. However, without this information detecting a port scan is far less obvious. A lot of heuristic detection methods have published, spanning from evaluating intensity of connection establishments[22] to probabilistic models.[21]

Both legal status and ethical aspects of port scanning have been a controversial topic. Opponents consider port scans as a service misuse, mainly because of resources spent on establishing and maintaining connection. Besides, some service providers perceive such thorough examination as inappropriate from ethical standpoint. Legal status and ethical issues will be discussed in section 6.

3. SCANNING SOFTWARE

With rising popularity of the Internet in the mid-1990's, the first publicly available port scanning software emerged. Nmap was one of the first port scanners and has outlived most of its – meanwhile discontinued – competitors such as scanrand and unicornscan. Virtually every port scanner is capable of scanning IP address and port ranges (read: *block scans*), however scanners with limited scan scope also exist, for instance the OS X built-in Network Utility is only capable of vertical scans. Over time, various port scanners have introduced different probe request and response handling paradigms. For example, scanrand was among the first that worked with two processes for sending and retrieving responses asynchronously.[3]

In the following we will introduce the most renowned port scanning software, namely Nmap, as well as newcomers ZMap and masscan, which have drawn much attention promising to complete an Internet-wide scan within minutes. We will also use the opportunity to explain the different approaches used by these port scanners regarding the probe handling and dispatching.

3.1 Nmap

Nmap is the best-known port scanning command-line utility. It was specifically designed for both massive scans as well as scanning single hosts. The original author, Lyon "Fyodor" Gordon, has been developing Nmap since 1997, and with increasing functional complexity, development was later overtaken by the user community. Over its fairly long period of existence, Nmap has become almost synonymous with port scanning. Besides port scanning it has an extensive set of features, including host discovery, detection of running services and operating systems, as well as scripting for automation purposes. Nmap is also highly tunable through a variety of command-line parameters.[24]

Nmap excels in offering diverse scanning techniques, being dubbed a Swiss Army knife of port scanning for that reason:[24]

TCP SYN scan is "the default and most popular scan option". Nmap sends a SYN message awaiting a SYN|ACK for an open or RST for a closed port. Lack of response indicates, that the port status cannot be determined.

TCP connect scan utilizes `connect()` – an operating system call – instead of crafting own messages. As opposed

doi: 10.2313/NET-2015-09-1_11

to TCP SYN scan, complete connection is established thus exposing the scan to application layer services.

UDP scan sends UDP messages prompting a UDP (in case of open port) or an ICMP response.

SCTP INIT scan utilizes SCTP INIT chunks prompting an INIT|ACK response in case of an open port, or ICMP error response otherwise.

The following scan options try to exploit/circumvent protocol peculiarities and surrounding infrastructure in order to retrieve more *precise* results for initially undetermined port status.

TCP NULL/FIN/Xmas scan tries to bypass the firewall by setting atypical or nonsensical TCP flag combinations within requests in order to examine the host's reaction. A FIN scan sends a FIN message to the targeted host, which may return an RST in case of a poorly configured non-stateful firewall. For NULL scans no flags are set, whereas for Xmas scan every flag is set.

TCP Maimon scan is similar to a FIN scan, except it sends a FIN|ACK "request". Host must return an RST with information about port status. Some BSD-derived systems drop this message in case of an open port.

Custom TCP scan allows setting arbitrary flags.

TCP ACK scan tries to map firewall rules by sending TCP ACK messages. It only determines whether a port is filtered (RST received?) or not.

TCP Window scan sends ACK requests similarly to ACK scan, however it also considers TCP window size, which enables the distinction between a closed and an open port. Positive window size within an RST response suggests an open port, zero size a closed one.

SCTP COOKIE ECHO scan detects whether a port is closed or not. It sends a SCTP COOKIE ECHO message. If the port is closed, an ABORT message is received. The host must drop the packet for an open port, thus making it indistinguishable from a filtered one.

TCP idle scan works through exploiting the so called zombie host, that utilizes incremental IP ID generation. First, the scanning entity contacts the zombie host to check its IP ID generation strategy. Then, the scanning entity masquerades as a zombie host by spoofing its IP address and sends a SYN request to the target. The target then responds (or not) to the zombie host with SYN|ACK or RST. Finally, the scanning entity once again contacts zombie for checking its IP ID. If it has increased by one, then the port is open, which means, that the zombie host has "surprisingly" received a SYN|ACK and answered with an RST incrementing its internal IP ID variable. Otherwise the targeted port is either filtered or closed. The intention behind the idle scan is to check the port status while remaining completely invisible to the targeted host.

IP protocol scan iterates over Internet protocol numbers, in order to find supported protocols. Should there be any encapsulated Transport layer response, its port will be included in the report.

FTP bounce scan exploits the FTP PORT call, which – where supported – allows the usage of a *remote* FTP server running in passive mode to check port status of the targeted host by sending files to its ports. As with the TCP idle, the idea behind bounce scan is not to disclose oneself to the host.

Nmap also has various parameters to adjust the accuracy–stealthiness–speed tradeoff. The `-T`n parameter allows to choose one of the six presets (numbered 0–5). Each preset includes predefined settings for round-trip timeout, packet delay and parallelism. For instance, a `-T0` or *paranoid* scan will wait 5 min (!) before sending each packet in order to evade intrusion detection/protection systems. By contrast, a `-T5` *insane* scan does not practically involve any artificial delay, sending packets to multiple hosts in parallel with a minimal RTT timeout, thus tolerating only sufficiently fast responses. `-T5` suggests usage on a very fast network, preferring speed over accuracy and stealthiness.[24]

3.2 ZMap

ZMap is being developed by Zakir Durumeric, Eric Wustrow, and J. Alex Halderman at the University of Michigan. It was initially released in August 2013,[12] and, since then, it has received considerable coverage by media and other academic researchers.[2][10][31] The reason for this is the claim to perform a full Internet-wide scan in under 45 minutes.[12]

Unlike the general-purpose Nmap, ZMap was custom-built with Internet-wide scans in mind. As a result, ZMap is far less customizable. Despite having functionality for group scans, developers state the *horizontal* scans as its main modus operandi. In order to achieve such short times, ZMap differs from Nmap by utilizing the following features:[12]

Probe randomization. As we already mentioned before, scan performance depends on network bandwidth. Randomizing the order in which probes are sent helps to avoid bandwidth saturation. Using a random order will drastically decrease the probability of simultaneously sending probes to hosts which belong to the same network, and, as a consequence, overloading this network's infrastructure will become far less likely. ZMap uses address space permutation via multiplicative cyclic group modulo *prime* p: $(\mathbb{Z}/p\mathbb{Z})^\times$ with $p = 2^{32} + 15$. This group reaches the entire IPv4 space except for 0.0.0.0. For each scan ZMap generates a new primitive root g and randomly chooses the initial IP address a_0. The next value a_{i+1} is then calculated with $a_{i+1} = (a_i \cdot g) \mod p$. ZMap also may utilize *sharding* in order to split the workload between n threads. In this case g^n is taken into calculation, resulting in $a_{n(i+1)+j} = a_{ni+j} \cdot g^n \mod p$ for the j-th shard $(0 \leq j < n)$.[1]

Asynchronous design. Unlike Nmap, ZMap *does not keep state* in order to match responses to requests. Instead, sending and receiving packets happens independently in separate threads, which in turn increases overall performance. Thus, the sending thread "forgets" about the connection immediately after it has been initiated. However, it is still possible for receiving threads to get to know, if the incoming response was intended for ZMap. In a similar manner to SYN cookies, ZMap calculates a UMAC using a scan-specific key over the destination IP address. The UMAC value is then written into available probe fields, e. g. source port and SEQ in case of TCP. Should there be a response to this probe, the receiving thread will be able to verify the request origin by comparing the destination port and decremented ACK field value with computed UMAC over the IP source field.[12][1]

No retransmission. ZMap tolerates packet loss for the sake of performance. ZMap sends a fixed number of probes to the target, yet a single packet is sent by default. Durumeric et al. concluded, that there are no significant losses with this setting.

Besides, ZMap is able to use the PF_RING ZC driver in order to bypass kernel routines and construct Ethernet frames on its own. This allows to exceed the 1 GbE bottleneck of the Linux kernel (assuming faster connection, e. g. 10 GbE).[1]

3.3 Masscan

Masscan is being developed by security researcher Robert David Graham, with its initial release dating back to October 2013. Graham went so far as to say masscan would perform an Internet scan within 3–6 minutes, assuming a 10 GbE connection.[15] Since then, there has been a silent rivalry between Graham and ZMap developers. Graham stated that ZMap's speedup over other port scanners is due to improvements in the Linux kernel.[16] In response, Durumeric et al. tried to experimentally disprove masscan's advance in their works.[1]

Masscan uses the same asynchronous model – splitting responsibilities for sending and receiving between threads – as scanrand, unicornscan, or ZMap. The main advantage of masscan lies in **probe randomization**. As with ZMap, the aim is to distribute IP addresses on-the-fly while iterating over the IPv4 address space. In his implementation Graham relinquished a certain degree of statistical randomness to save computation time for very high packet rates. At the rate of 10 million packets per second (Mpps), one has roughly 100 ns for packet processing. In order to beat this time masscan uses the *BlackRock* algorithm – a modified implementation of symmetric encryption algorithm DES with less rounds and modulo operations in place of binary ones allowing arbitrary ranges. Since DES is a Feistel cipher with substitution boxes, the ciphertext blocks appear as uniformly distributed pseudorandom bit strings.[15] Graham has also spoken about improving statistical distribution while retaining performance in the future.[14]

Masscan is based on the C10M paradigm proposed by Graham – handling 10 million connections at 10 Gbps / 10 Mpps with 10 µs latency, etc. Graham calls for removing the network routines from kernel in order to achieve this goal. Masscan may optionally utilize PF_RING ZC driver to bypass kernel and use its own TCP/IP stack instead.[13]

4. SCANNING ENTITIES

In the following we will introduce several scanning entities that perform Internet-wide scans. Since the persons behind some scan initiatives are actually unknown, we use the name of event for describing them.

CAIDA The Center for Applied Internet Data Analysis (CAIDA) is a research institute based in San Diego, California. CAIDA emerged in 1998 as a cooperation between the San Diego Supercomputer Center (SCDC) at UC San Diego as well as various commercial and governmental organizations. The main purpose of CAIDA is measurement, monitoring, and analysis of the Internet infrastructure. Though it is merely one of many research fields, CAIDA is performing extensive active and passive measurements. For instance, port scan detection is performed on the so called UCSD Network Telescope – a "globally routed /8 network" – using packet capture. For active port scanning CAIDA uses the task-based tool *scamper* being developed by Matthew Luckie. Scamper supports TCP, UDP and ICMP probing as well as pinging and traceroute. Scan results and visualizations are available at `http://www.caida.org/data/`.[4]

University of Michigan The developers of ZMap are maintaining the "Internet-Wide Scan Data Repository" publicly available at `https://scans.io`. The website is hosting the scan results done by ZMap team, but also by any third party willing to publish their research data.

Project Sonar is a community effort guided by Rapid7, a US company specializing on IT security. The project was initially defined by horizontal IPv4 Internet-wide scans on port 443 (HTTPS) using ZMap, and, furthermore, by collecting SSL/TLS certificates. However, Project Sonar was later expanded to UDP scans as well as gathering HTML index files from port 80 and DNS records. The goal of the project is to deliver a global view on the security of web services. The results are hosted by the aforementioned ZMap Team's scans.io repository.[27]

SIPscan was an Internet-wide group scan on UDP ports 5060, 5061, 5070 and TCP port 80. SIPscan was detected and observed by UCSD Network Telescope for twelve days in February 2011. The name originates from the Session Initiation Protocol (SIP) which operates on these UDP ports. SIP is mainly used for Internet telephony (voice and video calls). The protocol is famous for being vulnerable to a great variety of attacks. SIPscan was a globally distributed scan performed by a botnet spanning over 3 million IPv4 addresses. Several computers worldwide were co-opted into the botnet via Sality malware, however the initiator(s) of SIPscan remain(s) unknown.[8]

Conficker traffic is another prominent example of using port scanning for finding vulnerable hosts. Conficker is the name of a computer worm, which exploits the Server service vulnerability in Windows operating systems. First, the attacker initiates an SMB session on the TCP port 445 of the victim. Then the attacker plants malicious code into the request which forces the victim to download an executable file. After analyzing captured Conficker traffic, the ZMap Team has reported, that 445/TCP scans are rather short-range horizontal scans. Additionally, Conficker traffic is the dominant port scanning initiative among small scans (targeting $< 10\,\%$ of IPv4 space).[11]

Internet Census 2012 was another Internet-wide scan using a botnet – dubbed Carna Botnet – of roughly 420.000 embedded devices (mostly routers with either default, weak, or even no password). These devices were supplied with lightweight Nmap builds, each scanning a small host range. Internet Census 2012 is a distributed Internet-wide group scan on 150 most used ports. The initiator chose to remain anonymous, however an extensive description, evaluation, and visualization of results along with scan data is available at `http://internetcensus2012.bitbucket.org`.[17]

Shodan is a controversial search engine created 2009 by the "Internet cartographer" John Matherly for finding various devices connected to the Internet. Since CNNMoney released an article about a vast amount of insufficiently

doi: 10.2313/NET-2015-09-1_11

protected services publicly available on the Internet and disclosed by Shodan, spanning from baby cams to power station remote control systems, there have been major concerns about legal and ethical background of the service. Security expert Richard Bejtlich called Shodan an "intrusion as a service" (as a reference to various cloud service delivery models). For its continuous scans, Shodan utilizes its own software simply named "Shodan crawler", which does both port scanning and banner grabbing. Shodan also provides a scan result repository ScanHub, where users can upload their Nmap/masscan XML files for searching and visualization purposes, but also to make it available publicly in Shodan's search results. An interesting fact is that the Internet Census 2012 was only possible due to the variety of unprotected devices on the Internet, whereas Shodan drew media attention to the possible extent of such botnets.[30]

NSA/GCHQ HACIENDA is the name for a reconnaissance program led by the US National Security Agency (NSA) and the UK Government Communication Headquarters (GCHQ) – both governmental intelligence organizations of their respective countries. The program was classified top secret, however, in August 2014, presentation slides leaked into the public domain. HACIENDA aims at port scanning of country IPv4 ranges. Port scans were complete for 27 countries. According to slides, HACIENDA staff also takes orders from associates for scanning countries originally not on the list. HACIENDA uses Nmap as scanning tool with host randomization parameter.[20]

Open Resolver projects aim at finding poorly configured recursive DNS servers (scanning for port 53), which may be be misused for DNS amplification attacks – a form of DDoS attack performed via flooding the target with DNS responses. The intention of the project is to provide an overview over vulnerable DNS hosts, thus encouraging system administrators to properly configure their servers. Open Resolver projects carried out by different groups of individuals, the Shadowserver Foundation among others. Aggregated reports are available at `https://dnsscan.shadowserver.org/` and `http://openresolverproject.org/`. Raw data is only provided by request.

5. ENTITY CLASSIFICATION

In this section we will propose a categorization model for scanning entities. The main purpose of this model is to give a compact but extensive description of a scanning entity, so that both similarities and differences between individual entities become clearly visible. The following model is heavily based on examining the behavior and the background of a particular entity. We decided to classify entities by the following attributes:

Organization/institution says a lot about the scanning entity. In fact, further attributes heavily depend on the type of organization the scanning entity is representing. Clear attribution is only possible if the entity is publicly known. However, in most cases scanning entities choose to remain anonymous. In this case we use a generic term "individual". We propose the following categories:
1) *academic*
2) *governmental*
3) *commercial*
4) *individuals(s)*

Intention behind Internet-wide scans is the most important aspect when speaking about legality and ethics. There are basically two major reasons for performing a scan – finding vulnerabilities to exploit and doing research/auditing to provide a global overview. But despite that, we found out, that Shodan and HACIENDA clearly stand out from other entities. Shodan neither does research nor wants to exploit any vulnerabilities, whereas HACIENDA is a reconnaissance program, which is different from the common understanding of vulnerability exploits as a means to gaining personal profit. We also want to emphasize the presence of *military* operations among port scanning activities. Thus, we chose to add a third category:
1) *exploit*
2) *research / security audit*
3) *discovery/reconnaissance*

Spatial distribution of scan sources is another attribute which defines a scanning entity. As an example, Internet Census 2012 was performed from a worldwide botnet of constrained devices.
1) *single host / local cluster*
2) *wide-area/globally distributed*

Publication of results is important for researchers who either want to use the data for their own work, or to verify the work based on this data. As described in section 2.3 particularly metadata may give crucial knowledge about the targeted hosts. We distinguish three degrees of publication extent:
1) *undisclosed*
2) *aggregated report* (w/o raw data)
3) *complete*

Used software category tells, whether these tools are available to public:
1) *enterprise*
2) *publicly available* (incl. custom builds)

Timing describes if a scan initiative is still present or has taken place in the past:
1) *passed*
2) *ongoing*

Table 1 presents classification of the aforementioned scanning entities. Note, that for SIPscan and Conficker we only consider malicious intents. Security auditing must be examined separately. Additionally, some behavior patterns can be seen: An academic institutions is most likely doing research and hence will also publish scan results (University of Michigan being a good example). Attacks on vulnerable services (here: Conficker, SIPscan) include a distributed approach – botnets. By contrast, Internet Census 2012 bears all the hallmarks of a research undertaking, but it utilized a botnet which is more common for malicious activities. Almost every entity is using publicly available software with the single exception of Shodan. However, Shodan also processes XML reports from Nmap and masscan which were uploaded by ScanHub users.

6. LEGAL STATUS & ETHICS

As we already mentioned before, port scanning affects both scanning and scanned entities. Port scanning has been a controversial topic for more than two decades.[23] Some consider it a harmless examination, while others regard port scanning as an intrusion. Roughly summarized, there are two contrary points on this topic:[5]

doi: 10.2313/NET-2015-09-1_11

	CAIDA	UMich	Sonar	SIPscan	Conficker	IC2012	Shodan	HACIENDA	Open Resolver
organization									
academic	■	■							
governmental	■							■	
commercial	■		■				■		
individual(s)			■	■	■	■			■
intention									
exploit				■	■			■	
discovery/recon							■	■	
research/audit	■	■	■			■			■
distribution									
single/local	■	■	■				■	■	
wide/global				■	■	■			■
publication									
undisclosed				■	■			■	
aggregated									■
complete	■	■	■			■	■		■
used software									
enterprise							■		
public	■	■	■	■	■	■	■	■	■
timing									
passed				■		■			
ongoing	■	■	■		■		■	■	■

Table 1: An attempt to classify Internet-wide scan initiatives and scanning entities

- *A port scan is the precursor to an attack.*
- *A port scan is an act of curiosity about services, which are offered to the public anyway.*

The ethical debate apparently looks like a matter of taste at the first sight, however, it is the key to understanding the legal situation concerning the port scanning in most countries. In the following we will briefly introduce the legal situation in a few chosen areas of jurisdiction.

One of the most famous legal cases involving port scanning is Moulton v. VC3. Network engineer Scott Moulton was appointed to maintaining a municipal emergency network in Georgia, US. Moulton was concerned about network security as he launched a port scan for security auditing a network he had been previously setting up. Eventually his scan reached the IP addresses of the consulting firm VC3. Since Moulton did not conceal his identity, VC3 informed the police, whereupon Moulton was arrested. He was sued for violation of the Computer Fraud and Abuse Act (CFAA) as well as the Georgia Computer Systems Protection Act. After months of litigation, the court ruled, Moulton did not violate CFAA, hence all charges against him were dropped.[23] Severe violation of CFAA may be punished with up to 10 years of imprisonment [18 U.S.C. § 405(c)(1)(A)].

Laws dealing with computer fraud exist all around the world, however explicit mentioning of port scanning is far less likely to find. In 2006 the UK Computer Misuse Act 1990 was amended with the Police and Justice Act 2006, that prohibited "supplying or obtaining articles for use in computer misuse offences".[Computer Misuse Act 1990 (amended by Police and Justice Act 2006) F63A] For example, downloading the "network stress testing application" Low Orbit Ion Cannon (LOIC) used mainly for denial-of-service attacks (DoS) is a crime within the UK jurisdiction. The legal phrasing "computer misuse offences" is vague, thus any port scanning tool may be considered as such an "article". By contrast, the § 202c of the German penal code clearly prohibits the use of *purpose-built intrusion* software.[§ 202c StGB (German penal code) Abs. 1] This, however, is also open to interpretation, since the use of dubious scan methods such as idle scan or FTP bounce may be considered an intrusion. Another point is afflicting damage to the targeted host(s). Aggressive scanning as well as some scan parameters may overload the network or even crash some hosts. This can be surely considered as an unintentional tort (prosecuted in most countries with a fine) or even a DoS attack (computer fraud laws apply).

Aside from legal implications, there are several ISPs that prohibit port scans. US-based cable company Comcast explicitly prohibits port scanning for its customers.[7] German ISP Deutsche Telekom does not allow establishing connection to a remote host as an end in itself.[9] Violating terms of use will most likely result in contract void, but rather rarely in further legal disputes.

The specific character of Internet-wide scanning implies, that a scanning party may break several laws in multiple countries. Depending on the severity of law violation and international criminality treaties, the home country may be obliged to extradite the criminal suspects. Scottish system administrator Gary McKinnon was accused of computer fraud by the US authorities after port scanning and exploiting vulnerabilities of several US military organizations and nearly faced extradition[28] until the order was withdrawn in 2012 by the UK Home Secretary.[19] Though, this was a clear case of intrusion, port scans are definitely portraying a suspicious activity for military organizations – whatever consequences that means.

doi: 10.2313/NET-2015-09-1_11

7. CONCLUSION

In this paper we have introduced the state-of-the-art publicly available software tools for Internet-wide scans. The classic among port scanners Nmap, while highly tunable and accurate, cannot reach the speed of the newcomers ZMap and masscan – specialized software for running /0 scans.

In order to answer the question "Who is scanning the Internet?" we presented various organizations and individuals who are openly performing port scans on a large scale. We learned that they pursue different goals, with the most interested in security/exploits, but some of them also being strikingly different, namely Shodan and the HACIENDA program. In order to systematically organize scanning entities by their attributes we proposed a classification model. The model exhibits some interesting behavior patterns concerning the various attributes of individual entities.

Finally, we discussed some some legal and ethical aspects of Internet-wide scanning. We focused on striking abuse allegations due to port scanning with further elaborating on consequences on legality of performing it on a large scale. We concluded that the vagueness of legal acts may present a serious burden for those willing to perform such scans on their own.

8. REFERENCES

[1] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman. Zippier zmap: internet-wide scanning at 10 gbps. In *Proceedings of the 8th USENIX Workshop on Offensive Technologies*, 2014.

[2] J. Blagdon. Take a snapshot of the entire internet in just 44 minutes with ZMap scanner. *The Verge*, 2013.

[3] B. Burns, J. S. Granick, S. Manzuik, P. Guersch, D. Killion, N. Beauchesne, E. Moret, J. Sobrier, M. Lynn, E. Markham, C. Iezzoni, and P. Biondi. *Security Power Tools*. O'Reilly, 2007.

[4] CAIDA. Frequently asked questions about caida. `http://www.caida.org/home/about/faq.xml`, January 2015.

[5] Chaos Computer Club Cologne e. V. Warum wir nichts gegen portscans haben. `http://koeln.ccc.de/ablage/portscan-policy.xml`.

[6] R. Clayton. Brute force attacks on cryptographic keys. `http://www.cl.cam.ac.uk/~rnc1/brute.html`.

[7] Comcast. Acceptable use policy. `http://business.comcast.com/customer-notifications/acceptable-use-policy`, February 2013.

[8] A. Dainotti, A. King, F. Papale, A. Pescape, et al. Analysis of a/0 stealth scan from a botnet. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 1–14. ACM, 2012.

[9] Deutsche Telekom. Allgemeine geschäftsbedingungen. festnetz- und mobilfunkanschlüsse (privatkunden). `http://www.telekom.de/dlp/agb/pdf/40810.pdf`, December 2012.

[10] P. Ducklin. Welcome to Zmap, the "one hour turnaround" internet scanner. *Naked Security*, 2013.

[11] Z. Durumeric, M. Bailey, and J. A. Halderman. An internet-wide view of internet-wide scanning. In *USENIX Security Symposium*, 2014.

[12] Z. Durumeric, E. Wustrow, and J. A. Halderman. Zmap: Fast Internet-wide scanning and its security applications. In *USENIX Security*, pages 605–620, 2013.

[13] R. D. Graham. The C10M problem. `http://c10m.robertgraham.com/p/manifesto.html`, 2013.

[14] R. D. Graham. Masscan: designing my own crypto. `http://blog.erratasec.com/2013/12/masscan-designing-my-own-crypto.html`, December 2013.

[15] R. D. Graham. Masscan: Mass IP port scanner. `https://github.com/robertdavidgraham/masscan`, 2013.

[16] R. D. Graham. Masscan: the entire Internet in 3 minutes. `http://blog.erratasec.com/2013/09/masscan-entire-internet-in-3-minutes.html`, September 2013.

[17] IC2012. Internet Census 2012: Port scanning /0 using insecure embedded devices. `http://internetcensus2012.bitbucket.org/`, 2012.

[18] Internet Society. Measurements | world ipv6 launch. `http://www.worldipv6launch.org/measurements/`, July 2015.

[19] M. Kennedy. Gary McKinnon will face no charges in UK. `http://www.theguardian.com/world/2012/dec/14/gary-mckinnon-no-uk-charges`, December 2012.

[20] J. Kirsch, C. Grothoff, M. Ermert, J. Appelbaum, L. Poitras, and H. Moltke. NSA/GCHQ: The HACIENDA Program for Internet Colonization. `http://heise.de/-2292681`, August 2014.

[21] C. Leckie and R. Kotagiri. A probabilistic approach to detecting network scans. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 359–372. IEEE, 2002.

[22] C. B. Lee, C. Roedel, and E. Silenok. Detection and characterization of port scan attacks. *Univeristy of California, Department of Computer Science and Engineering*, 2003.

[23] G. Lyon. *Legal Issues*, chapter 1. Nmap Project `http://nmap.org`, 2009.

[24] G. Lyon. *Nmap Reference Guide*, chapter 15. Nmap Project `http://nmap.org`, 2009.

[25] V. Paxson. Strategies for sound Internet measurement. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 263–271. ACM, 2004.

[26] E. Raftopoulos, E. Glatz, X. Dimitropoulos, and A. Dainotti. How dangerous is internet scanning? In *Traffic Monitoring and Analysis*, pages 158–172. Springer, 2015.

[27] Rapid7. Project sonar by rapid7. `https://sonar.labs.rapid7.com/`, 2013.

[28] G. Roberts. Gary McKinnon: Inside the head of a super hacker. `http://www.independent.co.uk/news/science/gary-mckinnon-inside-the-head-of-a-super-hacker-407656.html`, July 2006.

[29] Sciengines. Copacobana: A Codebreaker for DES and other Ciphers. `http://www.sciengines.com/copacobana/index.html`.

[30] Shodan. Shodan: The search engine for Internet of Things. `https://www.shodan.io/`, 2015.

[31] I. Thomson. New tool lets single server map entire internet in 45 minutes. *The Register*, 2013.

[32] Y. G. Zeng, D. Coffey, and J. Viega. How vulnerable are unprotected machines on the internet? In *Passive and Active Measurement*, pages 224–234. Springer, 2014.

ISBN 978-3-937201-50-4
DOI 10.2313/NET-2015-09-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)