Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III - Agrar- und Ernährungswissenschaften,
Geowissenschaften und Informatik
Institut für Informatik

Bachelorarbeit

Möglichkeiten zur Deanonymisierung anhand von Metadaten für zentrale & föderative Anwendungen

am Beispiel von WhatsApp, Matrix & der Corona-Warn-App

Dominik Weiß

Betreuer:
Dr. Sandro Wefel
apl. Prof. Dr. Klaus Reinhardt

14. Januar 2021
Halle (Saale), Deutschland

CampusPUBLIK
Schriften zur politischen Bildung

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich bei der Erstellung meiner Abschlussarbeit für den Bachelor unterstützt haben. Diese Arbeit ist das Ergebnis meines Bachelorstudiums an der Martin-Luther-Universität Halle-Wittenberg. In den drei Jahren meines Studiums erlaubte mir die Universität einen Einblick in die verschiedensten Bereiche der Informatik, wobei mich gerade das Gebiet der IT-Sicherheit reizte.

Zunächst möchte ich meinen Betreuern Dr. Sandro Wefel und apl. Prof. Dr. Klaus Reinhardt danken. Ein besonderer Dank gilt dabei Dr. Sandro Wefel, der mir nicht nur im Rahmen des Studiums die Grundlagen der IT-Sicherheit beigebracht und dabei mein Interesse für das Gebiet geweckt hat, sondern mir auch maßgeblich als Betreuer bei Fragen und Problemen im Verlauf dieser Arbeit zur Seite stand. Von Ihm stammt zudem das Thema dieser Arbeit, welches gerade zu Beginn eine große Herausforderung für mich darstellte. Im Verlauf der Arbeit lehrte mich Dr. Sandro Wefel diese Probleme eigenständig zu überkommen, wobei ich mich stets auf Ihn verlassen konnte, sollte ich an einer Stelle einen Rat benötigen oder den Fokus verlieren. Unter seiner Betreuung lernte ich ein größeres Vertrauen in meine eigenen Fähigkeiten zu haben, auch komplexere Probleme zu überwinden und schließlich meine Kenntnisse im Bereich der IT-Sicherheit auszubauen und in dieser Arbeit anzuwenden.

Des Weiteren möchte ich meinen Freunden danken, die während meiner Zeit in Halle stets für mich da waren und mit denen ich das Glück hatte, unglaublich schöne Erinnerungen zu teilen. Ein besonderer Dank gilt Julian Müller, David Grölle, Sean William Cean Chadwick und Jann Lucas Pischke, die mich beim Erstellen meiner Arbeit unterstützt haben und diese Korrektur gelesen haben.

Zuletzt möchte ich mich bei meinen Eltern Dana und Enrico Pape bedanken, welche mich stets in meinem Leben und den Entscheidungen, die ich getroffen habe, unterstützt haben. Beide waren immer für mich da und ich konnte mich blind auf sie verlassen. Besonders meine Mom hat mir gezeigt, wie man durch harte Arbeit und den Glauben an sich selbst alles erreichen kann, was man sich vornimmt und ich bin unglaublich stolz, ihr Sohn zu sein.

Abstract

Die Bedeutung von mobilen internetbasierten Messenger-Diensten und anderen Applikationen stieg in den letzten Jahrzehnten enorm an. Mit ihnen steigt die Relevanz der Sicherheit von Daten und Metadaten. Dabei werden Metadaten zunehmend relevanter für die Privatsphäre von Nutzern, erfahren dabei jedoch nicht den gleichen Schutz wie die Daten der Nutzer selbst. Diese Arbeit betrachtet die Sicherheit der Metadaten und damit die Möglichkeit der Deanonymisierung von Nutzern im Bereich der Instant Messenger, praktisch an den Beispielen von WhatsApp und Matrix, sowie der aktuell relevanten Corona-Warn-App zur Kontaktverfolgung. Es werden die Aspekte der verschlüsselten Kommunikation für zentrale und föderative Systeme betrachtet und analysiert. Dabei konnten in den durchgeführten Tests Informationen über die Sicherheit der Paketübertragung gewonnen werden, welche zu teilen auch Rückschlüsse auf die zusätzlich übertragenen Metadaten zuließen. Die verschlüsselten Pakete selbst legten dabei ebenso Metadaten der Kommunikation offen. So konnte gezeigt werden, dass eine Verschlüsselung der Daten allein nicht genügt, um sämtliche Metadaten zu verschleiern, wobei die Ursache hierbei auch in den grundlegenden Protokollen TCP und IP zu begründen ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

Abkürzungsverzeichnis

1. Einleitung

Unsere Daten sind verschlüsselt – unsere Metadaten sind es nicht. In den letzten Jahren ist die Relevanz von Messengern und anderen Applikation in unserem Alltag zunehmend gestiegen und mit ihr die Bedeutung des Schutzes unserer Daten. Dabei bedeutet Sicherheit nicht gleich Sicherheit. Wo die eigentlich versandten Daten durch *state-of-the-art* Krypto-Verfahren stetig besser verschlüsselt werden, steigt gleichzeitig die Anzahl der übermittelten und ausgewerteten Metadaten. Mit den Enthüllungen in Bezug auf die Arbeit der NSA durch Edward Snowden im Jahre 2013 [Gre13] wurde die Wichtigkeit von Metadaten für Geheimdienste besonders deutlich.

> *"Metadata absolutely tells you everything about somebody's life. If you have enough metadata you don't really need content."*
> – Stewart Baker, former NSA general counsel [Sch15]

Ebenso haben Unternehmen und Regierungen ein Interesse an Metadaten, weshalb deren Schutz in Anwendungen, welche wir alltäglich benutzen, umso wichtiger ist. In dieser Arbeit soll daher der aktuelle Stand der Sicherheit von Messengern wie WhatsApp und Matrix, aber auch der Kontaktverfolgungsapp Corona-Warn-App (CWA), insbesondere im Bezug auf Metadaten betrachtet werden. Die anfangs genannte These wird im Verlauf der Arbeit begründet und anhand der betrachteten Applikationen gestärkt. Dabei entsteht ebenfalls die Frage, ob föderative Systeme eine sicherere Alternative im Vergleich zu zentralen Anwendungen darstellen.

1.1. Ziel & Abgrenzung dieser Arbeit

Gegenstand dieser Arbeit ist eine Evaluation der Möglichkeiten der Deanonymisierung mittels Metadaten im Bereich der Nachrichtenübertragung von Instant Messenger (IM) wie WhatsApp, Matrix oder anderen, sowie der Corona-Warn-App. Dabei werden die Verschlüsselungen der Nachrichten nicht ausführlich betrachtet, da diese Verfahren bereits in Arbeiten wie [BM19],[CL19],[SZ20],[SCS05] geprüft wurden. Der Inhalt der übertragenden Nachrichten wird daher als sicher verschlüsselt im Rahmen einer End-to-End (E2E)-Verschlüsselung angesehen. Des Weiteren werden keine technischen Sicherheitsschwachstellen betrachtet, da diese Fehler oft durch Updates schnell behoben werden können. Vielmehr wird dargelegt, dass all diese Messenger in ihrer Konzeption oder Umsetzung keine völlige Sicherheit bieten, besonders in Anbetracht von Metadaten. Betrachtet werden die Szenarien, in welchen ein Angreifer mittels eines Man-in-the-Middle-Angriffs den Übertragungskanal abhört, sowie die Betrachtung, dass der Anbieter des Servers nicht vertrauenswürdig ist. Der Inhalt der Nachrichten ist dabei weniger Gegenstand dieser Arbeit, als mehr die zugehörigen Metadaten zu diesen Nachrichten und wie diese von Angreifern oder nicht vertrauenswürdigen Einrichtungen genutzt werden können, um den Nutzer zu deanonymisieren. Somit liegt ein Hauptaugenmerk der Arbeit auch auf dem Schutz der Privatsphäre der Nutzer.

Insbesondere wird dabei der Messenger WhatsApp, das Open-Source-Messenger Projekt Matrix, als auch die Corona-Warn-App in Bezug auf die Sicherheit ihrer Metadaten überprüft. Somit möchte diese Arbeit folgende Fragen beantworten:

- Wie sicher sind aktuelle Messenger, insbesondere in Bezug auf Metadaten?
- Wie sicher ist WhatsApp (Web) und dessen Metadaten?
- Wie sicher ist Matrix und dessen Metadaten?
- Wie sicher ist die Corona-Warn-App und deren Metadaten?

1.2. Verwandte Arbeiten

Das übergeordnete Gebiet der *IT-Sicherheit* ist als Teilgebiet der *Informatik* vergleichsweise neu, weist aber gerade in den Bereichen der *Kryptologie* eine Vielzahl von wissenschaftlichen Arbeiten auf. Das Werk von Eckert, IT-Sicherheit Konzepte - Verfahren - Protokolle., 2014 [Eck14] bietet einen Einblick in den Bereich der IT-Sicherheit und ist daher für die theoretischen Grundlagen dieser Arbeit relevant. Gerade im letzten Jahrzehnt stieg das Bewusstsein für sichere Kommunikation, nicht zuletzt aufgrund des rasanten Anstiegs der Verbreitung von Messengern (siehe Abschnitt 4.1) und damit das Forschungsvolumen und die Anzahl an Veröffentlichungen in dem Gebiet. Betrachtet man hingegen die Sicherheit der Metadaten jener verschlüsselten Nachrichten, so finden sich weitaus weniger Veröffentlichungen, was im Kontrast steht zur realen sicherheitstechnischen Relevanz. Zwar zeichnet sich ein Trend ab, da zum Sicherheitsaspekt der Metadaten von Kommunikation in den letzten Jahren mehr veröffentlicht wurde als in den Jahren zuvor, dennoch kann man davon ausgehen, dass noch viele Erkenntnisse in diesem Bereich möglich sind. Nicht zuletzt finden sich Überschneidungen zu dem Gebiet der *Data Science* im Bereich des *Datamining*. Die Relevanz von Metadaten wird hier besonders deutlich und Erkenntnisse können ebenso im Bereich der Kommunikation angewandt werden, wobei die Sicherheitsaspekte der Daten von Nutzern zunehmend eine größere Rolle spielen. Im Kern werden Aussagen über große Datenmengen (*Big Data*) getroffen. Diese sind ebenso relevant für die reelle sicherheitstechnische Betrachtung von Messengern, konnten in dieser Arbeit jedoch nicht praktisch getestet werden und werden daher folgend nicht weiter erläutert.

Einen wichtigen Beitrag auf dem Gebiet der Analyse von Metadaten des Netzwerkverkehrs von Messengern lieferten Bahramali u. a., *„Practical Traffic Analysis Attacks on Secure Messaging Applications“*, 2020 [Bah+20]. Es wird gewarnt, wie bei weit-verbreiteten Messengern (Telegram, Signal, WhatsApp,...) Informationen gewonnen werden können – trotz sicherer Verschlüsselung – durch Analyse des Netzwerkverkehrs. Konkret wird eine Methode gezeigt, wie man Administratoren und Mitglieder von verschlüsselten Kanälen mit großer Wahrscheinlichkeit bestimmt, was von Regierungen und Geheimdiensten missbraucht werden kann und damit eine reale Bedrohung darstellt. Dem zugrunde liegt das Problem, dass diese Messenger keine Mechanismen zur Verschleierung statistischer Merkmale ihrer Kommunikation etablieren. Die Autoren bieten ein Open-Source-System

(*IMProxy*) als Gegenmaßnahme, welches das Analysieren des Netzwerkverkehrs zumindest erschwert, auf Kosten zusätzlicher Overheads und Latenz. Es wird deutlich, dass eine sichere E2E-Verschlüsselung allein nicht genügt, um Informationen von Nutzern zu schützen, worauf diese Arbeit ebenfalls aufmerksam machen möchte.

Die Masterarbeit von Kaczmarczyck, *„Metadata Reduction for the Signal Protocol“*, 2017 [Kac17] behandelt den Messenger Signal. Es werden zwei Möglichkeiten zur Informationsgewinnung über Nutzer und Erkennung von Vernetzungen zwischen Nutzern beschrieben: Kontakterkennung durch das Hochladen des Adressbuches und durch Kommunikationspartner von Konversationen. Für beide Probleme erläutert Kaczmarczyck jeweils Methoden, um die Vernetzung zu verschleiern. Seit 2016 wird das Signal Protokoll ebenfalls von dem Messenger WhatsApp verwendet [ORP20], welchen wir in dieser Arbeit betrachten. Das es bereits eine Abschlussarbeit auf diesem Gebiet gibt, welche konkret Signal thematisiert, grenzte den Bereich dieser Arbeit und die Auswahl an möglichen zu betrachtenden Messengern ein, was ein Grund dafür war, die Metadaten von Signal in dieser Arbeit nicht zu analysieren. Dennoch wird im Hauptteil kurz die Sicherheit von Signal erläutert (Abschnitt 4.4.3).

Zuletzt eine kurze Anmerkung zum Artikel von Baumgärtner u. a., Mind the GAP: Security & Privacy Risks of Contact Tracing Apps, 2020 [Bau+20]. Dieser behandelt die Möglichkeiten zur Deanonymisierung von Kontaktvervolgungsapps, welche auf dem ”Google/Apple Proposal (GAP)” basieren, insbesondere der Corona-Warn-App. Trotz der gleichen Thematik wurde der Artikel nicht aufgegriffen, da es keine Reviews zu dem Artikel gibt und dieser in diversen Podcast und Beiträgen [PM20],[TP21] für unwissenschaftliches Arbeiten und einer zu wertenden Ausdrucksweise kritisiert wurde.

1.3. Aufbau der Arbeit

Als Grundlage für die im Hauptteil durchgeführten Testdurchläufe und besprochenen Aspekte der Sicherheit von IM und der CWA werden im Kapitel 2 die wichtigen Begriffe der IT-Sicherheit und von relevanten Netzwerkprotokollen erläutert. Ebenso wichtig für diese Arbeit ist der generelle Aspekt der Metadaten, welcher im Kapitel 3 definiert und im Kontext der Netzwerkübertragung eingeordnet wird. Anschließend werden im Kapitel 4 die aktuellen Konzepte und Verfahren für die Sicherheit von Instant Messenger betrachtet und für einige weitere IM erläutert. Kapitel 5 beschreibt die Methoden zur Durchführung der Testdurchläufe in dieser Arbeit, wobei die Paketaufzeichnung durch Wireshark zentral ist. Ebenso werden hier die verwendeten Testsysteme und der Ablauf der Test beschrieben, damit diese nachvollziehbar und wiederholbar sind. Der praktische Teil dieser Arbeit behandelt die Aufzeichnung des Netzwerkverkehrs von WhatsApp Web (Kapitel 6), Matrix Element (Kapitel 7) und der Corona-Warn-App (Kapitel 8). Die aufgezeichneten Pakete werden jeweils anhand der einsehbaren Metadaten ausgewertet. Schlussendlich werden Möglichkeiten zur Deanonymisierung von allen drei betrachteten Applikationen diskutiert. Die darauf aufbauend gewonnenen Erkenntnisse, sowie die zuvor formulierten Forschungsfragen werden im letzten Kapitel 9 besprochen.

2. Grundlagen der IT-Sicherheit & Netzwerkprotokolle

Diese Arbeit ist hauptsächlich in das Teilgebiet der *Informatik* – der *IT-Sicherheit* – einzuordnen. Daher werden zunächst einige grundlegende Begriffe der IT-Sicherheit geklärt, sowie Konzepte, welche als Grundlage im Verlauf der Arbeit notwendig sind. Zusätzlich werden diese theoretischen Grundlagen folgend im Kontext von Messengern betrachtet und damit verknüpft. Ein Großteil dieser Grundlagen stammt aus dem Buch von Eckert, IT-Sicherheit Konzepte - Verfahren - Protokolle., 2014 [Eck14].
Des Weiteren ist ein wesentlicher Punkt in dieser Arbeit die Analyse von Netzwerkprotokollen, weshalb die Netzwerkschicht und das ISO/OSI-Referenzmodell erläutert werden.

2.1. Grundlegende Begriffe der IT-Sicherheit

2.1.1. Objekte & Subjekte

Definition 2.1.1 (Objekt)**.** Die Aufgabe von IT-Systemen ist es, Informationen zu speichern und zu verarbeiten. Die Information ist aber abstrakt und wird in Form von Daten bzw. Datenobjekten repräsentiert [Eck14, S. 4]. Man unterscheidet:

- **passive Objekte** : z.B. Bilder, Audiodaten, Programme oder kryptographische Schlüssel, abgelegt auf der Festplatte oder in einer Datenbank
- **aktive Objekte** : laufende Programme (Prozesse)

"Informationen und die Objekte, die sie repräsentieren, sind schützenswerte Güter (engl. asset) eines Systems." [Eck14, S. 4]

Definition 2.1.2 (Subjekt)**.** "Die Benutzer eines Systems und alle Objekte, die im Auftrag von Benutzern im System aktiv sein können, wie z.B. Prozesse, Server und Prozeduren, werden als die Subjekte des Systems bezeichnet." [Eck14, S. 5]

In dem konkreten Fall von Messengern sind die Subjekte somit die Nutzer, welche Nachrichten verschicken. Diese Nachrichten wiederum sind passive Objekte. Beide gilt es zu schützen.

2.1.2. Informationen als Schützenswerte Güter in IT-Systemen

Hauptfunktion von Messengern ist die Nachrichten- oder Informationsübermittlung zwischen zwei oder mehreren Personen. Dabei muss die übermittelte Information in der IT-Sicherheit gesondert definiert werden:

Definition 2.1.3 (Information)**.** "Die Information, die durch ein Datum repräsentiert wird, ergibt sich aus einer festgelegten Interpretationsvorschrift." [Eck14, S. 4]

Diese Informationen werden als (Daten)objekte ausgetauscht und sind somit schützenswerte Güter. Jedoch geschieht diese Einschätzung oft nur unvollständig bei vielen Messengern: zwar werden die übertragenden Nachrichten verschlüsselt und damit geschützt, oft aber nicht die mit ihnen verbundenen Metadaten, was später noch genauer betrachtet wird.

2.1.3. Zugriffsrechte & Informationskanäle

Als nächstes werden die Begriffe Zugriff und Kanal definiert. Beide werden benötigt, um zwischen erlaubten (autorisierten) und unerlaubten (unautorisierten) Informationsaustausch unterscheiden zu können.

Definition 2.1.4 (Zugriff)**.** "Eine Interaktion zwischen einem Subjekt und einem Objekt in einem System, durch die ein Informationsfluss zwischen Subjekt und Objekt auftritt, nennen wir einen Zugriff auf die Information." [Eck14, S. 5]

Ein Zugriff auf ein Datenobjekt entspricht somit auch einem Zugriff auf die dadurch repräsentierte Information [Eck14, S. 5]. Nun muss festgelegt werden, wie und von wem auf dieses Objekt zugegriffen werden darf.

Definition 2.1.5 (Zugriffsrecht/ Autorisierung)**.** "Für den Zugriff auf die zu schützende Information bzw. auf die sie repräsentierenden Daten eines IT-Systems sind Zugriffsrechte festzulegen und an die Subjekte zu vergeben. Besitzt ein Subjekt die Berechtigung zum Zugriff auf eine Information bzw. auf ein Datenobjekt, so sagen wir, dass das Subjekt zu diesem Zugriff autorisiert ist." [Eck14, S. 5]

Somit werden oft unterschiedliche Arten von Subjekten mit unterschiedlichen Zugriffsrechten benötigt. Ein recht intuitives Beispiel wäre, dass ein Nutzer nur auf eine Teilmenge von Objekten autorisiert zugreifen kann, für welche der Administrator sämtliche Zugriffsrechte hat. Aus datenschutzrechtlicher Perspektive kann es aber auch durchaus sinnvoll, oder notwendig sein, dass ein Nutzer auf Objekte zugreifen kann (beispielsweise seine eigenen Nachrichten), während ein Administrator keine Zugriffsrechte für diese persönlichen Informationen hat. Dazu ebenfalls später mehr.

Informationen oder Daten werden zwischen Objekten über sogenannte *Kanäle* übertragen. Dazu zählen unter anderem die Speicherung in einer Datei oder Datenbank (Speicherkanal), sowie die Übertragung in einem Netzwerk und Übergabe von Kommandozeilenparametern. Es wird zwischen zwei Arten von Kanälen unterschieden:

Definition 2.1.6 (Legitimer Kanal)**.** "Die legitimen Kanäle sind diejenigen, die ein Subjekt in der Regel für den Informationsaustausch nutzt." [Eck14, S. 5] Sie sind für diesen Zweck vorgesehen.

Definition 2.1.7 (Verdeckter Kanal)**.** "Verdeckte Kanäle (engl. covert channel) sind solche, die nicht für einen Informationstransfer vorgesehen sind, aber dazu missbraucht werden können." [Eck14, S. 5]

Dabei ist zu erwähnen, dass allgemein eine höhere Komplexität im IT-System verdeckte Kanäle begünstigt [Eck14, S. 5]. Somit können je nach festgelegten Schutzzielen – welche im Anschluss definiert werden – verdeckte Kanäle oftmals eine Sicherheitslücke darstellen und genutzt werden, um schützenswerte Informationen von Nutzern preiszugeben. Ziel sollte es somit allgemein sein, verdeckte Kanäle zu schließen und den Informationsgewinn über diese zu minimieren.

2.1.4. Informationssicherheit & zugehörige Begriffe

Definition 2.1.8 (Funktionssicherheit)**.** "Unter Funktionssicherheit (engl. safety) eines Systems verstehen wir die Eigenschaft, dass die realisierte Ist-Funktionalität der Komponenten mit der spezifizierten Soll-Funktionalität übereinstimmt." [Eck14, S. 6]

Dieses Ziel des Einschränkens von verdeckten Kanälen, welches aus mehreren Aspekten besteht, kann man allgemein unter dem Begriff des Erlangens der Informationssicherheit zusammenfassen.

Definition 2.1.9 (Informationssicherheit)**.** Die Informationssicherheit (engl. security) ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keiner unautorisierten Informationsveränderung oder -gewinnung führen." [Eck14, S. 6]

Welche Aktionen auf Objekte und von welchem Subjekt konkret erlaubt sind, hängt von der jeweiligen Spezifikation des Systems, sowie den formulierten Schutzzielen ab.

Definition 2.1.10 (Verlässlichkeit)**.** "Unter der Verlässlichkeit (engl. dependability) eines Systems verstehen wir die Eigenschaft, keine unzulässigen Zustände anzunehmen (Funktionssicherheit) und zu gewährleisten, dass die spezifizierte Funktion zuverlässig (engl. reliability) erbracht wird." [Eck14, S. 7]

2.1.5. Datenschutz

Definition 2.1.11 (Datensicherheit)**.** "Die Datensicherheit (engl. protection) ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keinem unautorisierten Zugriff auf Systemressourcen und insbesondere auf Daten führen." [Eck14, S. 6]

Ein Unterbegriff der Datensicherheit, welcher nachfolgend für diese Arbeit von Bedeutung sein wird, ist der Datenschutz.

Definition 2.1.12 (Datenschutz)**.** "Unter dem Begriff Datenschutz im engeren Sinn (engl. privacy) [...] versteht man die Fähigkeit einer natürlichen Person, die Weitergabe von Informationen, die sie persönlich betreffen, zu kontrollieren." [Eck14, S. 6]

Dafür erfordert der Datenschutz bestimmte Regularien und gesetzliche Vorgaben, in Deutschland das Bundesdatenschutzgesetz [BDSG],[Eck14, S. 6]. Zudem ist es schwer bis unmöglich, das globale Internet und die Kommunikation in diesem in staatliche Bereiche einzuteilen und nationale Gesetze umzusetzen. Das Problem des Datenschutzes war in den letzten Jahren für viele Nutzer äußerst relevant, gerade in den Apps der sozialen Medien, wie Instagram, Facebook und vielen weiteren. Um zu kontrollieren, welche Daten von Nutzern gesammelt und weiterverkauft, oder ohne deren Wissen ausgewertet werden, benötigt es strenge Gesetze und Regularien, wobei die Anzahl der dazu erlassenen Gesetze in den letzten Jahren zunahm [Wha19].

2.2. Schutzziele

Allgemein können für ein funktionssicheres System folgende Schutzziele festlegt werden, welche umgesetzt und überwacht werden müssen. Dabei ist dies nur eine mögliche Auswahl von weiteren Sicherheitszielen:

- Authentizität 2.2.1
- Integrität (Datenintegrität) 2.2.2
- Vertraulichkeit (Informationsvertraulichkeit) 2.2.3
- Verfügbarkeit 2.2.4
- Verbindlichkeit 2.2.5
- Anonymität & Privatsphäre 2.2.6

Nachfolgend werden diese zunächst genauer erläutert und dann im späteren Teil der Arbeit geprüft, wie gut ausgewählte Messenger diese Schutzziele umsetzen, insbesondere Authentizität, Anonymität und Privatsphäre. Die Gliederung und wesentliche Informationen entstammen [Eck14, S. 8–14].

2.2.1. Authentizität

Definition 2.2.1 (Authentizität)**.** "Unter der Authentizität eines Objekts bzw. Subjekts (engl. authenticity) verstehen wir die Echtheit und Glaubwürdigkeit des Objekts bzw. Subjekts, die anhand einer eindeutigen Identität und charakteristischen Eigenschaften überprüfbar ist." [Eck14, S. 8]

Diese Eigenschaften werden mittels einer Authentifikation (engl. authentication) überprüft [Eck14, S. 8]. Besteht das zu prüfende Subjekt oder Objekt diese Überprüfung, so gilt es als authentifiziert. Wie oft diese Überprüfung stattfindet, ist je nach System unterschiedlich. Oft muss dabei eine Balance zwischen Sicherheit und Nutzbarkeit gefunden werden. In der Praxis erfolgt oft nur eine einmalige Authentifizierung pro Session, oder sogar nur pro neu zu authentifizierendem Gerät, auf welchem sich ein Nutzer (Subjekt) identifizieren möchte.

Allgemein muss für diese Identifizierung nachgewiesen werden, dass "eine behauptete Identität eines Objekts oder Subjekts mit dessen charakterisierenden Eigenschaften übereinstimmt" [Eck14, S. 8]. Im Falle von den zu betrachtenden Messenger-Apps, steht vor allem die Authentifizierung von Nutzern im Vordergrund. Dabei wird jedem Nutzer ein Account eindeutig zugewiesen, welcher beim erstmaligen Nutzen der App erstellt wird. Um einen Account eindeutig einem Nutzer zuzuweisen, gibt es in der Praxis mehrere Möglichkeiten: die Auswahl eines noch nicht verwendeten und somit einmaligen Nutzernamens, oder die Angabe einer E-Mail oder Telefonnummer, welche auch zur Authentifizierung genutzt werden kann. Um bei der Authentifizierung eines Nutzers dessen charakterisierende Eigenschaften nachzuweisen, werden so genannte *Credentials* verwendet, wobei der Begriff als Abstraktion die in der Praxis verwendeten Verfahren zum

Identitätsnachweis zusammenfasst [Eck14, S. 8]. Beispiele für solche Credentials können klassischerweise Passwörter sein, aber auch Bestätigungscodes, welche an die angegebene E-Mail, Telefonnummer oder eine dritte Authentifizierungsapp gesendet werden, sowie biometrische Merkmale. Seltener und prinzipiell weniger sicher sind persönliche Daten wie das Geburtsdatum oder der Geburtsort, da diese potentiell von Dritten in Erfahrung gebracht werden können. Nicht zuletzt wird oft beim Verlust des Passworts die Methode einer vorher eingerichteten Sicherheitsfrage verwendet, wobei die Antwort auf diese nur der tatsächliche Besitzer des Accounts wissen sollte.

In den letzten Jahren setzte sich gerade die *2-Faktor-Authentifizierung* durch. Dabei werden zwei Authentifizierungsmethoden miteinander verbunden, wobei eine erfolgreiche Authentifizierung nur beim bestehen beider Verfahren erfolgt. Oft wird dies zusätzlich zum Abfragen des Passworts genutzt, indem ein Sicherheitscode verwendet wird. Beide Methoden kombiniert bieten wesentlich mehr Sicherheit, da nun das Erlangen des Passworts eines Nutzers allein nicht genügt, um sich Zugang zu dessen Account zu verschaffen. In der Praxis wird die 2-Faktor-Authentifizierung gerade dann genutzt, wenn sich ein Nutzer mit einem neuen Gerät authentifizieren möchte. Somit kann der Besitz eines Geräts, auf welchem der jeweilige Account authentifiziert ist, auch als eine Art der Authentifizierung gesehen werden. Auch hier gilt erneut: je öfter diese Authentifizierung abgefragt wird, desto sicherer ist diese Methode, auf Kosten der Nutzbarkeit. Banking-Apps fragen zum Beispiel bei jedem Login oder Transaktion einen Bestätigungscode ab. Es kann aber auch die Authentifizierung von Objekten notwendig sein, zum Beispiel von Web-Servern, Access Points oder Code [Eck14, S. 8]. Ziel ist es in dem Fall die Echtheit von Daten zu prüfen, welche durch einen unsicheren Kanal übertragen wurden. Dazu kann unter anderem die Challenge-Response Methode mit RSA oder ElGamal verwendet werden. Zuletzt sei noch zu erwähnen, dass in den meisten Fällen lediglich der Ursprung und damit die Vertrauenswürdigkeit der Quelle geprüft wird, nicht jedoch, ob die angegebene Funktionalität des Objekts mit der tatsächlichen übereinstimmt [Eck14, S. 8]. Die vorliegende Arbeit beschäftigt sich hauptsächlich mit der Authentifizierung von Subjekten.

2.2.2. Datenintegrität

Definition 2.2.2 (Datenintegrität)**.** "Wir sagen, dass das System die Datenintegrität (engl. integrity) gewährleistet, wenn es Subjekten nicht möglich ist, die zu schützenden Daten unautorisiert und unbemerkt zu manipulieren." [Eck14, S. 9]

Ist die Integrität der Daten gewährleistet, kann der Besitzer der Daten davon ausgehen, dass diese unverändert vorliegen und nicht von Dritten unbemerkt verändert wurden. Der Zugriff auf Daten und das Verändern dieser – also die Art der Nutzung – muss dabei durch Rechte festgelegt werden, welche bestimmte Subjekte haben. Diese Rechte müssen überprüft werden. Hat ein Subjekt das Recht zur wohldefinierten Nutzung von Daten, so ist dieses Subjekt für diese Art der Nutzung autorisiert. Je nachdem, welche wohl definierte Methoden für die Nutzung eines Objekts festgelegt wurden, können unterschiedliche Aussagen über die Integrität des Objekts getroffen werden [Eck14, S. 9]. Des Weiteren muss die Manipulation von Daten erkannt werden können. Dafür werden

kryptografische Hashfunktionen verwendet [Eck14, S. 9]. Zu den Daten werden dann entsprechend Hashwerte berechnet.

2.2.3. Informationsvertraulichkeit

Definition 2.2.3 (Informationsvertraulichkeit)**.** "Wir sagen, dass das System die Informationsvertraulichkeit (engl. confidentiality) gewährleistet, wenn es keine unautorisierte Informationsgewinnung ermöglicht." [Eck14, S. 10]

Informationen, welche als vertraulich angesehen werden, können in diesem System nur autorisiert gelesen werden. Dafür ist die Festlegung von Berechtigungen notwendig, sowie das Bestimmen und die Kontrolle autorisierter Informationsflüsse zwischen Subjekten [Eck14, S. 10]. Nur diese Subjekte sollen in der Lage sein, Informationen untereinander auszutauschen in der spezifizierten Weise. Insbesondere sollen unautorisierte Subjekte nicht in der Lage sein, diese Informationen zu gewinnen. Dieses Problem wird als "Confinement Problem" [Eck14, S. 10] bezeichnet. Auch hier findet die Kryptographie Anwendung, indem vertrauliche Informationen verschlüsselt übertragen werden und nur autorisierte Subjekte den Schlüssel zum Entschlüsseln der Informationen besitzen. Die Kontrolle der Zugriffe allein genügt jedoch noch nicht für die Vertraulichkeit der Daten, weshalb so genannte "Labeling-Techniken" [Eck14, S. 11] eingesetzt werden. Dies weist Datenobjekten eine konkrete Sicherheitsstufe zu, wobei Informationen tatsächlich nur an autorisierte Subjekte weitergegeben werden dürfen [Eck14, S. 11]. Dass diese Autorisierung auch wahrgenommen werden kann, wird im nächsten Schutzziel definiert.

2.2.4. Verfügbarkeit

Definition 2.2.4 (Verfügbarkeit)**.** "Wir sagen, dass das System die Verfügbarkeit (engl. availability) gewährleistet, wenn authentifizierte und autorisierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht unautorisiert beeinträchtigt werden können." [Eck14, S. 11]

Dies bedeutet insbesondere, dass es einem Subjekt zu jeder Zeit möglich sein muss, die Aktionen und wohl definierten Methoden auf Objekte auszuführen, für welche es autorisiert ist. Dabei muss gerade sichergestellt werden, dass das entsprechende System erreichbar ist, sowie dessen Funktionalität nicht verändert oder eingeschränkt werden kann. In IT-Systemen können unterschiedliche Prozesse von einer Großzahl von Nutzern um die verfügbaren Ressourcen konkurrieren, was zu einer Beeinträchtigung führen kann. Dabei ist es schwer, eine klare Trennung zwischen autorisierten und unautorisierten Aktionen zu definieren und deshalb auch, diese zu erkennen [Eck14, S. 12]. Angriffe gegen die Verfügbarkeit eines IT-Systems werden oft als *Denial-of-Service-Attack* bezeichnet. Dem gegenüber steht eine autorisierte Beeinträchtigung des Systems, durch eine Großzahl autorisierter Nutzeranfragen, ohne die Absicht, die Verfügbarkeit des Systems zu beeinträchtigen.

2.2.5. Verbindlichkeit

Definition 2.2.5 (Verbindlichkeit)**.** "Wir sagen, dass das System die Verbindlichkeit bzw. Zuordenbarkeit (engl. non repudiation) einer Menge von Aktionen gewährleistet, wenn es nicht möglich ist, dass ein Subjekt im Nachhinein die Durchführung einer solchen Aktion abstreiten kann." [Eck14, S. 12]

Dies ist besonders von Belang in den Bereichen des elektronischen Handels ("engl. Electronic Commerce" [Eck14, S. 12]), als auch der elektronischen Geschäfte ("engl. Electronic Business" [Eck14, S. 12]). Allgemein werden dabei Aktivitäten eine rechtliche Verbindlichkeit zugeschrieben, wie in dem Beispiel geschäftlicher Transaktionen, Käufe, Verträge und weiteres [Eck14, S. 12–13]. So muss zum Beispiel das Akzeptieren von AGBs von Apps oder Webseiten, als auch von Cookies im Browser der Verbindlichkeit unterliegen. Praktisch wird dies oft durch digitale Signaturen realisiert [Eck14, S. 13]. Dazu benötigt man die Erfüllung der Abrechenbarkeit, welche wiederum eine Überwachung und Protokollierung von Nutzeraktivitäten voraussetzt [Eck14, S. 13].

2.2.6. Anonymität und Privatsphäre

Folgende Definitionen sind wesentlich für diese Arbeit, welche sich weniger mit den konkreten Inhalten der Nachrichten von Subjekten befasst, als viel mehr mit den Metadaten über diese Nachrichten, die Möglichkeiten des Informationsgewinns über deren Verfasser (Subjekte) und damit deren Deanonymisierung.

Definition 2.2.6 (Anonymisierung)**.** "Unter der Anonymisierung versteht man das Verändern personenbezogener Daten der Art, 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." [Eck14, S. 13]

Definition 2.2.7 (De-Anonymisierung)**.** "Bei der De-Anonymisierung werden Daten, die in anonymer oder anonymisierter Form vorliegen, wieder konkreten Personen zugeordnet. Durch die Kombination von verfügbaren Daten wird eine individuelle Bezugnahme ermöglicht, so dass die Zuordnung zu dieser Person sehr wahrscheinlich oder sogar sicher ist." [Jä18]

Ziel der Anonymisierung ist es also, dass keine Deanonymisierung möglich ist, also konkret, keine "Bewegungs-, Kommunikations- oder Zugriffsprofile" [Eck14, S. 14] von Subjekten erstellt werden können. Dies ist ohne gewisse Vorkehrungen durchaus möglich, da beim Zugriff auf Webseiten im Internet oder dem Nutzen von Messengern meist mehr Daten versendet werden, als für die reine Funktionalität des Dienstes notwendig. Dazu gehören beispielsweise die IP-Adresse des Subjekts, URLs zuvor besuchter Webseiten, der genaue Zeitpunkt des Zugriffs oder des Sendens einer Nachricht [Eck14, S. 13], sowie weitere personenbezogene Daten. Allgemein kann man diese Daten oft unter dem Begriff der Metadaten (Abschnitt 3.1) zusammenfassen. Diese können nun entweder von böswilligen Dritten, oder den Anbietern der Dienste zum unautorisierten Erstellen von Profilen

der Nutzer verwendet werden [Eck14, S. 13]. Diese Profile sind äußerst lukrativ für die Betreiber der Seite, da sie unter anderem zur gezielten Schaltung von Werbung oder politischer Einflussnahme genutzt und somit an Werbetreibende oder politisch motivierte Institutionen verkauft werden können.

Mittels Anonymisierungsdiensten kann dies zumindest eingeschränkt werden. Diese verwenden grundlegend zwei Techniken [Eck14, S. 13]:

- Vermeidungstechnik (z.B. Daten unterdrücken)
- Verschleierungstechniken (z.B. Ersetzen durch Standardmuster, Verschlüsseln)

Als Unterform der Anonymisierung bietet die Pseudomisierung weniger Schutz der persönlichen Daten.

Definition 2.2.8 (Pseudomisierung)**.** "Dabei handelt es sich um das Verändern personenbezogener Daten durch eine Zuordnungsvorschrift (z.B. die Verwendung von Pseudonymen) derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse ohne Kenntnis oder Nutzung der Zuordnungsvorschrift nicht mehr einer natürlichen Person zugeordnet werden können." [Eck14, S. 13]

Wesentlich sind hierbei somit das Verwenden von Pseudonymen. Dies schränkt die Kenntnis der Identität des Nutzers auf diejenigen ein, welche Wissen über die Zuordnungsvorschrift haben und somit als vertrauenswürdig angesehen werden. Allgemein kann man die hier angesprochenen Probleme unter dem Begriff der Privatsphäre und deren Bedrohung zusammenfassen.

Definition 2.2.9 (Privatsphäre)**.** "Die oben angesprochenen Möglichkeiten, anhand von Verkehrs- und Aufenthaltsdaten Profile zu erstellen, führen zu einer Bedrohung der Privatsphäre (engl. privacy) des Nutzers." [Eck14, S. 14]

Dabei ist anzumerken, dass es viele verschiedene Definitionen für den Begriff der *Privatsphäre* gibt und diese schwer unter einer Definition zusammenzufassen sind. Die Bedeutung und der Schutz der Privatsphäre ist in den letzten Jahren enorm gestiegen.

Ein wesentliches Problem ist, dass Nutzer oft nicht selbst bestimmen können, in welchem Umfang und wofür ihre persönlichen Daten gewonnen, gespeichert und letztlich verarbeitet und genutzt werden. Das Recht darauf bezeichnet man als informationelle Selbstbestimmung.

Definition 2.2.10 (Informationelle Selbstbestimmung)**.** Die informationelle Selbstbestimmung […] gewährleistet die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen." [Eck14, S. 14]

Dies wurde bereits 1983 vom Bundesverfassungsgericht beschlossen, womit auch der Anspruch von Nutzern auf Anonymisierung anerkannt wurde [Eck14, S. 14]. Heute gilt das Recht auf Privatsphäre ("Recht auf Achtung des Privat- und Familienlebens" [Eum]) als Menschenrecht und ist Bestandteil aller Demokratien.

2.3. Netzwerkprotokolle

Da diese Arbeit vorrangig die Paketübertragung analysiert, spielen die dazugehörigen Protokolle eine zentrale Rolle. Das *ISO/OSI-Referenzmodell* (Abschnitt 2.3.1) und die damit verbundenen Protokolle finden zwar in der Praxis kaum Anwendung, das Modell selbst ist aber recht allgemein und heute noch gültig. Dem entgegen steht das *TCP/IP-Referenzmodell* (Abschnitt 2.3.2), wobei das Modell selbst kaum praktischen Nutzen hat, dessen Protokolle jedoch weit verbreitet sind [Tan03, S. 37].

2.3.1. ISO/OSI-Referenzmodell

Das *ISO/Open Systems Interconnection (OSI)-Referenzmodell* teilt die Netzwerkkommunikation in 7 Schichten auf und ordnet jeder Schicht eine Funktion zu.
Die Kommunikation zwischen Systemen (hier Rechner) entspricht somit diesem Modell. Es abstrahiert gezielt von den konkret verwendeten Protokollen und der verwendeten Netzwerk- und Hardwarearchitektur. Es basiert auf einem Vorschlag aus dem Jahr 1983, entwickelt von der International Standards Organization (ISO) zur Standardisierung der verschiedenen Protokolle in den einzelnen Netzwerkübertragungsschichten [Tan03, S. 37].

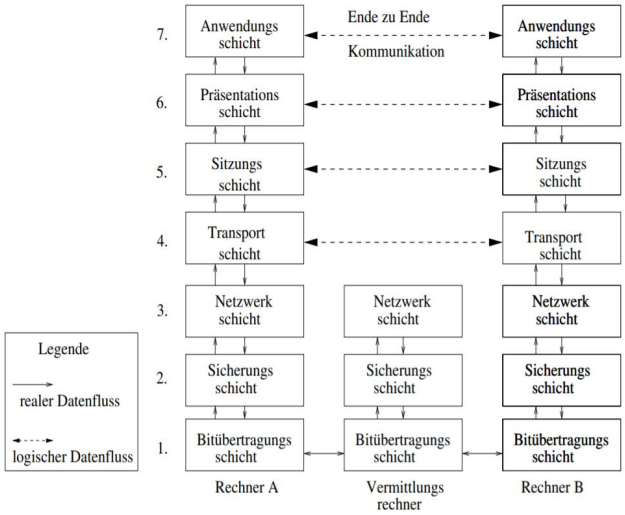


Abb. 1: Die Schichten des ISO/OSI-Modells [Eck14, S. 94]

Folgend werden kurz die Aufgaben der einzelnen Schichten erläutert [Tan03, S. 38–39],[Eck14, S. 94–96]:

1. **Bitübertragung**: physische Verbindung der Kommunikationspartner und Übertragung der Bitströme.
2. **Sicherung**: Bündelung von Datenpaketen, Prüfsummen zum Erkennen von Übertragungsfehlern und Kontrolle des Nachrichtenflusses mit *Sliding Window-Verfahren*.
3. **Netzwerk**: Kontrolle von Teilnetzen und Routing
4. **Transport**: paketorientierte Kommunikation zwischen Prozessen, Aufbau logischer Verbindungen und Flusssteuerung
5. **Sitzung**: Koordination und Synchronisierung der Kommunikation zwischen Anwendungsprozessen, sowie Sicherungspunkte für Unterbrechungen
6. **Präsentation**: Syntax und Semantik von Daten, Dateninterpretation, -kompression und -verschlüsselung
7. **Anwendung**: diverse Protokolle für Nutzer (HTTP, SMTP, FTP,...)

2.3.2. TCP/IP-Referenzmodell

Das Internet, wie wir es heute kennen, entstammt dem damaligen *ARPANET*. Dies war ein Forschungsnetzwerk, gesponsert vom U.S. Department of Defense (DoD), welches Universitäten und staatliche Einrichtungen miteinander verbunden hat. Anfangs baute das Netzwerk noch auf Telefonleitungen auf. Als Radio- und Satellitennetzwerke dazu kamen, stieß das Protokoll an seine Grenzen und eine neue Referenzarchitektur wurde benötigt: das *TCP/IP-Referenzmodell* entstand (1974) [Tan03, S. 39].

Die Internet-Protokollfamilie abstrahiert ebenfalls von der verwendeten Hardwarearchitektur der kommunizierenden Geräte. Diese umfassen PCs, Mobiltelefone, Router, Gateways [Eck14, S. 100] und in den letzten Jahren auch vermehrt Systeme des Internet of Things (IoT), sowie weitere Geräte. Die Kommunikation erfolgt paketorientiert und findet ihren Weg durch das Internet mittels Routingverfahren über Router und Switches. Dem zugrunde liegen das Internet Protokoll (IP) und die Transportprotokolle Transmission Control Protocol (TCP) und User Datagram Protocol (UDP). Diese entsprechen der Netzwerk- und Transportschicht des ISO/OSI-Modells, zu sehen in Abbildung 2. Dabei wird von Eckert hervorgehoben, dass "die Anwendungsdienste im TCP/IP-Modell direkt auf der Transportebene, also dem TCP- bzw. UDP-Protokoll aufsetzen. Das heißt, dass Sicherheitsprobleme der TCP/IP-Protokolle eine unmittelbare Auswirkung auf diese Anwendungsprotokolle haben, da keine ausgleichende Zwischenschicht, die zum Beispiel Verschlüsselungsaufgaben durchführt, existiert." [Eck14, S. 100]. Die Internetprotokolle wurden zu ihrer Zeit nicht entwickelt, um sicher zu sein [GIL19, S. 89]. Dabei war noch nicht abzusehen, welche gigantischen Ausmaße das Internet später annehmen und dass es böswillige Angreifer geben wird, die jene Schwachstellen ausnutzen. Privatsphäre (Abschnitt 2.2.6) und Datenintegrität (Abschnitt 2.2.2) werden heute beispielsweise mittels des Transport Layer Security (TLS)-Protokolls auf der Anwendungsebene gewahrt. Für diese Arbeit sind die Protokolle IP, TCP und TLS von Bedeutung, welche folgend erläutert werden.

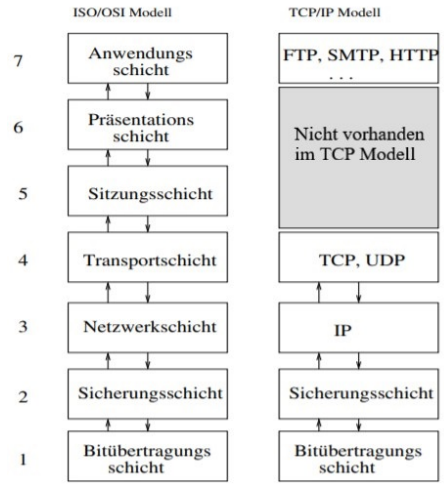


Abb. 2: OSI- versus TCP/IP-Modell (vereinfachte Darstellung) [Eck14, S. 101]

Internet Protokoll (IP)

Das Internet Protokoll (IP) ist vollständig in [RFC791] beschrieben. Es realisiert die Netzwerkschicht, vermittelt Pakete und stellt dabei jedoch nicht sicher, ob Pakete tatsächlich erhalten werden. Daher wird es auch als "unzuverlässige[r] Dienst"[Eck14, S. 102] bezeichnet. Die einzelnen Pakete müssen deshalb die Adressen des Sender und Empfänger kennen – die *IP-Adressen*. Jedes IP-Paket hat einen *Header*, welcher alle notwendigen (Meta)daten für das Protokoll und das Übermitteln der Nutzdaten enthält.

Transmission Control Protocol (TCP)

Das Transmission Control Protocol (TCP) hingegen ist verbindungsorientiert und zuverlässig. Es setzt auf dem IP-Protokoll auf und befindet sich in der Transportebene [Eck14, S. 107]. Ziel ist es, eine logische Verbindung zwischen zwei Systemen aufzubauen. Dabei wird geprüft, ob alle Pakete angekommen sind, sowie die Reihenfolge dieser mittels Sequenznummern. Der Empfänger bestätigt den Erhalt von Paketen und bei Verlust werden diese erneut versandt. Zudem wird ein Kanal nicht nur zwischen Systemen, sondern auch Prozessen auf diesen Systemen aufgebaut. Daher können mehrere Dienste auf einem System gezielt mit mehreren auf einem anderen System kommunizieren. Diese Zuordnung geschieht über *Port-Adressen*. Die vollständige Spezifikation von TCP ist in [RFC793] beschrieben.

2.3.3. Transport Layer Security (TLS)

Das ursprünglich unter dem Namen Secure Socket Layer (SSL) [RFC6101] entwickelte und heute industrieweit als Transport Layer Security (TLS) bekannte Protokoll, sichert Dienste und Protokolle der Anwendungsebene – üblicherweise Hypertext Transfer Protocol (HTTP) – von der darunter liegenden Transportebene ab. Damit befindet es sich im ISO/OSI-Modell in Schicht 5 (Sitzungsschicht). Das Protokoll gilt heute als Industriestandard zum Aufbau sicherer HTTP-Verbindungen und wird von allen gängigen Browsern unterstützt [Eck14, S. 809]. Die dadurch entstehende Hypertext Transfer Protocol Secure (HTTPS)-Verbindung, auch bekannt als "HTTP over TLS oder "HTTP over SSL" [RFC2818], verschlüsselt die Anwendungsdaten der HTTP-Verbindung. TLS wird dabei nicht nur zum Aufbau eines sicheren Kanals für HTTP eingesetzt, wie auch in dieser Arbeit zu sehen ist.

Zentral für die Funktionalität ist das TLS (Version 2 [RFC5246]) Record Protocol. Es fragmentiert die zu übermittelnden Daten in Blöcke, komprimiert diese optional, wendet einen Message-Authentication-Code (MAC) an, verschlüsselt und versendet die Daten. Es sind vier Teilprotokolle beschrieben, die das Record Protocol nutzen. Sie werden nach dem Wert des ersten Bytes unterschieden:

- **ChangeCipherSpec (Type = 20):** Bestätigung der verwendeten Cypher Suite
- **Alert (Type = 21):** Warnungen, Fehler und Beenden der Sitzung
- **Handshake (Type = 22):** Übereinkunft der verwendeten kryptografischen Verfahren (Cypher Suite), Authentifizierung und Aufbau eines sicheren Kanals
- **Application (Type = 23):** meist verschlüsselte Anwendungsdaten

Zudem geschieht aktuell ein Umstieg von der 2008 erschienenen TLS Version 1.2 [RFC5246] zur 2018 veröffentlichten Version 1.3 [RFC8446]. Dabei wird die Unterstützung für veraltete, unsichere kryptografische Verfahren wie MD5 und SHA-224, sowie unsicheren Features wie die Datenkomprimierung eingestellt. Zudem kann der Handshake nun in 1-Round Trip Time (RTT) und nicht wie in Version 1.2 in 2-RTT abgeschlossen werden. Die Verwendung von *Ephermal Keys* sichert perfekte Vorwärtssicherheit (Abschnitt 4.3.1) und alle Handshake-Nachrichten nach dem "ServerHello" sind verschlüsselt.

2.4. Angriffe/ Unbefugter Informationsgewinn

Diese Arbeit betrachtet zwei Szenarien, in welchen Informationen von Nutzern missbraucht werden. Erläutert werden diese anhand der bei der Paketübertragung anfallenden Metadaten. Deren Menge hängt vom jeweiligen Szenario und dessen Skalierung ab. Sämtliche aufgezeichneten Daten wurden mit dem zuvor erläuterten TLS-Protokoll verschlüsselt. Diese und weitere kryptografischen Methoden werden nicht umgangen und es werden auch keinerlei Sicherheitslücken ausgenutzt. Kernpunkt dieser Arbeit ist, dass Informationen über Nutzer gewonnen werden können, mittels welchen man diese deanonymisieren kann, ohne Sicherheitsmechanismen zu umgehen und nur unter Verwendung der anfallenden Metadaten. Damit soll insbesondere verdeutlicht werden, dass die Verschlüsselung der Daten allein nicht genügt, um die Anonymität der Nutzer zu wahren.

2.4.1. Man-in-the-Middle-Angriff

Im ersten Szenario – dem Man-in-the-Middle (MitM)-Angriff – hält die Verschlüsselung (TLS) den Angreifer davon ab, die Applikationsdaten zu lesen. Der Angreifer zeichnet sämtlichen verschlüsselten Datenverkehr zwischen Nutzer und Server auf und wertet ihn aus. Dieses Szenario wird praktisch an verschiedenen Anwendungen getestet. Das Vorgehen wird dabei im Abschnitt 5 beschrieben und für die einzelnen Applikationen in den jeweiligen Kapiteln konkret erläutert, sowie die dabei aufgezeichneten Daten ausgewertet. Dieser Angriff wird umso mächtiger, je mehr Netze überwacht werden. Das reelle Äquivalent dazu wäre ein global agierender Nachrichtendienst oder Staat [GIL19, S. 88], [Bah+20]. Als Angriffspunkt nutzt der Angreifer das externe Netz, welches den Nutzer mit dem Server verbindet. Er hat somit keinen Zugang zu den serverinternen Daten und dessen Netzwerk. Zudem bleibt der MitM-Angreifer passiv; er liest also nur den Datenstrom mit und verändert diesen nicht. Praktisch kann dieser Angriff mittels verschiedenen Methoden durchgeführt werden: ARP-, DNS-, oder DHCP- Manipulation, Vortäuschen eines WLAN-Hotspots und weitere.

2.4.2. Deanonymisierung & Privatsphäreverletzung durch Serverbetreiber

Das zweite Szenario betrachtet, wie viele Daten die Server lesen können. Die sichere TLS-Verbindung besteht lediglich zwischen Nutzer und Server, weshalb der Server selbst die nach außen verschlüsselten Applikationsdaten lesen kann. Dies können Daten sein, welche für die Funktionalität der Anwendung notwendig sind. Gleichzeitig können diese Daten auch eine Vielzahl an Informationen über den Nutzer enthalten. Die vom Nutzer übertragenen Nachrichten sind üblicherweise E2E-verschlüsselt und damit auch nicht für den Server einsehbar. Bei den restlichen Daten muss man dem Server hingegen vertrauen, dass dieser die Daten nicht missbraucht. Zudem können jene Server selbst von Schadsoftware befallen sein, wodurch auch Nutzerdaten von nicht-böswilligen Serverbetreibern komprimiert werden können. Zuletzt gibt es auch den Fall, dass Sicherheitsbehörden oder Regierungen Zugang zu den Serverdaten einfordern [GIL19, S. 88], um Informationen über ihre Bevölkerung und Oppositionelle zu erhalten, zuletzt geschehen in Russland und den Hongkong-Protesten [Bah+20].

3. Metadaten

Der folgende Abschnitt behandelt die Definition von Metadaten. Diese stammt von Pujol u. a., „*Spying on Instant Messaging Servers: Potential Privacy Leaks through Metadata*“, 2019 [Puj+19], welche die allgemeine Definition von Metadaten für den Bereich der *Datensicherheit* konkretisiert.

Pujol [Puj+19] greift die Arbeit von Georges aus 2009 [Geo09] auf, nach welcher die digitale Identität einer Person aus drei Schichten aufbaut ist:

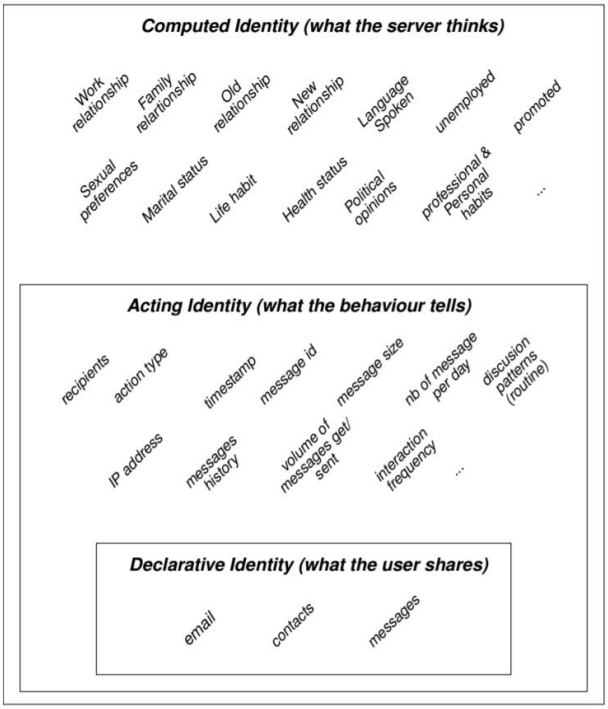


Abb. 3: Repräsentation digitale Identität [Puj+19]

Figur 3 verdeutlicht diesen Aufbau und beinhaltet für jede Schicht einige Beispiele. Es werden folgende Schichten beschrieben [Puj+19]:

- **Berechnete Identität:** vom Server anhand der Daten aus den deklarativen und handelnden Identitäten berechnet
- **Handelnde Identität:** zusammengesetzt aus den Aktivitäten und Ereignissen des Benutzers (Metadaten)
- **Deklarative Identität:** vom Benutzer (aktiv) eingegebene Daten

Beim Schutz der Privatsphäre von Nutzern werden oft nur dessen eingegebenen Daten betrachtet, nicht jedoch die zugehörigen Metadaten. Laut einer Studie von [Geo09] bestimmt hingegen die digitale Identität eines Benutzers auf Facebook weniger die Deklarative Identität, als mehr die Handelnde und Berechnete Identität [Puj+19].

Gerade die Ebene der Handelnden Identität ist stark von Metadaten geprägt. In diesem Bezug unterscheidet sich Facebook von klassischen Instant Messagern (wie WhatsApp, Telegram,...), da hier direkt die Metadaten der Nachrichten mit denen des Profils und der Aktivitäten des Benutzers verknüpft werden können, was folglich viel größeres Potential zur Deanonymisierung bietet.

3.1. Allgemeine Definition

Definition 3.1.1. Metadaten [Puj+19, S. 7] sind als die Daten definiert, welche Informationen über andere Daten bereitstellen.

Diese Definition deckt sich mit der von Dublin Core™ [Hil05], einem Metadatenstandard und eine Konvention zur Beschreibung von Dokumenten und anderen Daten. Eine konkretere Definition bietet Rühle, *„Kleines Handbuch Metadaten“*, 2014 [Rü14]:

> *”Metadaten sind Daten, die dazu dienen, Objekte, Konzepte aber auch Daten strukturiert zu beschreiben.”*

Für diese Arbeit beschreiben die anfallenden Daten bei der Paketübertragung Metadaten der Nachrichten im weitesten Sinne. Die Definition scheint jedoch zutreffender, wenn man die Kommunikation im Ganzen als Menge von Daten betrachtet und sämtliche Daten um die Protokolle und versandten Nachrichten als Metadaten zu diesen Daten. Besonders von Interesse werden dafür Daten sein, wie etwa die IP-Adressen der Kommunikationspartner, Zeitpunkte der eintreffenden Nachrichten und deren Größe, sowie Informationen über die verwendeten Protokolle. All diese Daten liefern zwar nur indirekt und teilweise überhaupt Informationen über die versendeten Nachrichten selbst, um Informationen über die Nutzer zu erfahren sind diese aber oft weitaus wichtiger, als jene versandten Nachrichten.

3.2. Das Problem mit den Metadaten & Deanonymisierung im Internet

Spätestens seit den Enthüllungen von Edward Snowden im Jahre 2013 [Gre13] wird deutlich, dass Nachrichtendienste ein großes Interesse an Metadaten haben, welche oft mehr Informationen über die Nutzer preisgeben, als die eigentlichen Daten selbst [EM13]. In der Betrachtung der Sicherheit von Metadaten wird oft von solch einem globalen Widersacher ausgegangen, welcher eine Vielzahl von Netzwerken überwachen kann. Die hierbei kritischen Informationen sind jedoch meist tief in den verwendeten Protokollen verankert und nicht einfach zu verschleiern. So ist es trivial, Kommunikationspartner zu verknüpfen, wenn man die IP-Adressen der Nachrichten mitliest. Aber auch ohne die IP-Adressen kann man die Zeiten der jeweils eingehenden und ausgehenden Nachrichten überprüfen und daher eine Verbindung beider Seiten zueinander abschätzen. Empfängt Alice stets Nachrichten, kurz nachdem Bob diese versandt hat und umgekehrt, so kann man davon ausgehen, dass beide miteinander kommunizieren, auch ohne die Zieladressen der Nachrichten zu kennen [GIL19, S. 87].

3.3. Verschleierung von Metadaten & das Problem mit TCP/IP

Eine verbreitete Methode zum Verschleiern von Metadaten ist TOR[1]. Es verbirgt die IP-Adressen des Nutzers, indem die Webanfrage über eine Menge von zufälligen Relais geleitet wird. TOR ist dezentral und bietet allgemein mehr Sicherheit als die Verwendung eines Virtual Private Network (VPN), ist dafür aber langsamer. VPNs können ebenfalls IP-Adresse verbergen und Nachrichten verschlüsseln. Die Sicherheit hängt aber vom jeweiligen Betreiber ab und man muss diesem vertrauen, dass dieser nur notwendige Daten speichert und ob er im Zweifel Daten an Dritte weitergibt. Beide Methoden können jedoch fehlschlagen, wenn der Angreifer einen hinreichend großen Teil des Netzes überwacht und dadurch Nachrichten miteinander korrelieren kann, so geschehen durch die National Security Agency (NSA), aufgedeckt in den Enthüllungen rund um Edward Snowden [GIL19, S. 87], [Gre13].

Es existieren diverse Methoden und Protokolle zum Verschleiern der Metadaten von Nachrichtenübertragung [LGZ18], [AS16], [Hoo+15], [Arn18], [JSH20]. Viele dieser Protokolle und Verfahren setzen auf den bestehenden Protokollen der Netzwerkübertragung auf und versuchen durch nachträgliche und zusätzliche Methoden minimale oder gar keine Metadaten nach außen hin preiszugeben. Dieses Vorgehen ist im Vergleich zur Nachrichtenübertragung ohne Metadatenverschleierung teuer: die Pakete sind größer und benötigen länger, um ihr Ziel zu erreichen. Zudem muss je nach Anwendung konstant ein zusätzlicher Datenstrom gesendet werden, um Zeitpunkte zu verbergen, wann tatsächlich Pakete übertragen werden. Hinzu kommt, dass Pakete durch Routingverfahren oft unterschiedliche Wege vom Sender zum Empfänger zurücklegen, als umgekehrt. Einem Angreifer genügt ein Punkt auf einem der beiden Wege, um Metadaten über die Kommunikation zu erlangen. Dadurch verdoppelt sich praktisch die Fläche der möglichen Angriffspunkte im Internet für bidirektionale Verfahren, insbesondere TCP. Sämtliche Maßnahmen versuchen das Problem zu beheben, dass "die zugrunde liegenden Protokolle, welche die Kommunikation über das Internet erleichtern, nicht entworfen wurden, um sicher zu sein." [GIL19, S. 89]. Selbes gilt für jene kryptografische Verfahren, welche zwar Informationen schützen und verschlüsseln, die dabei anfallenden Metadaten der Protokolle jedoch nicht. Nun kann in den kommenden Jahren vielleicht ein Durchbruch auf dem Gebiet geschehen und ein Protokoll entwickelt werden, welches auf dem TCP/IP-Protokoll aufsetzt und dessen Schwächen im Bereich der Preisgabe von Metadaten verbessert, ohne große zusätzliche Kosten und Latenz zu erzeugen. Eventuell sollte man jedoch jene Protokolle selbst in Bezug auf Sicherheit und Menge an Metadaten überdenken und neu entwickeln. Dass aktuell hauptsächlich versucht wird, Metadatenreduktion auf den bestehenden Protokollen aufzusetzen, ohne jene Protokolle von Grund auf neu zu entwerfen, liegt wohl an den enormen Umstiegskosten, welche dies zur Folge hätte. Ein Großteil unserer Kommunikation beruht auf dem TCP/IP-Modell und damit einem Modell, welches ohne weitere Vorkehrungen Metadaten seiner Nutzer preisgibt. Dieses Modell umfassend zu hinterfragen und Alternativen aufzuzeigen übersteigt den Rahmen dieser Arbeit, aber sie soll zumindest zum Nachdenken darüber anregen.

[1] Offizielle Webseite: `https://www.torproject.org/` (abgerufen am 4. September 2022)

4. Sicherheit von Instant Messengern

Wie zuvor erwähnt werden in dieser Arbeit keinerlei Sicherheitslücken ausgenutzt oder kryptografische Verfahren und Implementierungen umgangen. Die Messenger und deren Verschlüsselung werden als sicher angesehen. Dennoch muss diese Aussage belegt werden, was in zahlreichen Arbeiten geschehen ist. Folgend sollen die wichtigsten Aspekte und der aktuelle Stand sicherer Kommunikation per Messenger zusammengetragen werden.

4.1. Relevanz von Messengern

Kommunikation ohne Messenger ist heutzutage kaum noch vorstellbar. Gleichauf mit der Verbreitung von Handys stieg die Nutzung von Messengern in den letzten Jahren rasant an. Im Jahr 2018 geschah es zum ersten Mal, dass ein Rückgang vom Sprachverkehr zu verzeichnen war, während der Internet-Netzwerkverkehr rasant anstieg [SZ20, S. 408]. Mittlerweile verzeichnen die größten Messenger Apps weltweit mehr als 4 Milliarden Nutzer. Unter den größten 3 befinden sich WhatsApp mit 2 Milliarden, Facebook Messenger mit 1,3 und WeChat mit 1,2 Milliarden aktiven Nutzern [Meh20a, S. 1]. Die Verteilung der Nutzerzahlen in Deutschland ist in Abbildung 4 zu sehen.

Gerade in Deutschland ist WhatsApp mit 58 Millionen aktiven Nutzern (Stand November 2019) [Meh20a, S. 6] der mit Abstand am meisten genutzte Messenger. Außerhalb von persönlichen Gesprächen findet ein Großteil unserer Kommunikation mittels ebendieser Messenger-Dienste statt. Doch wie sicher ist diese Kommunikation?

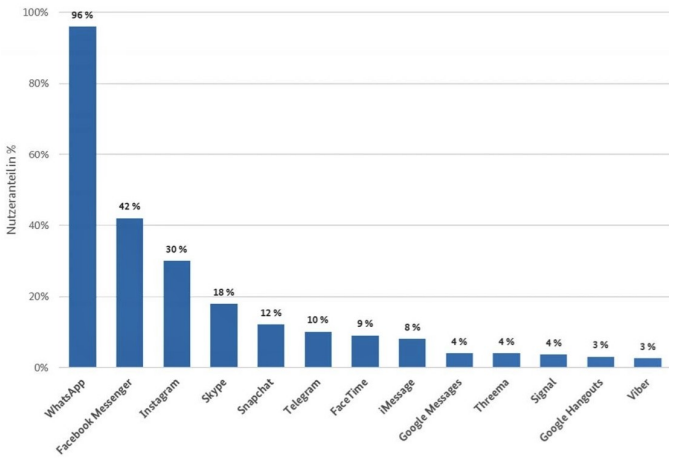


Abb. 4: Nutzungsanteile Messenger Deutschland: 2020 [Meh20b]

Natürlich gilt es bei solch einer enorm relevanten Technologie, besonders Wert darauf zu legen, dass die derzeit mehr als 4 Milliarden Nutzer geschützt sind. Die Frage der Sicherheit ist angesichts der enormen Verbreitung aber nicht nur relevant für Privatpersonen, sondern auch für Unternehmen. Diese verwenden für ihre innerbetriebliche Kommunikation aus Effizienzgründen, sowie aufgrund der geringen Umstiegskosten, da viele Privatpersonen diese Messenger bereits nutzen, immer mehr Messenger-Dienste, anstelle von z.B. E-Mails. So können unter anderem kleinere Arbeitsgruppen oder Produktionsbereiche in WhatsApp-Gruppen organisiert werden, um direkter und schneller miteinander kommunizieren zu können. Natürlich entfallen die eher klassischen Kommunikationskanäle wie Mails oder Telefonie nicht komplett. Oft dienen Messenger eher einer weiteren, äußerst effizienten Möglichkeit der Kommunikation.

4.2. Sicherheitsanforderungen an Messenger-Dienste

Von den zuvor beschriebenen Schutzzielen (2.2) sind für das Bereitstellen einer sicheren Datenverbindung die beiden Aspekte Informationsvertraulichkeit (2.2.3) und Authentizität (2.2.1) wesentlich. Als Hilfsmechanismen werden Protokolle zur Einrichtung authentifizierter Schlüssel, als auch Protokolle für Vorwärts- (engl. forward-) und Post-Kompromittierungs- (engl. post-compromise security) Sicherheitseigenschaften genutzt [SZ20, S. 408].
Es folgt eine Aufzählung der wichtigsten sicherheitsrelevanten Anforderungen an moderne Messenger-Dienste nach [SZ20, S. 408]:

- Ende-zu-Ende-Verschlüsselung von Kommunikation zwischen Nutzern, Nachrichten dürfen nicht unverschlüsselt (im *plaintext*) auf dem Server erreichbar sein
- Methoden zum Widerstand im Falle einer Kompromittierung des Nutzergerätes mit dessen Schlüsseln, meist in Form von *Perfect Forward-* (4.3.1) und *Post-compromise-Secrecy* (4.3.2)
- sichere Kommunikation in Gruppenchats, insbesondere mit dynamischer Teilnehmermenge, um gegen Client- und Server-Key-Kompromittierung geschützt zu sein
- Sicherheit der Langzeitspeicherung des Benutzerinteraktionsverlaufs mit anderen Nutzern, beinhaltend die physische Änderung des Client-Gerätes, Änderungen der Benutzerattribute und das Benutzen externer Cloud-Dienste zur Speicherung

Hinzu kommt die Notwendigkeit einer Authentifizierung der Nutzer und damit derer Nachrichten. Nutzer müssen sich gegenseitig authentifizieren können und in Fällen von Kommunikation über zentrale Server müssen diese sich ebenfalls gegenüber dem Nutzer authentifizieren.

4.3. Sicherheit von Messengern

Zunächst ist zu sagen, dass es sich bei dieser Kommunikation um eine asynchrone Kommunikation handelt: Sender und Empfänger müssen nicht zur selben Zeit online sein, um miteinander zu kommunizieren. Ein besonderes Merkmal dieser Art der Kommunikation ist die obligatorische Fähigkeit eine sichere Verbindung zwischen den Kommunikationspartnern aufzubauen zu können, während diese nicht gleichzeitig online sind [SZ20, S. 408].

Die meisten weit verbreiteten Messenger (WhatsApp, Facebook Messenger, Signal) leisten größtenteils gute Arbeit darin, ihre Nutzer (Subjekt 2.1.1) und deren Kommunikation (passives Objekt 2.1.1) zu sichern. Wie sich jedoch im Verlauf der Arbeit zeigt, erfüllt keiner dieser Messenger die vollen Sicherheitvoraussetzungen, da ihnen trotz sicherer Verschlüsselung konzeptionelle Probleme zugrunde liegen. Dies kann damit erklärt werden, dass die Arbeit an diesen und damit ihre Konzeptionierung bereits begann, bevor der Großteil der aktuellen wissenschaftlichen Erkenntnisse vorlag [SZ20, S. 408].

4.3.1. Perfekte vorwärts gerichtete Geheimhaltung (engl. Perfect Forward Secrecy)

Perfect Forward Secrecy (PFS) beschreibt eine kryptografische Eigenschaft von Verfahren und Protokollen zum Schlüsselaustausch, bei welchen die Kenntnis eines Schlüssels nicht dafür sorgt, dass vorherige und folgende Schlüssel bekannt sind. Insbesondere hängen neue Schlüssel nicht von alten ab [Eck14, S. 436]. Somit kann ein Angreifer, welcher den aktuellen Schlüssel in Erfahrung gebracht hat, nur die Nachrichten der aktuellen Sitzung, aber nicht die der vorherigen entschlüsseln. Zusätzlich wird gefordert, dass auch die Kenntnis des Langzeitschlüssels nicht zur Kompromittierung der vorherigen Nachrichten führt. Alle weiteren Nachrichten können entschlüsselt werden, bis der Angreifer aufgedeckt und der Langzeitschlüssel erneuert wurde. Dies senkt die Motivation eines Angreifers einen Langzeitschlüssel in Erfahrung zu bringen. TLS (Abschnitt 2.3.3) mit dem Diffie-Hellman-Schlüsselaustauschverfahren setzt *Perfect Forward Secrecy* um [Eck14, S. 820], [RFC5246], ebenso wie das von WhatsApp verwendete Signal-Protokoll mittels des *Double Ratchet Algorithm*. Vom Bundesamt für Sicherheit in der Informationstechnik (BSI) wird PFS in Verbindung mit TLS Version 1.2 und 1.3 empfohlen [BSI19].

4.3.2. Post-Kompromittierungssicherheit (engl. Post-Compromise Security)

Hierbei soll Sicherheit gewährt werden, selbst nach der Kompromittierung des Schlüssels.

Definition 4.3.1 (Post-Compromise Security)**.** Ein Protokoll zwischen Alice und Bob bietet *Post-Compromise Security (PCS)*, wenn Alice eine Sicherheitsgarantie für die Kommunikation mit Bob hat, selbst wenn Bobs Geheimnisse bereits kompromittiert wurden [CCG16, S. 1].

Somit kann ein Angreifer, welcher den Schlüssel einer Nachricht in Erfahrung gebracht hat, zwar diese Nachricht entschlüsseln, aber nicht folgende Nachrichten. In Kombination mit PFS (Abschnitt 4.3.1) sinkt somit die Motivation von Angreifern sehr, Schlüssel von Nachrichten in Erfahrung zu bringen, da nur die zu diesem Schlüssel zugehörige, aber nicht vorherige oder folgende Nachrichten entschlüsselt werden können.

4.3.3. Authentifizierung

Die Authentifizierung von Nutzern erfolgt bei den untersuchten Messenger-Diensten in der Regel über deren Telefonnummer. Dabei wird ein einmaliger Code an die zuvor angegebene Nummer oft per SMS gesendet, welcher nach kurzer Zeit verfällt und dem Nutzer abgefragt wird. Einige Applikationen erlauben auch die Abfrage eines PINs, sowie vor allem in Banking-Apps biometrischer Merkmale wie zum Beispiel der Fingerabdruck bei jedem Login. Des Weiteren können oft mehrere Geräte für den selben Account genutzt werden. WhatsApp fragt hierbei einen QR-Code per Scan ab. Matrix erlaubt das speichern eines Schlüssels, zur Sicherung des Chats. Somit soll sichergestellt werden, dass nur authentifizierte Nutzer Zugang zu dem Account haben.

4.3.4. Schlüssel: X3DH & Double Ratchet Algorithmus

Das am weitesten verbreitete Sicherheitsmodell für die zuvor beschriebenen Sicherheitsanforderungen von Messengern ist der *Double Ratchet Algorithm* in Verbindung mit einem Protokoll zur Einrichtung eines authentifizierten Schlüssels zum Zeitpunkt des Aufbaus einer sicheren E2E-Kommunikation [SZ20, S. 409]. Das Schlüsselvereinbarungsprotokoll Extended Triple Diffie-Hellman (X3DH) wird von vielen Messenger genutzt, wie z.B. WhatsApp, Facebook Messenger, als Variante in Signal und weiteren [MM16b]. Es wird ein gemeinsamer geheimer Schlüssel für beide Parteien etabliert, welche sich anhand ihrer öffentlichen Schlüssel gegenseitig authentifiziert haben. X3DH bietet *Perfect Forward Secrecy* (4.3.1) und plausibel leugnbare Verschlüsselung (engl. deniable encryption). Der generierte geheime Schlüssel wird dann von beiden Kommunikationspartnern synchron als Ausgangs-*Seed* für die *Double Ratchet Verschlüsselung* verwendet.

Der *Double Ratchet Algorithm* ist ein Schlüsselmanagementalgorithmus der beiden Autoren von X3DH [MM16a]. Für jede neue Nachricht erhalten die Nutzer neue Schlüssel, sodass ältere und folgende Schlüssel nicht berechnet werden können. Dadurch wird die *Perfect Forward Secrecy* (4.3.1) und *Post-Compromise Security* (4.3.2) umgesetzt. Bei einer Kompromittierung eines Schlüssels sollen möglichst wenig Nachrichten für einen Angreifer entschlüsselbar sein. Der Name entstammt der Kombination des symmetrischen Schlüssels der Key Derivation Function (KDF) und der ”ratchet” (dt. Sperrklinke) basierend auf Diffie-Hellman (DH) [MM16a, S. 13]. Die KDF erzeugt stetig neue Schlüssel, sodass vorherige Schlüssel und die damit verschlüsselten Nachrichten nicht gelesen werden können. Dies allein führt aber zu dem Problem, dass ab dem Zeitpunkt der Kompromittierung des Schlüssels die folgenden berechnet werden können. Zur Sicherung der zukünftigen Schlüssel werden DH-Parameter versendet um einen neuen Schlüssel zu berechnen. Dies findet unter anderem Anwendung in Signal, Facebook Messenger, Element (Matrix) und WhatsApp.

4.3.5. Sicherheit von Gruppenchats

Ein weiteres für Messenger-Dienste relevantes Problem im Bezug auf die Sicherheit der Nachrichten ist die Verschlüsselung von Gruppenchats mit mehreren teils dynamisch beitretenden Teilnehmern. Die zuvor betrachteten kryptografischen Methoden thematisieren eine End-to-End (E2E)-Verschlüsselung der Kommunikation zwischen genau zwei Teilnehmern (meist kommuniziert ”Alice” mit ”Bob”). Die hier funktionierenden kryptografischen Methoden können nicht ohne weiteres auf Gruppenchats angewandt werden.

Eine naive Herangehensweise wäre, dass jeder Teilnehmer mit jedem anderen Teilnehmer eine sichere Ende-zu-Ende-Verschlüsselung aufbaut und Schlüssel etabliert. Die Menge der benötigten Schlüssel und versandten Nachrichten würde jedoch exponentiell zur Teilnehmeranzahl steigen, wodurch dieses Vorgehen in der Praxis für große Gruppen ungeeignet ist. Hinzu kommt, dass Diffie-Hellman (DH) – die Grundlage der meisten Schlüsselübereinkünfte – für genau zwei Personen designt wurde und nicht ohne weiteres auf mehr als zwei Teilnehmer angewandt werden kann. Man könnte eine Gruppen-DH-Schlüsselübereinkunft per binärer Baumstruktur lösen, indem stets paarweise Schlüssel

mit je zwei Teilnehmern generiert werden. Dies ist jedoch auch schwierig in asynchroner Kommunikation umzusetzen, in der nicht jeder Teilnehmer gleichzeitig online sein muss. Somit könnte ein neues Gruppenmitglied erst hinzugefügt werden, nachdem jedes andere Mitglied online war und seinen Teil zur Schlüsselgenerierung beigetragen hat. Die Authentifizierung neuer Teilnehmer oder Änderung von Eigenschaften bestehender Teilnehmer (z.B. Sicherheitsnummer) ist ebenfalls ein offenes Problem. Zuletzt hätte diese Anwendung nicht die Eigenschaft der PFS (4.3.1) [RMS18, S. 1]. Dafür müssten stetig neue Schlüssel für alle Teilnehmer generiert werden.

In der Praxis gibt es kein einheitliches Vorgehen zur Verschlüsselung von Gruppenchats [RMS18]. Eine Möglichkeit ist, zu jeder Nachricht einen Sender-Schlüssel hinzuzufügen und einen neuen Schlüssel zu generieren, analog zu KDF (4.3.4). Da dieser Schlüssel nicht geändert werden kann, geht jedoch die Post-Compromise Security (4.3.2) verloren, da ab dem Zeitpunkt der Kompromittierung der Angreifer ebenfalls die neuen Schlüssel generieren kann. Das Problem der sicheren Kommunikation in Gruppen ist wie die Sicherheit von Instant-Messengern selbst ein in der Wissenschaft vergleichsweise neues und auch wenn es in den letzten Jahren Fortschritte auf dem Gebiet gab [CG+18], so ist dies aktuell in der Praxis ein noch offenes Problem, welches gelöst werden muss.

4.4. Sicherheit aktueller Messenger

Zu jedem dieser Messenger wurde umfangreich der Aspekt der Sicherheit in zahlreichen Artikeln betrachtet. Dazu steigt die Liste der Messenger stetig, weshalb an dieser Stelle nur ein Überblick über die wichtigsten Messenger und deren sicherheitsrelevante Aspekte erfolgen soll.

4.4.1. WhatsApp & Matrix (Element)

Beide Messenger sind wesentlicher Bestandteil des praktischen Teils dieser Arbeit und werden daher gesondert in Abschnitt 6.2 und 7.3 betrachtet.

4.4.2. Telegram

Die Ende-zu-Ende-Verschlüsselung wird in den sogenannten ”Secret Chats” unterstützt, aber nur von dem Gerät aus, welches die sichere Verbindung aufgebaut hat. Somit ist dies nicht auf anderen Geräten des Benutzers möglich, wodurch die Sicherheit deutlich reduziert wird [SZ20, S. 408]. Gruppenchats sind in Telegram nicht zusätzlich geschützt.

Zudem benutzt Telegram ein eigens entwickeltes Protokoll und verstößt damit gegen ein allgemein bekanntes Prinzip der Kryptographie, nachdem man ein kryptografisches Verfahren nicht selbst entwickeln soll, wenn es für den selben Zweck bereits Methoden gibt, welche als sicher angesehen werden [Reh12]. Das von Telegram entwickelte Protokoll heißt *MTProto 2.0* [Tur16] und wird von Telegram selbst zwar als sicher dargestellt, in vielen anderen Artikeln aber stark kritisiert [JO15], [SZ20], [EPP20], [Tur16].

4.4.3. Signal

Der von Edward Snowden genutzte Messenger[2] Signal wurde im Juli 2014 von der gemeinnützigen *Signal Foundation* veröffentlicht [Mar14] und basiert auf dem gleichnamigen, öffentlich einsehbaren Signal-Protokoll. Dessen Hauptbestandteile X3DH und den *Double Ratchet Algorithmus* wurden vorher im Abschnitt 4.3.4 besprochen. Signal selbst ist open-source[3] und wird allein deswegen bereits als sicherer angesehen als WhatsApp und dessen geheimer Quellcode, welcher dadurch stets die Möglichkeit offen lässt, dass die Entwickler eine Hintertür eingebaut haben [ORP20]. Ein neuer Account wird mit der Telefonnummer registriert und per SMS-Code authentifiziert. Mit dem Account verbunden ist ein selbst gewählter Name und ein Profilbild. Das Kontaktbuch der Nutzer mit allen Telefonnummern liegt gehasht auf dem Server und wird über Oblivious RAM (ORAM) ausgewertet, wodurch selbst Serverbetreiber die Klartexte der Telefonnummern nicht lesen können [Sch17]. Zudem ist Signal metadatenfreundlich konzipiert und speichert möglichst wenig dieser Metadaten über dessen Nutzer [Dof21]. Unter anderem deshalb wird Signal von zahlreichen Datenschützern und Fachkundigen im Bereich der IT-Sicherheit empfohlen [ORP20], [Sta], [Rud16].

4.4.4. WeChat

Das Thema "WeChat" und dessen vollständige, auch politische Einordnung, soll in diesem Abschnitt nur kurz betrachtet werden. Die primär in China dominante Messenger-App gehört mit 1,2 Milliarden aktiven Nutzern zu den größten weltweit. Die internationale App-Store Variante verzeichnet zumindest 100 Millionen Downloads [Was]. Es bietet einen enormen Funktionsumfang: von Messaging, Blogs und Timelines [Was], bis hin zur Bezahlfunktion und Verknüpfung der ID-Karte für digitale Behördengänge [Heg17]. Damit ist die App für viele Chinesen der zentrale Aspekt ihrer Internetaktivität. Das Prekäre daran ist, dass WeChat als Überwachungs-App für die chinesische Regierung dient und sämtliche Daten der Nutzer auswertet und an die Behörden weiterleitet [Mon]. Dabei steht es im engen Zusammenhang mit Chinas *Social Credit System* [Dtk]: seit 2014 werden sämtliche Daten über Chinas Bevölkerung durch dessen Regierung gesammelt und ausgewertet. Jedem Bürger wird ein Punktestand zugeordnet; bei gutem und regierungstreuen Verhalten steigt die Punktzahl, bei Verstößen gegen das Gesetz oder die Ideologie der *Kommunistischen Partei* werden Punkte abgezogen. Diese haben Einfluss auf sämtliche Bereiche des Lebens der Bürger in China, wie mögliche Beförderungen, Miete, Gehalt, Lohn und vieles mehr [Dtk]. WeChat spielt für die Überwachung der chinesischen Bevölkerung eine zentrale Rolle, verletzt ganz klar die Persönlichkeits- und Datenschutzrechte jener betroffenen Nutzer und wurde deshalb von vielen Datenschützern, Journalisten [Coc19], [Dou17], [Heg17] und nicht zuletzt Menschenrechtsorganisationen wie *Amnesty International* [Heg19] schärfstens kritisiert. WeChat zeigt, zu welchem Umfang Messenger-Dienste als Mittel zur Überwachung eingesetzt werden können.

[2] `https://twitter.com/Snowden/status/661313394906161152` (abgerufen am 4. September 2022)
[3] `https://github.com/signalapp/` (abgerufen am 4. September 2022)

5. Testumgebung und Aufbau: Analyse der Metadaten

Das folgende Kapitel erläutert den grundlegenden Aufbau der Tests und damit den praktischen Teil dieser Arbeit. Ziel der Tests ist es herauszufinden, wie viele und welche Art von Metadaten bei den einzelnen Protokollen der Messenger, sowie der Corona-Warn-App übermittelt werden. Des Weiteren werden die relevanten Metadaten den einzelnen Schichten im ISO/OSI-Referenzmodell zugeordnet und damit geprüft, auf welcher Ebene man wie viele Informationen über die Metadaten erhält. Dadurch kann eine Aussage getroffen werden, ob relevante Metadaten für alle einsehbar sind, welche den Netzwerkverkehr mitschneiden, oder lediglich für den Server, der die Daten verarbeitet.

5.1. Testsysteme/-plattformen

Da der Fokus dieser Arbeit auf der Betrachtung von Messengern liegt, wurden zwei unterschiedliche Mobiltelefone getestet. Des Weiteren dienten als Testplattform, aufgrund der großen Verbreitung unter Privatpersonen und mittelständischen Unternehmen, ein Computer mit dem Betriebssystem Windows, als auch Linux, welches über eine virtuelle Maschine (VM) läuft.

Es soll überprüft werden, ob die unterschiedlichen Plattformen oder Betriebssysteme anhand der Metadaten erkannt werden können und damit Rückschlüsse auf den Nutzer möglich sind. Dasselbe Ziel wurde bei den unterschiedlichen Browsern verfolgt.

5.1.1. Testsystem A - Computer (Windows 10)

Als grundlegendes Testsystem diente ein Desktopcomputer mit installiertem Betriebssystem Windows 10 Home. Die genauen Systemspezifikationen sind im Anhang der Abbildung 19 zu entnehmen. Auf diesem System lief zum Mitschneiden des Netzwerkverkehrs das frei verfügbare Netzwerkanalysetool Wireshark [4](Abb. 20). Als Browser wurden sowohl Google Chrome (Abb. 21a), als auch Firefox (Abb. 21b) verwendet.

WhatsApp dient für diese Arbeit als Referenzapp und Ausgangspunkt zum Analysieren und zum Vergleich der Sicherheit von Matrix. Gleichzeitig konnte damit auch die Sicherheit des Protokolls von WhatsApp selbst im Bezug auf Metadaten geprüft werden. Auf dem Desktop-PC geschah dies zum einen mit der WhatsApp Desktop-App (Version 2.2037.6), als auch innerhalb der Browser mittels WhatsApp Web (Version 2.2043.8) über `https://web.whatsapp.com/` (abgerufen am 4. September 2022).

Matrix selbst wurde über den Web- und Desktop-Client Element getestet. Dieser hieß ursprünglich Riot Web und baut auf der Matrix-React-SDK auf, mit einem Fokus auf Leistung und Benutzerfreundlichkeit[5], wodurch er am interessantesten für eine breite Masse von Endnutzern und damit für diese Arbeit ist. Sämtliche Versionen von Elements auf dem Testsystem A liefen auf Version 1.7.10 (Abb. 22).

[4] `https://www.wireshark.org/` (abgerufen am 4. September 2022)
[5] `https://matrix.org/docs/projects/client/element` (abgerufen am 4. September 2022)

5.1.2. Testsystem B - Virtuelle Maschine Linux (Ubuntu 18.04)

Die Entscheidung, die Testläufe und Programme auch auf Linux auszuführen hatte mehrere Gründe. Zum einen sollte überprüft werden, ob ein unterschiedliches Betriebssystem für Unterschiede in den Metadaten der übertragenden Pakete sorgt und darüber Informationen über den Nutzer gewonnen werden können. Des Weiteren ist es wesentlich einfacher, den Netzwerkverkehr auf einem schlanken, neu-installierten Linux-System mitzuschneiden, da hier wesentlich weniger Hintergrundprozesse als auf dem Windows Desktop-Computer laufen (besonders, da das Testsystem A (5.1.1) der über Jahre vom Autoren privat genutzte Computer ist). Aus Effizienzgründen, aber auch aufgrund der Portabilität des gesamten Systems, lief Linux über eine virtuelle Maschine (VM). Geplant war auch, diese zunächst auf dem Testsystem A einzurichten und dann auf den Rechnern der Universität zu testen. Aufgrund der Corona-Pandemie und den damit einhergehenden Beschränkungen und Risiken, wurde sich jedoch dagegen entschieden.

Als VM wurde Oracle VM VirtualBox (Abb. 23) verwendet. Auf dieser wurde das Linux Betriebssystem Ubuntu 18.04 ausgeführt (Abb. 24). Zum Installieren wurde die ISO "ubuntu-18.04.5-desktop-amd64.iso" von `https://releases.ubuntu.com/18.04/` (abgerufen am 4. September 2022) verwendet. Bei der Installation wurde die minimale Installation aus den vorher genannten Gründen gewählt, um das System möglichst schlank zu halten. Dies ist in Abbildung 5 dargestellt. Die Systeminformationen sind zusätzlich im Anhang in Abbildung 25, sowie ausführliche Systeminformationen mittels des Befehls "inxi -F" in Abbildung 26 zu sehen. Auf dem Ubuntu-System der VM wurde ebenfalls Wireshark (Abb. 27) installiert, sowie die selbe Desktop- und Browservesrion von Matrix Element (Abb. 22), wie im Testsystem A (5.1.1).

Abb. 5: Testsystem B (5.1.2) - Ubuntu, minimale Installation

5.1.3. Testsystem C - Mobiltelefon (Umidigi F2)

Des Weiteren wurden zum Testen zwei Mobiltelefone genutzt. Diese stellen den Anwendungsfall des klassischen Gebrauchs von Messengern dar, da die meisten Personen Messenger und insbesondere WhatsApp auf ihrem Handy nutzen. Die Daten dieser Systeme wurden nicht direkt aufgezeichnet. Sie dienten lediglich zum Senden und Empfangen der Nachrichten vom Testsystem A und B bei der Nachrichtenübertragung der Messenger. In dem Fall, in dem Daten von einem Mobiltelefon aufgezeichnet werden mussten, wurde das virtuelle Android im Testsystem E (5.1.5) verwendet.

Als Referenzsystem wurde das Umidigi F2, mit der zum derzeitigen Stand aktuellsten Version von Android – Android 10 – genutzt. Die Spezifikationen des Betriebssystems und der Hardware das Mobiltelefons sind der Abbildung 28 im Anhang zu entnehmen. Da alle getesteten Applikationen auf den meisten aktuellen Mobiltelefonen lauffähig sind, ist die Wahl des konkret verwendeten Systems weniger relevant, solange die Applikationen auf diesem unterstützt werden.

5.1.4. Testsystem D - Mobiltelefon (Umidigi F1)

Um mögliche Unterschiede in den aufgezeichneten Metadaten festzustellen, wurde auch ein zweites Mobiltelefon benutzt. Dieses ist die Vorgängerversion des Handys aus dem Testsystem C (5.1.3) und damit sehr nah an dessen Architektur. Somit würden mögliche Unterschiede in den Metadaten wesentlich mehr über das jeweilige Mobiltelefon verraten. Die Spezifikationen sind ebenfalls dem Anhang zu entnehmen, in Abbildung 29.

5.1.5. Testsystem E - Virtuelle Maschine (Android 9)

Als letztes Testsystem wird ein Android Handy über Oracle VM VirtualBox (Abb. 23) simuliert. Zusätzlich zu den beiden Mobiltelefonen für die Testsysteme 5.1.3 und 5.1.4 wurde noch ein virtuelles Android benötigt, um den Netzwerkverkehr des Android-Systems mitschneiden zu können. Gerade der Datenverkehr der CWA hätte sonst aufgrund der limitierten technischen Ausstattung nur schwer aufgezeichnet werden können. Auf dem Testsystem läuft Android in Version 9. Zum Installieren wurde die ISO "android-x86_64-9.0-r2.iso" von `https://www.fosshub.com/Android-x86.html` (abgerufen am 4. September 2022) heruntergeladen.

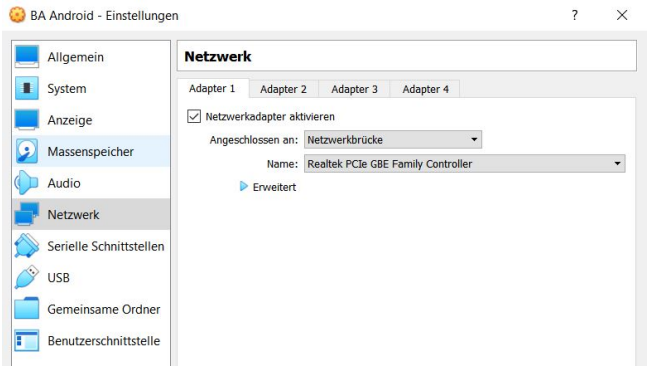


Abb. 6: Testsystem E (5.1.5) - Android, VirtualBox Netzwerkbrücke

Wichtig ist hierbei anzumerken, dass in den Netzwerkeinstellungen der VM nicht der standardmäßig ausgewählte NAT-Modus verwendet wurde, sondern eine Netzwerkbrücke, zu sehen in Abbildung 6. Netzwerkbrücken koppeln lokale Netzwerke auf Schicht 2 des ISO/OSI-Referenzmodells (Abschnitt 2.3.1). Dabei werden entweder homogene Netze vom gleichen Typ, oder heterogene Netze unterschiedlichen Typs über eine Brücke gekoppelt. Zur Hauptaufgabe einer Netzwerkbrücke zählt die Trennung des Intranetz-Verkehrs eines LAN vom Internet-Verkehr. Dadurch erhält die VM auf dem Testsystem A (5.1.1) ihre eigene IP-Adresse, wodurch Wireshark diese Adresse filtern kann und somit auf System A sämtlichen Netzwerkverkehr des simulierten Android und gleichzeitig nur diesen sehen kann. Da es sich bei dem Android-System nur um eine virtuelle Maschine handelt, kann in dem System weder Bluetooth, noch die Standortfreigabe oder GPS aktiviert werden. Diese technische Limitation konnte nicht zuletzt aufgrund der aktuellen Lage nicht überwunden werden und führte zumindest bei den Test der Corona-Warn-App zu gewissen Einschränkungen, welche in dem entsprechenden Abschnitt (8.5) näher erläutert werden. Dennoch diente dieses Testsystem sehr gut der Aufzeichnung des Datenverkehrs von Mobiltelefonen und es war in den Test möglich, auch ohne größeren Aufwand Datenverkehr der Messenger mitzuschneiden.

5.2. Funktion von Wireshark

Folgend wird die wesentliche Funktionalität von Wireshark beschrieben, welche grundlegend für diese Arbeit ist. Wireshark ist ein beliebtes, kostenloses Programm zur Aufzeichnung und Interpretation von Netzwerkverkehr. Es simuliert in dieser Arbeit einen MitM-Angriff, wie er in Abschnitt 2.4.1 beschrieben ist. Praktisch wird für solch einen Angriff typischerweise *Ethercap*[6] genutzt. Ziel dieser Arbeit ist jedoch lediglich den Datenverkehr aufzuzeichnen, sowie auszuwerten und keinen tatsächlichen MitM-Angriff durchzuführen, weshalb Wireshark besser geeignet ist.

5.2.1. Wireshark Netzwerkschnittstellen

Wireshark zeichnet sämtliche, dem Programm sichtbaren Datenpakete auf, welche vom System empfangen werden. Beim Start des Programms sieht man zunächst eine Übersicht aller verfügbaren Schnittstellen (engl. interfaces), zu erkennen in Abbildung 7:

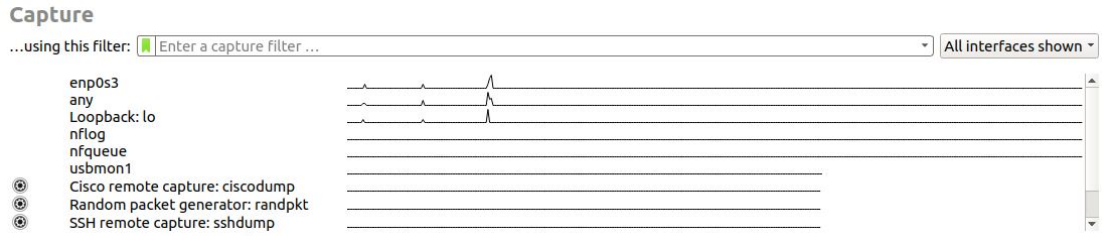


Abb. 7: Wireshark - Aufzeichnen (engl. Capture) Schnittstellen

Der Reiter "All interfaces shown" zeigt im Testsystem B (5.1.2) eine Auswahl von drei Möglichen Schnittstellenarten zum Aufzeichnen des Datenverkehrs, dargestellt in Abb. 8:

- **Wired:** Kabelanschluss des Netzwerkadapters
- **USB:** Aufzeichnen von USB-Verkehr (Maus,...)
- **External Capture:** WiFi, Bluetooth, Remote,...

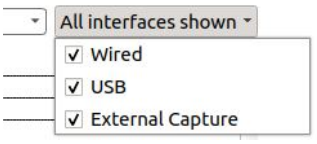


Abb. 8: Wireshark - All interfaces shown

Für den Testdurchlauf sind lediglich die Schnittstellen "Wired" von Interesse. Ist diese die einzig ausgewählte Schnittstelle, sind folgende verbleibende Schnittstellen zu sehen:

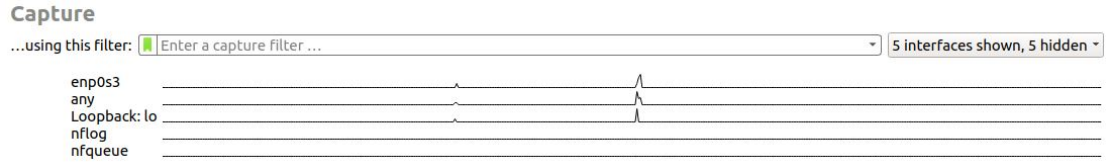


Abb. 9: Wireshark - Aufzeichnen "Wired" Schnittstellen

[6] `https://www.ettercap-project.org/` (abgerufen am 4. September 2022)

Diese 5 verbleibenden Schnittstellen lassen sich wie folgt erklären:

- **enp0s3:** Ethernet-Schnittstelle der VM
- **any:** virtuelles Interface, zeichnet alle Pakete auf
- **Loopback: lo:** Pakete zum eigenen System
- **nflog:** userspace-Pakete, die vom Kernel-Paketfilter protokolliert wurden
- **nfqueue:** individuelle Paketsteuerung

Zum Aufzeichnen des Datenverkehrs wird die Schnittstelle "enp0s3" verwendet.

Man kann sich die Konfiguration aller Netzerk-Schnittstellen des Systems mittels des Befehls *"ifconfig -a"* in dem Terminal ausgeben lassen. Dies erzeugt folgende Ausgabe:

```
$ ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::ce8a:3164:8c0d:dc28  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4b:aa:7e  txqueuelen 1000  (Ethernet)
        RX packets 3287  bytes 3600390 (3.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1561  bytes 194917 (194.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 362  bytes 36644 (36.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 362  bytes 36644 (36.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Code 1: Ausgabe "ifconfig" im Testsystem B

In der Ausgabe sind die beiden standardmäßigen Netzerk-Schnittstellen der Ubuntu VM zu erkennen. Diese decken sich mit den in Wireshark angezeigten Schnittstellen. Die Messenger werden über die Ethernet-Schnittstelle "enp0s3" Pakete versenden.

5.2.2. Wireshark Filter

Es ist in Wireshark möglich Filter anzuwenden, durch welche nur bestimmte Pakete aufgezeichnet werden. Dies ist in dem Anwendungsfall von Bedeutung, sobald die IP-Zieladresse der Anwendung oder des Servers gefunden wurde, zu welcher die jeweiligen Pakete der Anwendung gesendet werden. Dadurch kann gezielt nur der Netzwerkverkehr der Anwendung betrachtet werden, welcher untersucht werden soll, ohne Pakete anderer Prozesse mit aufzuzeichnen, welche ebenfalls in der Schnittstelle "enp0s3" von Wireshark aufgezeichnet werden. Näheres zu diesem Problem im folgenden Abschnitt 5.3.1.

In Wireshark gibt es zwei Arten von Filtern:

- **Mitschnittfilter** (engl. capture filter): filtert, welche Pakete aufgezeichnet werden
- **Anzeigefilter** (engl. display filter): Filtern der aufgezeichneten Pakete

Letztere können zu jeder Zeit angewandt werden und dienen lediglich dazu, die Übersichtlichkeit zu erhöhen, ohne wie die Mitschnittfilter die eigentliche Aufzeichnung zu begrenzen. Die Mitschnittfilter können wiederum nur vor dem Beginn des Aufzeichnens festgelegt werden und begrenzen die Menge des aufgezeichneten Datenverkehrs, abhängig vom ausgewählten Filter. Für diese Arbeit sind vor allem die Mitschnittfilter relevant. Sie werden in der Eingabeleiste über der Vorschau zu den einzelnen Schnittstellen eingegeben, zu sehen in Abbildung 10:

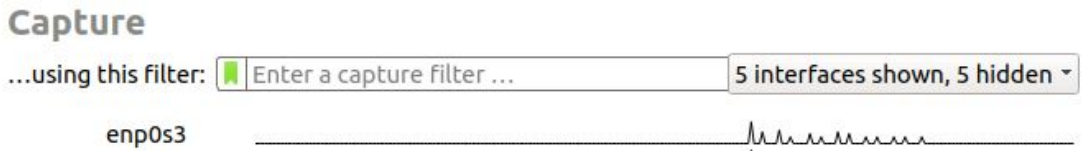


Abb. 10: Wireshark - Eingabe Mitschnittfilter (engl. Enter a capture filter)

Ein Aufzeichnungs- oder Mitschnittfilter ist mindestens ein primitiver Ausdruck, verbunden mittels Konjunktionen (and/or) und optional umgekehrt durch (not)[7,8]. Ein primitiver Ausdruck ist für jedes aufgezeichnete Paket entweder wahr oder falsch. Für den Anwendungsfall genügt ein einfacher Filter, welcher lediglich die Pakete von und zu einer bestimmten IP-Adresse mitschneidet:

```
host <IP der Host-Adresse>
```

Dieser Filter kann später verwendet werden, um gezielt den Netzwerkverkehr der zu betrachtenden Anwendung mitzuschneiden. Ohne weitere Vorkenntnisse über die Host-Adresse geht dies jedoch nicht, weshalb zunächst Netzwerkverkehr ohne Filter mitgeschnitten werden muss.

5.2.3. Wireshark Aufzeichnung Datenverkehr/ Pakete

Folgend wird beispielhaft die Aufzeichnung der Datenpakete durch Wireshark beschrieben. Dieses Vorgehen dient als Grundlage für die durchgeführten Testdurchläufe und zeigt gleichzeitig, wie Wireshark funktioniert und welche Informationen man mittels Wireshark über den Netzwerkverkehr in Erfahrung bringen kann. Diese Informationen sind in zwei Bereiche unterteilt: dem Paketlistenfenster und den Paketdetails- und Bytes.

[7] `https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureFilterSection.html` (abgerufen am 4. September 2022)
[8] `http://www.tcpdump.org/manpages/pcap-filter.7.html` (abgerufen am 4. September 2022)

Das Paketlistenfenster

Nach Beginn des Aufzeichnens werden alle durch Wireshark mitgeschnittenen Pakete in dieser Liste angezeigt. Dabei stellt jede Zeile ein aufgezeichnetes Paket dar. Ein Beispiel dazu ist in Abbildung 11 zu sehen:

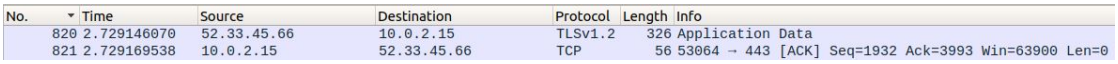


Abb. 11: Wireshark - Die Paketliste

Es folgt eine kurze Erläuterung der einzelnen Spalten[9]:

- **No.:** Nummer des aufgezeichneten Pakets
- **Time:** Zeitstempel des Pakets (hier Zeit seit Beginn der Aufzeichnung)
- **Source:** Absenderadresse des Pakets (IP)
- **Destination:** Zieladresse des Pakets (IP)
- **Protocol:** Protokollname (Kurzform) des verwendeten Protokolls
- **Length:** Länge des Pakets in Bytes
- **Info:** Zusätzliche Informationen über den Inhalt des Pakets

Genauere Informationen über ein aufgezeichnetes Paket erhält man, wenn man dieses auswählt. Diese Informationen dienen meist nur zum Identifizieren eines gesuchten Datenpakets, sowie als Übersicht über den gesamten aufgezeichneten Datenverkehr.

Die Paketdetails- und Bytes

Wird aus der vorherigen Beispielaufzeichnung das Paket Nummer 820 ausgewählt, erscheint folgende Übersicht, zu sehen in Abbildung 12:

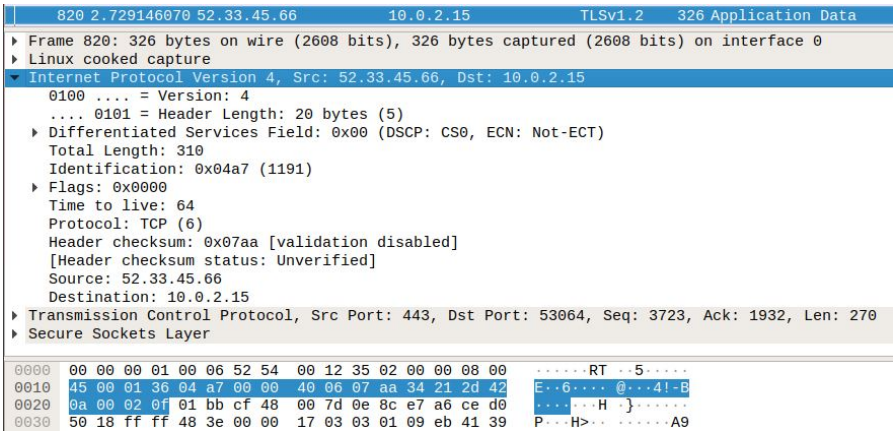


Abb. 12: Wireshark - Paketdetails und -bytes

[9] `https://www.wireshark.org/docs/wsug_html_chunked/ChUsePacketListPaneSection.html` (abgerufen am 4. September 2022)

Der Datenverkehr, oder konkret der Inhalt der einzelnen Pakete, liegt Wireshark zunächst nur als binärer Datensatz vor. Dieser wird für den Anwender als sogenannter *Hexdump* dargestellt, zu sehen im unteren Teil der Abbildung 12. Hexdump bezeichnet die hexadezimale Darstellung der vom Computer verarbeiteten Daten. Jede Zeile enthält 8 Bytes, dargestellt als 16 hexadezimale Bytes und 16 ASCII-Bytes, welche analog in Bits umgerechnet und angezeigt werden können. Nicht darstellbare Bytes der ASCII-Bytes werden als Punkt (".") für den Anwender angezeigt[10]. Wireshark übersetzt nun diese binären Informationen in die Informationen zu den entsprechenden Netzwerkprotokollen und einzelnen Schichten des OSI-Referenzmodells, zuvor besprochen in Abschnitt 2.3. Zu sehen sind diese Informationen über das Paket im Paketdetailsfenster in Abbildung 12. Sämtliche Bytes werden entsprechend ihrer Bedeutung im jeweiligen Protokoll interpretiert und als lesbare Information dargestellt.

Jede Schicht wird zur Übersicht in einem einzelnen Bereich dargestellt, welcher für weitere Informationen aufgeklappt werden kann (zu sehen durch ein "▸"). Zunächst werden allgemeine Informationen über das Paket angegeben, zu sehen im ersten Bereich "▸ Frame [No.]:" In diesem Ausschnitt ist zu sehen, dass das aufgezeichnete Datenpaket das 820. seit Beginn der Aufzeichnung ist, eine Länge von 326 Bytes (2608 Bits) hat, von denen auch alle aufgezeichnet wurden und das Paket die Schnittstelle 0 passierte, welches in dem Fall "enp0s3" ist. Diese allgemeinen Informationen über das Paket fasst Wireshark zusammen, ohne die aufgezeichneten Bytes zu interpretieren. Dies geschieht in den folgenden Ebenen.

Im Beispiel wurde die Vermittlungsschicht ausgewählt. Dabei hebt Wireshark auch die entsprechenden Bytes zu diesen Informationen hervor. Dies geht nicht nur mit allen Bytes des jeweiligen Protokolls, sondern auch mit den einzelnen entsprechenden Informationen der Bytes. So interpretiert man beispielsweise nach dem ausgewählten Internet Protocol Version 4 (IPv4) das erste Hexadezimalbyte "45_{HEX}" binär als "01000101_{BIN}". Die ersten 4 Bits "0100" des Bytes bedeuten, dass das IP-Paket die Version 4 hat und die letzten 4 Bits "0101", dass die Länge des gesamten Headers inklusive Optionen 20 Bytes ist, welches ein Vielfaches von 32 Bit ist ($0101_{\text{BIN}} = 5_{\text{DEZ}} \cdot 32Bit = 160Bit = 20Byte$). Die von Wireshark interpretierten Informationen zu der Protokollschicht stimmen in dem Fall mit der Interpretation der Protokollschicht überein.

Dieses Beispiel zeigt exemplarisch, dass Wireshark die Informationen korrekt interpretieren kann und somit Wireshark nicht nur genutzt werden kann, um die Datenpakete aufzuzeichnen, sondern auch um deren Informationen entsprechend der jeweiligen Protokolle zu interpretieren. Somit müssen die Informationen des Hexdump nicht selbst per Hand ausgewertet werden und es kann sich darauf konzentriert werden, wie man diese Informationen nutzen kann, um Daten über Nutzer zu gewinnen. Des Weiteren generiert Wireshark auch zusätzliche Informationen über die Protokolle und empfangenen Pakete, wie z.B. Checksummenvalidierungsinformationen, Antwortzeiten, TCP-Analysen oder weitere. Diese zusätzlich generierten Informationen werden durch eckige Klammern ("[...]") gekennzeichnet.

[10] `https://www.wireshark.org/docs/wsug_html_chunked/ChUsePacketBytesPaneSection.html` (abgerufen am 4. September 2022)

5.3. Ablauf der Test: Aufzeichnen des Netzwerkverkehrs

Ziel der Tests ist es mittels Wireshark die versendeten Pakete der Kommunikationspartner abzufangen und dabei die gewonnenen Informationen eines MitM-Angriffs zu simulieren. Nach erfolgreichem Aufzeichnen des Datenverkehrs, sollen die erhaltenen Informationen ausgewertet und daraus Rückschlüsse auf die Sicherheit der betrachteten Applikationen, insbesondere im Bezug auf deren Metadaten, getroffen werden.

5.3.1. Geringer Hintergrunddatenverkehr

Da Wireshark ohne weitere Konfiguration sämtlichen Datenverkehr einer Schnittstelle aufzeichnet, jedoch nur der Datenverkehr der Messenger und insbesondere nur der von bestimmten Aktionen (wie das Versenden oder Empfangen von Nachrichten oder Statusänderungen) mitgelesen werden soll, muss eine Umgebung geschaffen werden, in der keine Daten von anderen Prozessen versendet werden. Diese Daten oder Pakete, welche nicht relevant für die Testläufe sind, werden als *Hintergrunddatenverkehr* bezeichnet. Diesen gilt es somit zu minimieren oder im besten Fall während des Testdurchlaufs gänzlich zu vermeiden. Wie bereits in der Erläuterung des Testsystems B (5.1.2) beschrieben, eignet sich als Testumgebung daher das neu-installierte Ubuntu-System auf der VM am Besten, da dort nur minimale Hintergrundprozesse laufen. Dazu können üblicherweise System- oder Programmupdates gehören. Diese werden oft nach dem Starten des Systems automatisch ausgeführt. Es empfiehlt sich also, einige Minuten nach dem Systemstart mit den Testdurchläufen zu warten und gegebenenfalls manuell Updates durchzuführen. In Ubuntu geht dies beispielsweise mit den Kommandozeilenbefehlen

```
\$ sudo apt-get update
\$ sudo apt-get upgrade
\$ sudo apt-get dist-upgrade
```

für Programm- und Kernelupdates. Weiterer Hintergrunddatenverkehr sollte individuell betrachtet und minimiert werden. Diese Schritte sollten jedoch eine gute Grundlage liefern, um ungestört den Netzwerkverkehr der Applikationen auf dem Testsystem B (5.1.2) aufzuzeichnen. Des Weiteren können, wie vorher in Abschnitt 5.2.2 beschrieben, Filter angewandt werden, um gezielt den Datenverkehr der versandten Pakete der Applikationen zu filtern. Dies bietet sich jedoch nur an, sobald bereits Informationen zu diesen ermittelt wurden, wie beispielsweise die Host-Adresse des jeweiligen Servers.

Zuletzt sollte noch angemerkt werden, dass nicht zwingend sämtlicher Datenverkehr einer Applikation betrachtet und aufgezeichnet werden soll. Ist das Ziel, die anfallenden Metadaten beim Versenden von Nachrichten zu beurteilen, so können andere versendete Pakete und Datenverkehr der Applikationen auch in gewisser Weise als störender Hintergrunddatenverkehr angesehen werden. Dazu können beispielsweise Updates oder aktualisierte Informationen der Applikation zählen, aber auch Handshakes und andere Daten beim Etablieren einer Verbindung, sowie sonstige Daten. Die aufgezeichneten Datenpakete können zunächst nur mittels oberflächlicher Betrachtung und ohne weitere Informationen nicht der jeweiligen Aktion in der Applikation zugeordnet werden (wie z.B. dem Versenden von Nachrichten). Dies geschieht zunächst vor allem durch gezieltes

Timing bei nicht vorhandenem Hintergrunddatenverkehr, wobei dies im folgenden als *still* oder *stille Verbindung/ Kanal* bezeichnet wird. Der folgende Abschnitt erläutert das genaue Vorgehen.

5.3.2. Ablauf der Test

Ausgangssituation vor jedem Testdurchlauf ist, dass auf dem zu testenden System (meist Testsystem B (5.1.2)) sowohl Wireshark läuft, als auch die zu testende Applikation. Neben diesen beiden Anwendungen, sollte keine weitere Anwendung im Vordergrund laufen. Zudem läuft auf einem anderen Testsystem dieselbe Anwendung, welche während des Tests miteinander Daten austauschen. Als nächstes sollte, wie zuvor in Abschnitt 5.3.1 beschrieben, das Testen nicht direkt nach dem Systemstart geschehen und Hintergrunddatenverkehr minimiert werden, sodass ein stiller Kanal bei der zu betrachtenden Schnittstelle vorliegt. Dies kann im Vorschaufenster von Wireshark zur jeweiligen Schnittstelle betrachtet oder gegebenenfalls durch eine Testaufzeichnung von diesem geprüft werden. Nun zum eigentlichen Ablauf der Tests.

Ein Testdurchlauf beginnt mit dem Aufzeichnen des Datenverkehrs und endet mit dem Beenden der Aufzeichnung. Nach Start des Tests bietet es sich an, einige Sekunden zu warten, um erneut sicher zu stellen, dass keine weiteren Daten in dieser Zeit versendet werden. Als nächstes wird in einer Applikation auf einem der beiden Testsysteme eine Aktion ausgeführt, welche einen Datenaustausch mit der gleichen Applikation auf dem anderen Testsystem zur Folge hat. Ziel ist es, diesen Datenaustausch mitzuschneiden. Dabei werden insbesondere nur Daten zwischen beiden kontrollierten Testsystemen ausgetauscht, da das Senden von einem Testsystem zu irgendeinem anderen System nicht immer validiert oder kontrolliert werden kann. Für die Testdurchläufe selbst wurde somit eine kontrollierte reproduzierbare Umgebung geschaffen, deren Spezifikationen bekannt sind. Nach Ausführen der zu überprüfenden Aktion sollte Wireshark deren resultierenden Datenverkehr aufzeichnen. Sind alle Pakete erfolgreich empfangen und liegt erneut eine stille Verbindung vor, so sollte erneut einige Sekunden gewartet werden, um sicher zu stellen, dass keine weiteren Pakete versandt werden. Ist dies der Fall, kann die Aufzeichnung durch Wireshark beendet werden und der Testdurchlauf gilt als abgeschlossen. Als Resultat sollte eine Menge von Paketen aufgezeichnet wurden sein, welche in Wireshark aufgelistet werden und im folgenden Teil der Arbeit analysiert werden können.

5.3.3. Validierung der Zuordnung der aufgezeichneten Pakete

Betrachtet man lediglich die aufgezeichneten Pakete eines Testdurchlaufes, so kann man diese ohne zusätzliches Wissen nicht eindeutig einer Aktion der Applikation zuordnen. Dies liegt insbesondere daran, dass bei den betrachteten Messengern (WhatsApp und Matrix/ Signal) die Anwendungsdaten verschlüsselt werden. Die eindeutige Zuordnung erfolgt hier vor allem durch drei Überlegungen: Ausschlussverfahren und Timing, sowie mehrfaches Wiederholen der Testdurchläufe. Diese werden im folgenden erläutert, sowie kritisch beleuchtet.

Ausschlussverfahren

Liegt im gesamten Zeitraum des Testdurchlaufs ein stiller Kanal vor – also werden keine weiteren Hintergrunddaten aufgezeichnet und werden erst Pakete im Zeitraum nach dem Ausführen der jeweiligen Aktion aufgezeichnet – so kann es sich nur um Pakete handeln, die zu dieser Aktion zugeordnet werden können.

Das tatsächliche Vorhandensein eines stillen Kanals, auch insbesondere ohne andere von der Anwendung selbst versandte Daten, kann in der Praxis problematisch nachzuweisen und zu jedem Zeitpunkt sicherzustellen sein. Hilfreich ist hierfür die Verwendung von Filtern, wie in Abschnitt 5.2.2, um zumindest die Hintergrunddaten von anderen Applikationen zu ignorieren. Die Applikationen selbst können jedoch auch noch weitere Daten versenden, welche unabhängig von der ausgeführten Aktion sind. Die eindeutige Zuordnung unterstützen daher zusätzlich die beiden folgenden Überlegungen.

Timing

Wird in den Sekunden vor Beginn des tatsächlichen Aufzeichnens von Paketen (also während ein stiller Kanal vorliegt) keine Aktion ausgeführt und werden Pakete erst ab dem Zeitpunkt aufgezeichnet, ab dem die Aktion beendet wurde und somit ein Datenverkehr als Folge zu erwarten ist, sowie ohne weitere Aktion erneut ein stiller Kanal vorliegt, so ist davon auszugehen, dass die aufgezeichneten Pakete der ausgeführten Aktion zugeordnet werden können.

Mehrmaliges Wiederholen und reproduzierbare Testergebnisse

Können beide zuvor beschriebenen Überlegungen für einen Testdurchlauf bestätigt werden und ändert sich dies, sowie das Ergebnis des Testdurchlaufes, was eine bei gleichen oder ähnlichen Aktionen gleiche oder ähnliche Menge von mitgeschnittenen Paketen und enthaltenen Informationen ist, auch nach mehrmaligem Wiederholen der Testdurchläufe nicht, so ist dies ein noch stärkeres Argument dafür, dass die aufgezeichneten Pakete der ausgeführten Aktion zugeordnet werden können. Je öfter ein Testdurchlauf wiederholt wird und dabei das gleiche Testergebnis aufweist, desto stärker wird das Argument, dass die aufgezeichneten Pakete der in der Anwendung ausgeführten Aktion zuzuordnen sind.

Zwei Testergebnisse werden als *gleich* bezeichnet, wenn die exakt gleichen Aktionen vergleichbare Ergebnisse liefern. Dazu zählt z.B. das mehrmalige Versenden ein- und derselben Nachricht.

Zwei Testergebnisse werden als *ähnlich* bezeichnet, wenn zwei leicht abgeänderte Aktionen zwei vergleichbare Ergebnisse liefern. Dazu zählt z.B. das mehrmalige Versenden annähernd gleicher Nachrichten, wobei in diesem Beispiel die Anzahl (”a”,”aa”,”aaa”,…) oder Art der Zeichen (”aa”,”ab”,”ba”,…), oder beides leicht abgeändert werden kann und dennoch als Ergebnis vergleichbare Pakete auftreten.

So sollte üblicherweise dieselbe Anzahl von Paketen, sowie die selbe Art von Paketen oder genauer deren verwendeten Protokolle festgestellt werden. Diese sollten auch in etwa der gleichen Zeit nach Ausführen der Aktion eingegangen sein, wobei hier Schwankungen durch unterschiedliche Netzwerk- oder Serverauslastung zu berücksichtigen sind. Zudem sollte die Größe der einzelnen Pakete annähernd gleich sein. Dabei sei anzumerken, dass die verwendeten Verschlüsselungsalgorithmen aufgrund von sich dynamisch ändernden Schlüsseln bei gleicher Eingabe unterschiedliche Chiffren von unterschiedlicher Länge erzeugen, wodurch bei gleicher Eingabe auch die Länge der versendeten Information und damit des Pakets unterschiedlich sein kann. Diese Unterschiede sollten jedoch nur im Bereich von wenigen Bytes sein, wobei man hier keine allgemeingültige Zahl als Abweichung angeben kann. Die Quell- und Zieladresse, sowie die jeweiligen Ports sollten bei den jeweiligen Paketen der beiden Aufzeichnungen die selben sein, sowie Protokolle, deren Versionen, Flags, Time-to-Live sowie weitere protokollspezifische Eigenschaften, welche unabhängig vom möglichen Inhaltsänderungen innerhalb der Aktionen sind.

Ebendiese werden als unterschiedlich erwartet. Dazu gehören wie schon zuvor beschrieben Länge und Zeitpunkt des Einganges der Datenpakete, sowie Identifikations- und Cheksummeninformationen, Sequenz- und Acknowledgement-Nummern und insbesondere die verschlüsselten Daten der Applikationen. Ein Beispiel für die eben besprochenen Gemeinsamkeiten und Unterschiede gleicher Testergebnisse oder Pakete durch gleiche Aktionen ist in folgender Abbildung 13 zu sehen:

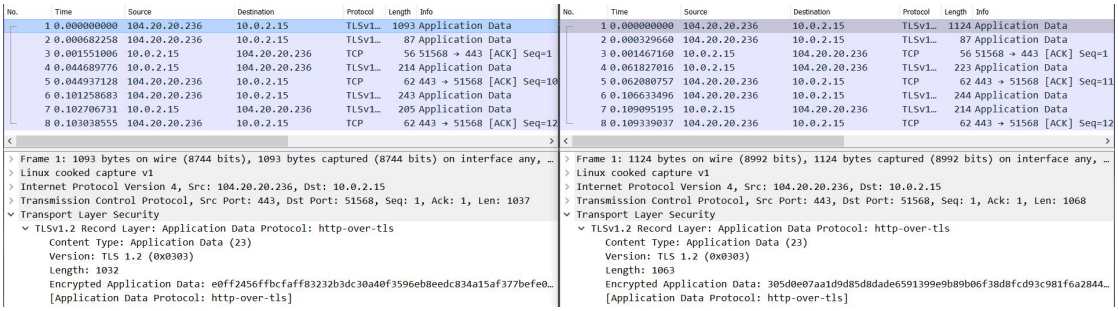


Abb. 13: Wireshark - Vergleich Paketliste und -details

Dies dient lediglich als Beispiel zur Veranschaulichung der zuvor besprochenen Aspekte der Gleichheit von Paketen und der eindeutigen Zuordnung dieser zu Aktionen. Es genügt zunächst zu sehen, dass die Anzahl der Pakete, die Art und bis auf einige Bytes Abweichung die Länge dieser, sowie Quell- und Zieladressen bei beiden Aufzeichnungen gleich sind. Die exakte Länge der Daten und der Inhalt der verschlüsselten Applikationsdaten unterscheidet sich, was zu erwarten ist. Konkreter wird darauf folgend in den einzelnen Kapiteln der zu prüfenden Applikationen eingegangen, sowie die Informationen und Rückschlüsse auf die Sicherheit der Anwendungen, welche die Testdurchläufe begründen. Bis hierhin sollte ein guter Überblick über den Ablauf und die Aussagekraft der Tests erfolgt sein. Dies wird nun in den folgenden Kapiteln praktisch angewandt.

6. WhatsApp

Zunächst soll der Marktführer näher betrachtet werden. WhatsApp dient hiermit als Ausgangspunkt und Referenz für Matrix. Die aus diesen Tests gewonnenen Erkenntnisse im Bezug auf das Aufzeichnen des Datenverkehrs nützen aber auch den folgenden Tests der Corona-Warn-App.

6.1. Installation von WhatsApp

WhatsApp kann für alle verbreiteten Mobiltelefone und Tablets im App-Store heruntergeladen werden. Zudem gibt es für Windows, Apple und Linux eine Desktop-Version. Diese funktioniert analog zu WhatsApp Web, welches unter `https://web.whatsapp.com/` (abgerufen am 4. September 2022) im Browser erreichbar ist. Es wird eine Verlinkung zu einem mobilen bestehenden WhatsApp-Account benötigt.

6.2. Sicherheit von WhatsApp

Zwar herrscht eine Ende-zu-Ende-Verschlüsselung, jedoch gilt nur ein Gerät des Nutzers als das Hauptgerät. Alle weiteren Client-Geräte (z.B. per WhatsApp-Web) senden ihre Nachrichten zunächst an das Hauptgerät, wo die Nachrichten erneut mit neuen Schlüsseln verschlüsselt werden und dann an den eigentlichen Empfänger übermittelt werden. Dies sorgt für längere Übertragungszeiten und der Notwendigkeit einer stabilen Verbindung zum Hauptgerät [SZ20, S. 408].

Zudem bietet WhatsApp die Möglichkeit, den Chatverlauf entweder intern auf dem Gerät selbst, oder extern in der eigenen Cloud. Dort liegen die Daten jedoch unverschlüsselt im Klartext vor, was das Prinzip der Vertraulichkeit (2.2.3) verletzt [BB16].

6.3. Prüfen der Sicherheit der Metadaten von WhatsApp

Geplant war eine Überprüfung der von WhatsApp verwendeten Protokolle zur Nachrichtenübertragung im Bezug auf Metadaten. Dafür sollte ein Android Handy über eine virtuelle Maschine simuliert werden, zuvor beschrieben als Testsystem E (5.1.5). Leider sorgten technische Probleme dafür, dass die Durchführung der Test nicht im Zeitrahmen der Arbeit gelang und aufgrund der anhaltenden Pandemie konnte der Netzwerkverkehr eines Handys wie Testsystem C (5.1.3) nicht in der Universität getestet und mitgeschnitten werden. Dies ist somit ein Ausblick dieser Arbeit: den Netzwerkverkehr von WhatsApp auf einem Mobiltelefon mitzuschneiden und auszuwerten.

Die Tests wurde folgend mittels WhatsApp Web durchgeführt und es konnten erfolgreich Informationen über WhatsApp und dessen Metadaten zusammengetragen werden. Dennoch müssen diese Ergebnisse nicht mit der Verwendung von WhatsApp auf dem Handy übereinstimmen.

6.4. Prüfen der Sicherheit der Metadaten von WhatsApp Web

Es folgen verschiedene Testabläufe zur Sicherheit der Nachrichtenübertragung von WhatsApp Web im Bezug auf Metadaten zwischen unterschiedlichen Testsystemen. WhatsApp Web kann in jedem Browser genutzt werden, theoretisch auch mobil. Da für die mobile Benutzung die eigene App wesentlich geeigneter ist, findet die Webvariante hauptsächlich Anwendung auf Desktopgeräten. Für diese eignet sich ebenfalls WhatsApp Desktop. Beide sind somit reelle Anwendungsfälle auf Arbeitsrechnern in Unternehmen, welche ihre interne Kommunikation über WhatsApp realisieren.

6.4.1. Referenztest Nachrichtenübertragung

Die folgend beschriebene Nachrichtenübertragung dient als Referenz für WhatsApp Web. Dabei wurden WhatsApp-Nachrichten mit einer durchschnittlichen Länge von 5-6 Zeichen vom Mobiltelefon (Testsystem C (5.1.3)) an das Testsystem B (5.1.2) gesendet. Auf diesem lief zum Aufzeichnen Wireshark, sowie zum Empfangen der Nachrichten der Webclient von WhatsApp in Firefox. Die Testergebnisse bleiben die gleichen, wenn Sender und Empfänger vertauscht werden, bis auf ebenfalls vertauschte Sender und Empfänger der Pakete, weshalb dieses Szenario hier nicht weiter betrachtet wird. Es wurden in wiederholten Testdurchläufen mehrfach teils verschiedene Nachrichten versandt, die sich sowohl in Größe, als auch Inhalt unterscheiden. Alle aufgezeichneten Datenpakete finden sich im Anhang unter NetCap 1 - 4. Das Ergebnis von NetCap 1 wird im folgenden erläutert.

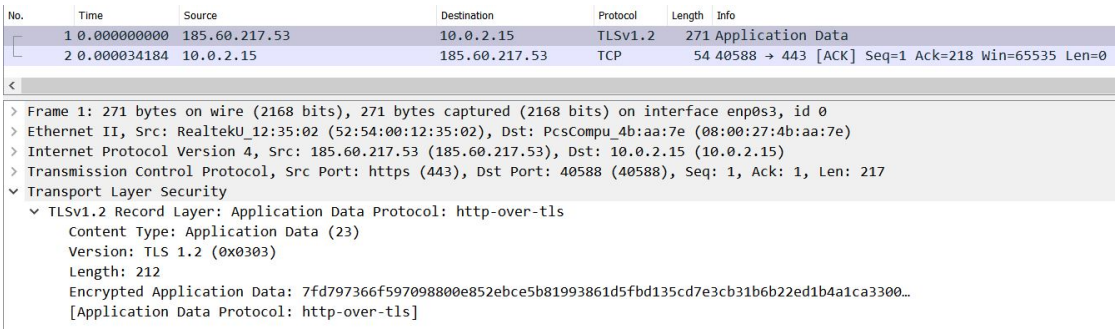


Abb. 14: Wireshark WhatsApp - Nachrichtenpakete

Abbildung 14 zeigt eine typische Aufzeichnung des Datenverkehrs von WhatsApp Web. Diese entspricht der Ansicht in Wireshark von NetCap 1 im Anhang. Empfängt der Client eine Nachricht, so erhält er ein TLSv1.2-Paket und sendet anschließend zur Bestätigung ein TCP-Paket zurück an den Server. Dies sind bereits Informationen von außen, welche man beim Mitschnitt des Datenverkehrs zur Bestimmung der Art des Datenverkehrs nutzen kann. Das TCP-Paket enthält kaum relevante Daten und im Gegensatz zum TLS-Paket auch keine neuen Metadaten über die eigentliche Nachricht. Es wird daher im Weiteren zwar mit aufgeführt, aber weniger detailliert betrachtet.

Folgend wird auf alle weiteren Daten eingegangen, welche Informationen über die eigentliche Nachricht und dem Paket liefern, mittels welchem diese versendet wurden.

Größe

Zunächst soll die Größe des Frames betrachtet werden. In diesem Beispiel beträgt diese 271 Bytes (2168 Bits). Diese hängt vor allem von der Größe der Applikationsdaten (*Application Data Length*) ab, welche wiederum von der Größe der versendeten Nachricht abhängt, wie später gezeigt wird. Dabei blieb die aufgezeichnete Größe nicht konstant. In manchen Testdurchläufen konnten für die selben Nachrichten aus zuvor durchgeführten Test unterschiedliche Größen der Pakete festgestellt werden. Diese blieben dann aber für eine Sitzung konstant und änderten sich erst nach Schließen des Browsers und erneuten Öffnen, zu sehen bei NetCap 3 (270 Bytes) und 4 (262 Bytes). Wodurch diese Unterschiede auftreten, konnte nicht genau bestimmt werden, da die eigentlichen Applikationsdaten verschlüsselt sind. In den meisten Fällen betrug die Größe der Pakete für die Nachrichtenübermittlung aber 271 Byte .

Zeit

Der Zeitpunkt des Eintreffens oder Versendens von Paketen zählt ebenfalls zu den aufgezeichneten Metadaten, welche Informationen über die Nachricht verraten. In der Abbildung 14 ist die relative Zeit seit Eintreffen des ersten Datenpakets angegeben. Dies ist lediglich eine andere Darstellungsform und der genaue Zeitpunkt des Aufzeichnens der Datenpakete kann vom Datum, bis zu neun Nachkommastellen der Millisekunde genau angegeben werden.

Sender & Empfänger: MAC-Adresse

Auf der untersten logischen Ebene im Paket – der Sicherungs-Ebene – sind die Media-Access-Control (MAC)-Adressen von Sender und Empfänger enthalten. Diese können Informationen über das jeweilige System verraten. Beide sind in der entsprechenden Sektion neben "Ethernet II" in Abbildung 14 ablesbar.

- MAC-Adresse Empfänger ("Dst"): 08:00:27:4b:aa:7e
- MAC-Adresse Sender ("Src"): 52:54:00:12:35:02

Die ersten 3 Bytes oder 6 Hexadezimalstellen der MAC-Adresse geben die Herstellerkennung (engl. Organizationally Unique Identifier (OUI)) an, welche von der IEEE vergeben wird. Die OUI ist stets gleich für den jeweiligen Hersteller der Netzwerkkarte. So konnte zumindest für die MAC-Adresse des Empfängers folgender Hersteller herausgefunden werden: "Pcs Systemtechnik GmbH", welche von Oracle Virtual Box verwendet wird[11].

[11] `https://macaddress.io/faq/how-to-recognise-an-oracle-virtual-machine-by-its-mac-address` (abgerufen am 4. September 2022)

Sender & Empfänger: IP-Adresse

Entsprechend den Spezifikationen der nächsten Ebene, können für die Vermittlungs-/Paketschicht die IP-Adressen von Sender und Empfänger in Erfahrung gebracht werden. Beide stehen neben der Spezifikation der Internet Protokoll Version, wobei hier auch gleich in Erfahrung gebracht werden kann, dass IPv4 verwendet wurde.

- IP-Adresse Sender ("Src"): 185.60.217.53
- IP-Adresse Empfänger ("Dst"): 10.0.2.15

Auch diese können Informationen über Sender und Empfänger preisgeben. So liefert die Auflösung des Namens des Senders folgende Adresse: "whatsapp-cdn-shv-01-ber1.fbcdn.net (185.60.217.53)". Damit handelt es sich eindeutig um einen von WhatsApp betriebenen Server. Die Namensauflösung übernimmt Wireshark für ihn bekannte Adressen. Des Weiteren kann diese Adresse wie zuvor beschrieben (5.2.2) nun als Filter genutzt werden, um gezielt weiteren Datenverkehr des Nutzers mit WhatsApp aufzuzeichnen.

Sender & Empfänger: Portnummern

Nicht zuletzt können die verwendeten Ports von Sender und Empfänger in Erfahrung gebracht werden. Diese befinden sich entsprechend auf der Transport-Schicht und erschließen Informationen über das verwendete Protokoll. Es wurde das Transmission Control Protocol (TCP) verwendet. Daneben finden sich die jeweiligen Ports.

- Port Sender ("Src"): 443
- Port Empfänger ("Dst"): 40588

Man kann erneut den Namen des Senders auflösen: "https (443)" und erfährt somit, dass das Paket über eine HTTPS-Internetverbindung versandt wurde. Wäre der Inhalt der Nachricht zu groß gewesen (bei TCP/IP maximal 1460 Bytes), wäre das Paket in mehrere Segmente aufgeteilt worden. Dies wird folgend in Abschnitt 6.4.3 besprochen. In diesem Fall finden sich Informationen zu den anderen Segmenten auf dieser Ebene.

Anwendungsdaten

Zuletzt befinden sich die eigentlichen Daten auf der Ebene der Anwendungsschicht. Im Fall von WhatsApp wurden die Daten mittels TLS Version 1.2 übertragen (TLSv1.2). Somit sind die Anwenderdaten verschlüsselt und vor einem MitM-Angriff geschützt. Es ist also kein Zugriff auf die Metadaten, welche WhatsApp zusätzlich zur verschlüsselten Nachricht verschickt, sowie den tatsächlichen Inhalt der Nachricht des Nutzers möglich. Dies ist auch nicht das Ziel dieser Arbeit, genau so wenig wie das eventuelle Brechen oder Umgehen der Verschlüsselung – sowohl von TLS, als auch des Signal-Protokolls von WhatsApp.

Dazu ist anzumerken, dass sich die aufgezeichneten Daten nach mehrmaligen Testdurchläufen und sogar über einen längeren Zeitraum mit Tests an unterschiedlichen Tagen, nicht geändert haben. Lediglich einzelne Paketgrößen, Port- und Ack-Nummern, sowie die Zeit des Eintreffens des 2. Datenpakets (des TCP Paketes) schwankte, was auf unterschiedliche Netzwerk- und Prozessorauslastung, sowie andere äußere Einflüsse zurückzuführen ist. Dies kann im Anhang unter NetCap 1 - 4 nachvollzogen werden.

```
No. Time        Source        Destination Protocol
  1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
Length  Info
271     Application Data
Frame 1: 271 bytes on wire (2168 bits), 271 bytes
      captured (2168 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02,
             Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53,
                              Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443,
      Dst Port: 40588, Seq: 1, Ack: 1, Len: 217
Transport Layer Security

No. Time        Source    Destination   Protocol
  2 0.000034184 10.0.2.15 185.60.217.53 TCP
Length Info
54     40588 → 443 [ACK] Seq=1 Ack=218
       Win=65535 Len=0
Frame 2: 54 bytes on wire (432 bits), 54 bytes
      captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e,
             Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15,
                              Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 40588,
      Dst Port: 443, Seq: 1, Ack: 218, Len: 0
```

NetCap 1: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy nach Ubuntu 01 (vom 15.11.2020)

```
No. Time        Source        Destination Protocol
  1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
Length Info
270    Application Data
Frame 1: 270 bytes on wire (2160 bits), 270 bytes
      captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02,
             Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53,
                              Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443,
      Dst Port: 37560, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No. Time        Sourcee   Destination   Protocol
  2 0.000039159 10.0.2.15 185.60.217.53 TCP
Length Info
54     37560 → 443 [ACK] Seq=1 Ack=217
       Win=63900 Len=0
Frame 2: 54 bytes on wire (432 bits), 54 bytes
      captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e,
             Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15,
                              Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 37560,
      Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 3: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy nach Ubuntu 03 (vom 30.11.2020)

Bei der Erarbeitung wurden noch wesentlich mehr Tests durchgeführt. Diese führten jedoch alle zum selben Ergebnis, weshalb aus Gründen der Übersichtlichkeit in dieser Arbeit hier lediglich vier Testdurchläufe dokumentiert wurden. Diese befinden sich im Anhang und weitere auf der mitgelieferten CD.

Als wichtiger Punkt sei noch anzumerken, dass sich die IP-Adresse des Servers von WhatsApp (Source-IP), zudem sich der Client des Nutzers im Test verbindet, selbst über einen längeren Zeitraum nicht ändert. Somit ist davon auszugehen, dass sämtlicher Datenverkehr von Nutzern stets von jeweils einem einzigen zentralen Server von WhatsApp verarbeitet wird, abhängig vom Standort. Dieser hat nicht nur Zugriff auf die in dieser Arbeit aufgezeigten Daten, sondern auch auf die unverschlüsselten Applikationsdaten der übermittelten Datenpakete. Was dies für die Sicherheit der Daten der Nutzer, vor allem im Bezug auf deren Metadaten bedeutet, wird später genauer in dieser Arbeit diskutiert.

Folgend werden verschiedene Testszenarien betrachtet, wobei geprüft werden soll, ob eine Änderung der Testumgebung oder des Testaufbaus eine Änderung in den versandten Daten zur Folge hat. Als Referenz wird hierzu das zuvor besprochenen Datenpaket NetCap 1 verwendet.

6.4.1. Verwenden eines anderen Handys als Absender

In diesem Testdurchlauf wurde zum mobilen Versenden der Nachrichten nicht wie zuvor das Testsystem C (5.1.3 - Umidigi F2), sondern das Testsystem D (5.1.4 - Umidigi F1), also die Vorgängerversion des Umidigi F2 verwendet. Da die beiden Mobiltelefone sich sehr ähnlich sind, sowohl von der Architektur, als auch vom verwendeten Betriebssystem Android (wobei das F2 Android 10 verwendet und das F1 nur die ältere Version Android 9, siehe Anhang 28, 29), würden feststellbare Änderungen in den Metadaten darauf rückschließen lassen, dass selbst diese sehr ähnlichen Systeme anhand der Metadaten unterscheidbar wären. Man könnte in diesem Fall also davon ausgehen, dass anhand der Metadaten Rückschlüsse auf das verwendete System des Absenders möglich sind, zumindest durch den Server.

Ansonsten blieb der Versuchsaufbau gleich. Die Nachrichten wurden an das Testsystem B (5.1.2) gesendet, welches WhatsApp Web im Firefox-Browser verwendete. Folgend werden die Ergebnisse der Tests dargestellt und erläutert. Links ist als Referenz der Datenverkehr durch Versenden der Nachrichten vom F2, rechts der vom F1 dargestellt.

```
No. Time        Source        Destination Protocol
  1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
Length Info
271    Application Data
Frame 1: 271 bytes on wire (2168 bits), 271 bytes
      captured (2168 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02,
             Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53,
                              Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443,
      Dst Port: 40588, Seq: 1, Ack: 1, Len: 217
Transport Layer Security
    TLSv1.2 Record Layer: Application Data Protocol:
          http-over-tls
        Content Type: Application Data (23)
        Version: TLS 1.2 (0x0303)
        Length: 212
        Encrypted Application Data: 7fd797366f597...
        [Application Data Protocol: http-over-tls]

No.  Time         Source     Destination    Protocol
  2  0.000034184  10.0.2.15  185.60.217.53  TCP
Length Info
54     40588 → 443 [ACK] Seq=1 Ack=218
       Win=65535 Len=0
Frame 2: 54 bytes on wire (432 bits), 54 bytes
      captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e,
             Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15,
                              Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 40588,
      Dst Port: 443, Seq: 1, Ack: 218, Len: 0
```

NetCap 1: Netzwerkverkehr WhatsApp Web - Nachrichten: F2 nach Ubuntu 01 (vom 15.11.2020)

```
No. Time        Source        Destination Protocol
  1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
Length Info
270    Application Data
Frame 1: 270 bytes on wire (2160 bits), 270 bytes
      captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02,
             Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53,
                              Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443,
      Dst Port: 50510, Seq: 1, Ack: 1, Len: 216
Transport Layer Security
    TLSv1.2 Record Layer: Application Data Protocol:
          http-over-tls
        Content Type: Application Data (23)
        Version: TLS 1.2 (0x0303)
        Length: 211
        Encrypted Application Data: 4863f48b7dd12...
        [Application Data Protocol: http-over-tls]

No. Time        Source     Destination    Protocol
  2 0.000035486 10.0.2.15  185.60.217.53  TCP
Length Info
54     50510 → 443 [ACK] Seq=1 Ack=217
       Win=65535 Len=0
Frame 2: 54 bytes on wire (432 bits), 54 bytes
      captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e,
             Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15,
                              Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 50510,
      Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 5: Netzwerkverkehr WhatsApp Web - Nachrichten: F1 nach Ubuntu 01 (vom 23.11.2020)

Dabei wurde bei dieser Darstellung zusätzlich die Informationen der "Transport Layer Security" mit ausgegeben. Diese finden sich nicht im Anhang zu den entsprechenden Dateien (NetCap 1 und NetCap 5). Die kompletten aufgezeichneten Daten sind auf der mitgegebenen CD unter *"Bachelorarbeit\Wireshark Capture\WhatsApp Web\…"*.

Erstaunlicherweise können tatsächlich Unterschiede in den aufgezeichneten Daten festgestellt werden, welche sich lediglich durch das zum Versenden der Nachrichten verwendete Mobiltelefon unterscheiden. Wesentlich ist hierbei die unterschiedliche Länge der empfangenen TLS-Pakete in Zeile 5. Während bei den vom F2 versendeten Daten (Anhang 1 - 4) die Paketgröße meist 271 Bytes beträgt, weisen die vom F1 versendeten Pakete (Anhang 5 - 8) eine Größe von 270 Bytes auf. Diese ist zurückzuführen auf die Größe der versendeten Applikationsdaten in Zeile 16. Diese Größe ist konstant für alle vom F1 versandten aufgezeichneten Pakete der Nachrichtenübertragung und es konnten entgegen den Ergebnissen im vorherigen Test keine Unterschiede in der Größe der Pakete festgestellt werden. Dies bedeutet jedoch nicht, dass die Paketgröße nicht doch für unterschiedliche Sitzungen variieren kann. Die hierbei gewonnenen Erkenntnisse führen zu zwei Folgerungen für die Betrachtung von nicht vertrauenswürdigen Serverbetreibern (in dem Fall WhatsApp) und Angriffen durch MitM, analog zu Abschnitt 2.4:

1) Der von WhatsApp betriebene Server kann zwar nicht den Inhalt der versendeten Nachricht lesen, die Applikationsdaten (engl. application data) werden bei der TLS-Verbindung jedoch entschlüsselt und sind somit vom Server lesbar. Zwar ist es in dieser Arbeit nicht möglich die Applikationsdaten zu entschlüsseln und nachzuvollziehen, welche Daten an WhatsApp gesendet werden, dennoch lässt die unterschiedliche Größe der Datenpakete darauf schließen, dass unterschiedliche Informationen versandt werden. Ob es sich dabei um die Informationen der vom Absender verwendeten Hardware handelt, oder anderer, lässt sich an dieser Stelle nicht nachvollziehen.

2) Ein MitM-Angreifer kann in diesem Testszenario die unterschiedliche Länge der Pakete feststellen und dadurch Ebenfalls Rückschlüsse ziehen, ohne die Verschlüsselung zu brechen. Zumindest kann er den Datenverkehr unterscheiden zwischen versandten Nachrichten und sonstigen Daten. Dies ist in allen Testszenarien möglich.

Beides lässt keine eindeutigen Schlüsse auf Informationen zum verwendeten Gerät zu, oder zumindest konnten diese hiermit nicht nachgewiesen werden. Ein Unterschied in den empfangenen Paketen konnte trotzdem erfolgreich festgestellt werden, welcher möglicherweise bei mehr vorliegenden Informationen zu stärkeren Rückschlüssen führt.

6.4.2. Verwenden eines anderen Browsers (Chrome)

Bislang wurde auf dem Testsystem B (5.1.2) der Webbrowser *Firefox* verwendet. Im folgenden Testdurchlauf soll nun ein anderer Browser verwendet werden, bei ansonsten gleichbleibender Testumgebung. Ziel ist es heraus zu finden, ob Unterschiede in den verwendeten Browsern zu Unterschieden in den aufgezeichneten Daten führen.

WhastApp Web ist auf sämtlichen verbreiteten Browsern lauffähig. Für welchen sich der Nutzer entscheidet, hängt von dessen Präferenzen ab und soll hier nicht weiter diskutiert werden. Als Alternative zu Firefox setzte sich in den vergangenen Jahren Google Chrome durch. Dieser Browser wird hier im Testdurchlauf in der Version 87.0.4280.66 verwendet.

Folgend werden die aufgezeichneten Testergebnisse erläutert. Erneut änderten sich im Vergleich zwischen den einzelnen Testdurchläufen lediglich die Portnummern und Zeiten des Eintreffens des 2. Datenpakets. Dies kann im Anhang unter NetCap 9 - 12 nachvollzogen werden. Als Referenz für diesen Test dient NetCap 9.

```
1  No. Time        Source        Destination Protocol
2    1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
3  Length Info
4  271    Application Data
5  Frame 1: 271 bytes on wire (2168 bits), 271 bytes
         captured (2168 bits) on interface enp0s3, id 0
6  Ethernet II, Src: 52:54:00:12:35:02,
7               Dst: 08:00:27:4b:aa:7e
8  Internet Protocol Version 4, Src: 185.60.217.53,
9                               Dst: 10.0.2.15
10 Transmission Control Protocol, Src Port: 443,
11       Dst Port: 40588, Seq: 1, Ack: 1, Len: 217
12 Transport Layer Security
13     TLSv1.2 Record Layer: Application Data Protocol:
             http-over-tls
14         Content Type: Application Data (23)
15         Version: TLS 1.2 (0x0303)
16         Length: 212
17         Encrypted Application Data: 7fd797366f597...
18         [Application Data Protocol: http-over-tls]

20 No.  Time         Source     Destination    Protocol
21   2  0.000034184  10.0.2.15  185.60.217.53  TCP
22 Length Info
23 54     40588 → 443 [ACK] Seq=1 Ack=218
24        Win=65535 Len=0
25 Frame 2: 54 bytes on wire (432 bits), 54 bytes
         captured (432 bits) on interface enp0s3, id 0
26 Ethernet II, Src: 08:00:27:4b:aa:7e,
27              Dst: 52:54:00:12:35:02
28 Internet Protocol Version 4, Src: 10.0.2.15,
29                              Dst: 185.60.217.53
30 Transmission Control Protocol, Src Port: 40588,
31       Dst Port: 443, Seq: 1, Ack: 218, Len: 0
```

NetCap 1: Netzwerkverkehr WhatsApp Web - Nachrichten: F2 nach Ubuntu Firefox 01 (vom 15.11.2020)

```
1  No. Time        Source        Destination Protocol
2    1 0.000000000 185.60.217.53 10.0.2.15   TLSv1.2
3  Length Info
4  262    Application Data
5  Frame 1: 262 bytes on wire (2096 bits), 262 bytes
         captured (2096 bits) on interface enp0s3, id 0
6  Ethernet II, Src: 52:54:00:12:35:02,
7               Dst: 08:00:27:4b:aa:7e
8  Internet Protocol Version 4, Src: 185.60.217.53,
9                               Dst: 10.0.2.15
10 Transmission Control Protocol, Src Port: 443,
11       Dst Port: 41342, Seq: 1, Ack: 1, Len: 208
12 Transport Layer Security
13     TLSv1.2 Record Layer: Application Data Protocol:
             http-over-tls
14         Content Type: Application Data (23)
15         Version: TLS 1.2 (0x0303)
16         Length: 203
17         Encrypted Application Data: d2cac67e97c56...
18         [Application Data Protocol: http-over-tls]

20 No. Time        Source     Destination   Protocol
21   2 0.000064072 10.0.2.15  185.60.217.53 TCP
22 Length Info
23 54     41342 → 443 [ACK] Seq=1 Ack=209
24        Win=65535 Len=0
25 Frame 2: 54 bytes on wire (432 bits), 54 bytes
         captured (432 bits) on interface enp0s3, id 0
26 Ethernet II, Src: 08:00:27:4b:aa:7e,
27              Dst: 52:54:00:12:35:02
28 Internet Protocol Version 4, Src: 10.0.2.15,
29                              Dst: 185.60.217.53
30 Transmission Control Protocol, Src Port: 41342,
31       Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 9: Netzwerkverkehr WhatsApp Web - Nachrichten: F2 nach Ubuntu Chrome 01 (vom 29.11.2020)

Wie im vorherigen Test zählt zu den wesentlichen Änderungen die Länge des übertragenen TLS-Pakets in Zeile 4. Diese ist erneut auf eine Änderung in der Länge der verschlüsselten Applikationsdaten zurückzuführen. Dabei weisen alle aufgezeichneten Datenpakete der Netzwerkübertragung eine Länge von 262 Byte auf. Erneut ist nicht ausgeschlossen, dass diese Länge nicht variieren kann und dieser Fall nur nicht in den Tests auftrat. Es können analog dieselben Schlüsse gezogen werden: aufgrund der unterschiedlichen Länge der Applikationsdaten, bei ansonsten gleichbleibender Testumgebung, kann dies darauf hindeuten, dass das Verwenden eines anderen Browsers sich in veränderten Applikationsdaten widerspiegelt und damit für jene erkennbar ist, welche Zugriff auf die unverschlüsselten Daten haben. Aufgrund limitierender technischer Voraussetzungen, gibt es in dieser Arbeit keinen Zugriff auf diese verschlüsselten Daten und dies kann somit nicht eindeutig beweisen werden. Analog könnte durch MitM-Angreifer durch die Länge der Pakete Informationen über den verwendeten Browser oder die Art der versandten Daten geschlossen werden.

6.4.3. Inhalt der versandten Textnachrichten

Bislang wurden willkürlich gewählte Nachrichten mit einer durchschnittlichen Länge von 5-6 Zeichen versandt. Nun soll durch gezieltes Ändern der Textnachrichten überprüft werden, ob allein durch das betrachten der Metadaten der Paketübertragung Rückschlüsse auf den Inhalt der Nachrichten möglich sind. Dabei werden sowohl verschiedene Zeichen verwendet, als auch die Anzahl der Zeichen und damit die Länge der zu verschlüsselnden Nachricht variiert. Zur Übersicht der Arbeit finden sich die aufgezeichneten Pakete nicht im Anhang, dafür aber auf der mitgelieferten CD unter *"Bachelorarbeit\Wireshark Capture\WhatsApp Web\Nachrichtenlaenge\…"*. Die Ergebnisse des Tests werden folgend in Tabellen 1 und 2 zusammengetragen, sowie anschließend besprochen. Der Testaufbau entspricht dem vom Referenztest (6.4.1), nur mit nicht-willkürlich gesendeten Nachrichten, welche ebenfalls aufgeführt werden.

Änderung der Länge der Nachrichten

Die Länge der entsprechenden Nachricht ist in der Anzahl der Zeichen, die Länge der Pakete und Applikationsdaten in Bytes angegeben.

Nachricht	Länge Nachricht (Anzahl Zeichen)	Länge Paket (Bytes)	Länge Applikationsdaten (Bytes)
a	1	255	196
aa	2	255	196
aaa	3	255	196
aaaa	4	271	212
aaaaa	5	271	212
⋮	⋮	⋮	⋮
a…a	18	271	212
a…a	19	271	212
a…a	20	287	228
a…a	21	287	228
⋮	⋮	⋮	⋮
a…a	34	287	228
a…a	35	287	228
a…a	36	303	244
a…a	37	303	244
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
a…a	1183	1439	1380
a…a	1184	1446 + 63	1396
⋮	⋮	⋮	⋮

Tabelle 1: WhatsApp Web Nachrichtenlänge - Handy (F2) nach Ubuntu

Als wesentlicher Unterschied konnte die Länge der Pakete festgestellt werden, welche auf die Länge der verschlüsselten Applikationsdaten zurückzuführen ist. Tabelle 1 zeigt, dass die Länge der versandten Nachricht direkt Einfluss auf die Länge der Applikationsdaten und damit der Pakete hat. Eben jene Nachrichten werden verschlüsselt und versandt und haben daher direkt Einfluss auf die Paketgröße. Dennoch kann man so durch die Größe der Pakete nicht nur einschätzen, ob es sich bei den Applikationsdaten um eine versandte Nachricht, oder andere Applikationsdaten wie Statusupdates oder sonstiges handelt, sondern auch die Größe jener versandten Nachricht abgrenzen. Dabei wurden erneut unterschiedliche Größen bei gleichen versandten Nachrichten festgestellt. Diese blieben dann aber für die jeweilige Sitzung konstant und es konnten weitere Erkenntnisse gewonnen werden. In dem aufgezeichneten Testdurchlauf beträgt die Paketgröße für eine Zeichenkette der zuvor verwendeten Größe von 5-6 Zeichen immer noch 271 Byte. Es wurde beobachtet, dass für Nachrichten von 1-3 Zeichen die Paketgröße 255 Bytes und damit 16 Bytes weniger beträgt. Ab einer Länge von 20 bis 35 Zeichen beträgt die Größe 287 Bytes und damit 16 Bytes mehr. Der Anstieg der Größe von 16 Bytes konnte stets ab einer Erhöhung der Zeichenanzahl von 16 Zeichen festgestellt werden. Dies blieb auch für Pakete gleich, welche bei anderen Testdurchläufen bei gleicher Zeichenlänge andere Paketgrößen aufwiesen: alle 16 Zeichen erhöht sich die Paketgröße um 16 Bytes.

Ab einer Zeichenlänge von 1184 Zeichen werden die Pakete aufgeteilt. Hier genügt es nun nicht mehr die Paketgröße zu betrachten, da diese für beide Pakete eine gewisse Anzahl von Informationen enthalten muss (IP-Adressen,…), welche sich nun zusammen genommen doppeln und damit die Größe um mehr als 16 Bytes erhöhen. Die Applikationsdaten steigen jedoch immer noch um einen Wert von 16 Bytes an. Die Aufteilung des Paketes in mehrere hängt vermutlich mit der Maximalen Länge der TCP/IP Pakete von 1460 Bytes zusammen. Bei einer wie zuvor festgestellten Erhöhung der Paketgröße um 16 Byte, hätte das Paket nun eine Größe von 1455 (1439+16) Bytes. Da dieser Wert sehr nah an der Maximallänge liegt, wurde das Paket vorher aufgeteilt. Dies ist lediglich ein Randfall um zu schauen, ab wann die Applikationsdaten in mehrere Pakete aufgeteilt werden und wie diese sich in dem Fall verhalten. In der Praxis werden kaum einzelne Nachrichten mit einer Länge von über 1000 Zeichen versandt.

Die hier gewonnenen Erkenntnisse können hauptsächlich von MitM-Angreifern genutzt werden, um mittels der Metadaten Informationen über die aufgezeichneten Pakete zu erhalten und die Länge der versandten Nachrichten einzugrenzen. Zudem kann durch die Art der aufgezeichneten Pakete in Verbindung mit der Größe dieser auf den Inhalt der Übertragung zurückgeschlossen werden.

Änderung der Zeichen der Nachrichten

Nun sollte abschließend ermittelt werden, ob eine Änderung der Zeichen – und damit konkreten Inhalts – bei gleichbleibender Länge Auswirkungen auf die Metadaten der versandten Nachricht hat. Dies hätte drastische Folgen, da somit nun nicht nur Informationen über die Metadaten der versandten Nachricht ermittelbar wären, sondern auch über deren konkreten Inhalt und somit die Verschlüsselung umgehbar wäre.

Nachricht	Länge Nachricht (Anzahl Zeichen)	Länge Paket (Bytes)	Länge Applikationsdaten (Bytes)
aaa	3	255	196
⋮	⋮	⋮	⋮
zzz	3	255	196
aaaa	4	271	212
⋮	⋮	⋮	⋮
zzzz	4	271	212

Tabelle 2: WhatsApp Web verschiedene Nachrichten - Handy (F2) nach Ubuntu

In allen durchgeführten Testdurchläufen konnte keine Information über den Inhalt der Nachrichten mittels der Metadaten gewonnen werden. Tabelle 2 fasst die zu den entsprechenden versandten Paketen gewonnenen Informationen zusammen. Dieses Ergebnis ist ebenso intuitiv: wäre es möglich, allein durch die Metadaten Informationen über den konkreten Inhalt der Applikationsdaten zu bekommen, so wäre die verwendete Verschlüsselung nicht sicher. Der Test bestätigt, dass anhand der Länge der verschlüsselten Nachrichten allein keine Informationen über diese gewonnen werden können, lässt dabei aber keine Schlussfolgerung über die tatsächliche Sicherheit der Verschlüsselung zu. Dennoch war es interessant, dies praktisch zu überprüfen und die Sicherheit der Verschlüsselung konnte zumindest durch den Aspekt der Metadaten nicht widerlegt werden.

6.5. Möglichkeiten zur Deanonymisierung von WhatsApp

Folgend sollen sämtliche Aspekte zusammengetragen werden, welche zur Deanonymisierung der Nutzer führen und damit die Privatsphäre (Abschnitt 2.2.6) dieser gefährden könnten. WhatsApp speichert auf ihren Servern eine folgend erläuterte Menge von Metadaten, welche bei der Kommunikation anfallen. Dazu gehören Telefonnummern, Zeitstempel, Dauer und Häufigkeit von Verbindungen, sowie Standorte von Nutzern [RH17]. Hinzu kommen "Name, Datum des Nutzungsbeginns, Datum des ‚Zuletzt online'-Zeitstempels, IP-Adresse(n) und E-Mail-Adresse(n)"[WaFAQ-St], sowie Daten über blockierte Nutzer, "Info"-Daten, Profilbilder, Informationen über Gruppen des Nutzers und Adressbücher [WaFAQ-St]. Die versandten Metadaten sind zwar bei der Übertragung verschlüsselt und somit auch für MitM-Angreifer nicht einsehbar, liegen aber WhatsApp vor und können im Rahmen einer Strafverfolgung den entsprechenden Behörden offengelegt werden. Dies sind mehr als genug Daten, um Nutzer zu deanonymisieren, ein Kontaktnetzwerk und Übersichten über deren Onlineaktivität aufzustellen und reichlich Informationen über diese zu erhalten, auch wenn die eigentlichen Inhalte der Nachrichten verschlüsselt sind. Somit muss man als aktiver Nutzer WhatsApp und deren Muttergesellschaft Facebook vertrauen, dass sie diese Daten nicht missbrauchen. Der Datenschutz der Nutzer durch das Unternehmen ist in WhatsApps Datenschutzrichtlinie festgehalten. Beim Registrieren stimmt man dieser in Form der Allgemeinen Geschäftsbedingung zu. Zugleich wird offengelegt, welche Daten über die Nutzer erhoben werden.

WhatsApps Datenschutzrichtlinie & Informationen über Nutzer

Unter dem Abschnitt "Informationen, die wir erheben" in der Datenschutzrichtlinie von WhatsApp werden sämtliche von WhatsApp angeführten Informationen aufgelistet, welche der Dienst über seine Nutzer sammelt. Diese werden in drei Kategorien eingeteilt[12]:

- **Vom Nutzer (manuell) bereitgestellte Informationen**
  - **Account-Informationen:** Telefonnummer, Name, Kontaktbuch, E-Mail, Profilbild, Info/Status
  - **Nachrichten:** verschlüsselte Chats, Fotos, Standort, usw., bei Nichtzustellung maximal 30 Tage auf dem Server gespeichert
  - **Netzwerk:** Kontakte in WhatsApp, Gruppeninformationen
  - **Nutzung der Zahlungsdienste:** Kauf- und Transaktionsinformationen
  - **Customer Support:** Nutzungs- und Kontaktinformationen, Nachrichtenkopien
- **Automatisch erfasste Informationen**
  - **Nutzungs- und Log-Informationen:** dienstspezifische- und Performance-Informationen, Aktivität und Einstellungen, Zeitpunkt, Häufigkeit und Dauer der Aktivitäten und Interaktionen, Log-Dateien sowie Diagnose-, Absturz-, Webseiten- und Performance-Logs und -Berichte, Registrierungszeitpunkt, vom Nutzer genutzte Funktionen und Onlinestatus
  - **Geräte- und Verbindungsdaten:** Hardware-Modell, Betriebssystem, Batteriestand, Signalstärke, App-Version, Browser und Mobilfunknetz sowie zu der Verbindung, einschließlich Telefonnummer, Mobilfunk- oder Internetanbieter, Sprache und Zeitzone sowie IP-Adresse, Gerätebetrieb und Kennungen wie Gerätekennungen
  - **Standort-Informationen:** IP, GPS, Bluetooth-Signale, WLAN-Zugangspunkte, Beacons und Funkzellentürme in der Nähe
  - **Cookies:** Analyse, wie die Dienste genutzt werden, Unterscheidung WhatsApp/-Desktop/-Web, Sprachpräferenzen, Individualisierung
- **Informationen Dritter über Nutzer**
  - **Von anderen Nutzern und Unternehmen:** Telefonnummer, Name
  - **Unternehmen auf WhatsApp:** Interaktionsinformationen, gesammelte Informationen durch zusammenarbeitende Drittunternehmen
  - **Drittanbieter und Facebook:** Standorte, Karten, Orte, Umfragen und Studien, App-Store Daten
  - **Dienste Dritter:** bei Interaktion mit Diensten Dritter oder Facebook, Teilen-Funktion, Nutzung von Werbeaktionen

An dieser Stelle wurde bewusst ausführlich auf die von WhatsApp gesammelten Daten eingegangen, um zu verdeutlichen, wie umfangreich jene gesammelte Datenmenge über den Nutzer ist.

[12] `https://www.whatsapp.com/legal/privacy-policy-eea` (abgerufen am 4. September 2022)

Des Weiteren ist nicht konkret bekannt, wie diese Daten praktisch im System von WhatsApp verwendet werden, welche Informationen aus ihnen berechnet wird oder ob dies die komplette Menge der erhobenen Daten ist. Sowohl der Quellcode von der Applikation WhatsApp selbst, als auch der Server ist nicht offen und kann daher nicht überprüft werden [ORP20]. Erneut muss der aktive Nutzer dem Unternehmen vertrauen, dass diese Daten nicht auf andere Weise verwendet werden. Zumindest ist das potentielle Risiko für den Datenschutz der Nutzer bereits für die zuvor angegebene Datenmenge enorm.

WhatsApp verdient aktuell kein Geld mit dem Verkauf der Daten seiner Nutzer an Dritte, dafür werden Informationen über deren Nutzer mit Facebook-Unternehmen geteilt. Als Gründe werden angeführt, "um von Leistungen in den Bereichen Infrastruktur, Technologie und Systeme profitieren zu können"[WaFAQ-FB]. Zu den geteilten Informationen gehören die Telefonnummer, "Geräteinformationen (Gerätekennung, Betriebssystemversion, App-Version, Plattforminformation, Ländervorwahl der Mobilnummer, Netzwerkcode sowie Markierungen, die es erlauben, deine Zustimmung zu Aktualisierungen und Steuerungsoptionen nachzuverfolgen) und einige deiner Nutzungsinformationen (wann du WhatsApp zum letzten Mal genutzt hast, wann du deinen Account registriert hast, sowie die Art und Häufigkeit deiner Nutzung von Features)"[WaFAQ-FB]. WhatsApp verkauft keine Daten ihrer Nutzer, gibt diese aber an Facebook weiter, welche mittels personalisierter Werbung die Daten ihrer Nutzer an Werbefirmen verkaufen.

Abschließend sei angemerkt, dass zumindest die übertragenen Metadaten gegenüber Angreifern geschützt sind. Die Kommunikation zwischen Clients und Servern und insbesondere deren Metadaten ist zusätzlich durch das *Noise Protocol*[13] abgesichert [Wha16, S. 9]. Zudem erschwert dies zusätzlich einen MitM-Angriff, zumindest für das Überwinden der Verschlüsselung. Ein Angreifer kann sämtliche zuvor beschriebenen Daten erlangen, ohne die Verschlüsselung zu umgehen und wird eine hinreichend große Menge an Knoten der Kommunikation überwacht, können Nutzer ihrer Kommunikation zugeordnet werden.

6.6. Erkenntnisse

Der Inhalt der Kommunikation zwischen Nutzern ist durch WhatsApp und dessen verwendete Ende-zu-Ende-Verschlüsselung zum derzeitigen Stand sicher. Dies gilt jedoch nicht für alle Metadaten der Nutzer. Eine große Menge dieser wird durch WhatsApp selbst erhoben. Nach außen geschützt entsteht allein schon aufgrund der Menge der Daten ein großes Potential zur Deanonymisierung der Nutzer durch WhatsApp und dessen Mutterunternehmen Facebook. Die restlichen Metadaten, welche durch die Kommunikation anfallen, sind in keiner Weise geschützt. Angreifer können IP-Adressen, Zeiten, Art und sogar abgegrenzt die Größe der Nachrichten bestimmen. Es fehlen Maßnahmen zur Verschleierung aller Metadaten und solange dies der Fall ist, können Nutzer von WhatsApp potentiell deanonymisiert werden.

[13] `https://noiseprotocol.org/noise.pdf` (abgerufen am 4. September 2022)

7. Matrix

Matrix[14] ist ein Open-Source-Projekt, welches die Möglichkeit bietet, dezentralisiert Nachrichten zu versenden und übertragen. Dafür können eigens für Matrix entwickelte Messenger wie Elements, aber auch herkömmliche Messenger und Nachrichtendienste wie WhatsApp, Telegram und Signal verwendet werden, sowie E-Mail, SMS und IP- oder Video-Telefonie. Ziel ist, all diese für sich getrennten Lösungen der Kommunikation miteinander zu verbinden und dabei Sicherheit und dezentrale, anonyme Kommunikation umzusetzen. Matrix wird aktuell zudem von den Behörden Frankreichs, sowie der deutschen Bundeswehr genutzt [Gru20].

7.1. Überblick und Funktion von Matrix

Das Protokoll Matrix hat einige Ansätze, wodurch sich deren Messenger von der klassischen Funktionalität von Messengern unterscheiden. Es setzt auf der Applikationsschicht des ISO/OSI-Referenzmodells (Abschnitt 2.3.1) auf und überträgt JSON-Dateien zwischen Clients und Diensten. Daten werden versandt, indem sie in einen *room* auf deren Server gesendet werden, welcher die Daten dann an alle anderen Server verteilt, welche an dem *room* teilnehmen [MSpec]. Das Grundkonzept von Matrix entspricht dem der E-Mail [Wei14]: Nutzer können sich gegenseitig Nachrichten senden, unabhängig von der verwendeten Plattform. Dies soll nun auch für die Vielzahl von Messengern möglich werden durch Matrix. Die wichtigsten Aspekte der Kommunikation werden folgend erläutert.

7.1.1. Dezentrale Kommunikation & Server

Im Gegensatz zu herkömmlichen, zentralisierten Messengern wie WhatsApp, können Nutzer von Matrix ihre eigenen Heimserver erstellen. Als Referenzsysstem dient dafür *Synapse*[MSyn]. Auf dem Server werden anschließend die Informationen über Nutzer und deren Chatverläufe gespeichert. Diese Daten werden mit dem Matrix-Ökosystem geteilt, indem die Chatverläufe mit anderen Homeservern und deren Clients synchronisiert werden [MSpec]. Die ”room”-Daten der Nutzer sind dupliziert auf all derer Homeserver, wodurch kein einzelner Server volle Kontrolle über diese Daten hat. Sollte dieser ausfallen, kann die Kommunikation zwischen den anderen Teilnehmern dennoch stattfinden.

Im Rahmen dieser Arbeit war geplant, einen eigenen Matrix-Homeserver aufzusetzen. Dadurch sollte zusätzlich überprüft werden, wie viele Daten der Server über die Nutzer sammeln kann, trotz sicherer Ende-zu-Ende-Verschlüsselung. So ist es zwar gelungen einen eigenen Matrix-Server aufzusetzen, dieser konnte jedoch nicht mit anderen Messengern wie WhatsApp oder Matrix-Element verbunden werden. In der vorgegebenen Zeit konnte dieses Problem nicht gelöst werden, weshalb die Überprüfung der einsehbaren Metadaten durch Matrix-Server als Ausblick beschrieben wird.

[14] `https://matrix.org/` (abgrufen am 4. September 2022)

7.1.2. Bridges

Eine weitere Besonderheit und gleichzeitig Hauptbestandteil von Matrix ist Interoperabilität; die Möglichkeit Nachrichten zwischen verschiedensten Messengern, sowie E-Mail und SMS zu versenden. Dies wird technisch realisiert mittels so genannten *Bridges* (dt. "Brücken"). Die unterschiedlichen Arten von Brücken regeln die Kommunikation innerhalb und außerhalb des Matrix-Ökosystems, botbasiert oder automatisch über authentifizierte Accounts auf anderen Plattformen[15].

7.2. Installation von Matrix

Es gibt verschiedenste Möglichkeiten mittels Matrix zu kommunizieren. Die verbreitetste ist aktuell der Messenger Element, welcher in dieser Arbeit betrachtet wird. Die Registrierung erfolgt dabei ohne persönliche Authentifikation. Die Angabe einer Telefonnummer oder Mailadresse ist optional und es genügt lediglich ein für den jeweiligen Server einzigartiger Nutzername (auch Matrix-ID) in Kombination mit einem Passwort. Hinzu kommt die Auswahl, auf welchem Server sich der Nutzer registrieren möchte. Standardmäßig ist das der "Matrix.org"-Server. Die Matrix-ID ist nach dem Schema "*@localpart:domain*" [MSpec] aufgebaut, unter welcher ein Nutzer erreichbar ist.

Matrix (Element) mobil im App-Store

Die App-Variante von Matrix kann für Android im Google Play Store unter `https://play.google.com/store/apps/details?id=im.vector.app` (abgerufen am 4. September 2022) und für Apple/iOS-Geräte in Apples App Store unter `https://apps.apple.com/de/app/riot-im/id1083446067` (abgerufen am 4. September 2022) heruntergeladen und installiert werden. Zudem gibt es eine Google-unabhängige Version: `https://f-droid.org/en/packages/im.vector.app/` (abgerufen am 4. September 2022).

Matrix (Element) als Desktop-Applikation

Für Windows und MacOS steht ein Installer unter dem jeweiligen Downloadlink der Webseite von Element `https://element.io/get-started` (abgerufen am 4. September 2022) zum Download bereit. Zudem ist dort die Installation für Debian/Ubuntu (64-bit) beschrieben. Nach der Installation der benötigten Pakete kann man Element mittels des folgenden Befehls installieren (Code 2):

```
$ sudo apt install element-desktop
```

Code 2: Installation Element Debian/Ubuntu

Matrix (Element) im Browser

Im Browser ist Element unter `https://app.element.io` (abgerufen am 4. September 2022) erreichbar für alle gängigen Browser wie Chrome, Safari oder Firefox.

[15] `https://matrix.org/bridges/` (abgerufen am 4. September 2022)

7.3. Sicherheit von Matrix

In diesem Abschnitt wird die Sicherheit von Matrix und dessen verwendeten Protokollen betrachtet. Matrix E2E-Verschlüsselung basiert auf *Olm*- und *Megolm cryptographic Ratchet* [ME2E]. Umgesetzt sind diese in Matrix durch die in C/C++ geschriebene *libolm*-Bibliothek[16]. Olm implementiert den doppelten kryptografischen Ratchet des Signal-Protokolls, zuvor beschrieben in Abschnitt 4.3.4. Der technische Mitbegründer von Matrix.org Matthew Hodgson schrieb 2020 in einem Blogartikel zur Intention der Benutzung des Signal-Protokolls: "one of the reasons for us [...] was to increase our chances of one day connecting with other apps using the same algorithm" [Hod20].

7.3.1. Olm & Megolm: E2E-Verschlüsselung

Die *Olm*-Bibliothek besteht aus zwei wesentlichen Komponenten: der End-to-End (E2E) -Verschlüsselung mit Ratchet durch *Olm* und der Gruppen-Ratchet-Mechanismus *Megolm*. Da Olm die Verschlüsselung des Signal-Protokolls umsetzt, gelten dessen kryptografische Eigenschaften analog. In 2016, vor der Beta-Veröffentlichung der End-to-End-Verschlüsselung von Matrix, fand eine sicherheitstechnische Analyse der NCC Group einige Sicherheitslücken in der *libolm*-Bibliothek. Dazu gehörte ein "Unknown Key-Share"-Angriff, mittels welchen die Authentizität von Kommunikationspartnern manipuliert werden konnte, sowie Bedenken zur PFS (Abschnitt 4.3.1) und PCS (Abschnitt 4.3.2) von Gruppennachrichten. Diese wurden nachträglich behoben [Hod16], wsodurch E2E-Verschlüsselung für Chats, Anhänge, Voice over IP (VoIP) und weitere Features umgesetzt werden konnten [Hod16].

Megolm ist ebenso AES basiert wie Olm, dafür aber mit einer Ausrichtung auf Gruppenkonversationen mit einer großen Menge an Teilnehmern. Empfangene Nachrichten können zudem mehrfach entschlüsselt werden, wodurch die verschlüsselte Nachricht auf einem nicht vertrauenswürdigen Server gespeichert werden kann, während der Nutzer den Session-Schlüssel speichert[17]. Jeder Nutzer baut eine eigene sichere Verbindung auf. Die Schlüssel werden über authentifizierte Kanäle versandt, wodurch die einzelne Verbindung zur Gruppenkonversation die kryptografischen Eigenschaften der einzelnen Verbindung erbt. PCS wird dadurch nicht und PFS nur teilweise für Gruppen umgesetzt.

7.3.2. Authentifizierung von Geräten

Ein neues Gerät muss beim erstmaligen Login des Nutzers auf diesem authentifiziert werden. Die Kommunikationspartner des Nutzers werden über die Verwendung eines neuen Gerätes informiert und bauen eine sichere Verbindung zu diesem auf. Dieser Prozess wird über Geräte (engl. *device*)-Schlüssel geregelt, wobei jeder Nutzer eine Liste der Geräte seiner Kommunikationspartner hat. Nun kann der neue Kanal des Nutzers mittels Verifikationscodes authentifiziert werden. Seit Mai 2020 erlaubt Element "Cross-signing" [Ele], wodurch Nutzer ihre Authentifizierung selbst durchführen können.

[16] `https://gitlab.matrix.org/matrix-org/olm` (abgerufen am 4. September 2022)
[17] `https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/megolm.md` (abgerufen am 4. September 2022)

7.4. Prüfen der Sicherheit der Metadaten von Matrix

Diese Tests bauen auf den Erkenntnissen der Testdurchläufe von WhatsApp (6.4) auf. Ziel ist es herauszufinden, wie viele Daten beim Mitlesen der Nachrichtenübertragung von Matrix (Element) erfasst werden, diese auszuwerten und im Vergleich zum aktuellen Marktführer WhatsApp im Bezug auf Sicherheit und Privatsphäre zu betrachten.

7.4.1. Referenztest Nachrichtenübertragung

Der folgende Abschnitt dient als Referenz für alle Tests der Nachrichtenübertragung von Matrix (Element). Ausgehend von diesen Szenario soll daraufhin betrachtet werden, wie und ob Änderungen in den Testumgebungen Einfluss auf die versandten Daten haben.

Im Test selbst wurden Nachrichten über Element vom Testsystem C (5.1.3) an die Element Desktop-Applikation auf dem Testsystem B (5.1.2) gesendet. Erneut lief auf System B Wireshark, um die Paketkommunikation aufzuzeichnen. Die vollständigen Mitschnitte befinden sich auf der mitgelieferten CD unter *"Bachelorarbeit\Wireshark Capture\Matrix-Element\…"*. Abbildung 15 (im Anhang NetCap 13) zeigt ein typischen Mitschnitt des Nachrichtenempfangs von Element.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.20.20.236	10.0.2.15	TLSv1.2	1093	Application Data
2	0.000377210	104.20.20.236	10.0.2.15	TLSv1.2	85	Application Data
3	0.001648749	10.0.2.15	104.20.20.236	TCP	54	47216 → 443 [ACK] Seq=1 Ack=1071 Win=63900 Len=0
4	0.052021946	10.0.2.15	104.20.20.236	TLSv1.2	211	Application Data
5	0.052366944	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1071 Ack=158 Win=65535 Len=0
6	0.104706885	104.20.20.236	10.0.2.15	TLSv1.2	242	Application Data
7	0.106381360	10.0.2.15	104.20.20.236	TLSv1.2	202	Application Data
8	0.106615086	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1259 Ack=306 Win=65535 Len=0

Abb. 15: Wireshark WhatsApp - Nachrichtenpakete

Beim Nachrichtenempfang sendet der Server zunächst ein TLS-Paket. Dieses ist stets wesentlich größer, als die weiteren aufgezeichneten Pakete, mit meist über 1000 Bytes. Dem folgt erneut vom Server ein TLS-Paket, mit einer konstanten Größe von 85 Bytes. Client und Server senden sich abwechselnd nun 3 mal TLS-Pakete hin und zurück, welche jeweils noch einmal mittels eines TLS-Pakets (ACK) bestätigt werden. Diese Anordnung der TLS-Pakete bleibt in den meisten Fällen gleich. Zeitweise erscheinen die Bestätigungen durch TCP-Pakete vor der Rückantwort, zu sehen in *"…\Matrix-Element\el_mobNew-to-ubuntuDesk_05"*, was jedoch nichts an der Abfolge der TCP- Pakete ändert. Die Größe aller Pakete, außer des ersten Pakets, ist in dem meisten Testdurchläufen konstant und entspricht der Größe der Pakete in NetCap 13. Eine Außnahme dafür ist z.B. *"…\el_mobNew-to-ubuntuDesk_03"*.

Damit kann die generelle Struktur von Paketen zur Nachrichtenübertragung für Element festgestellt werden. Diese kann von MitM-Angreifern genutzt werden, um Nachrichtenpakete zu identifizieren. Der Inhalt der Applikationsdaten – wobei das erste Paket die eigentliche verschlüsselte Nachricht enthält – kann nicht in Erfahrung gebracht werden. Dies ist jedoch auch nicht Ziel dieser Arbeit. Anders als bei WhatsApp (6.4.3) können keine Informationen über die Länge der Nachricht oder eine Eingrenzung dieser

vorgenommen werden. Die Größe des ersten Pakets schwankte sehr stark (für 5-6 Zeichen: 1200 Byte ±150 Byte) und es konnte kein Zusammenhang zwischen der Länge der versandten Nachrichten und der Größe der Pakete, welche von der Größe der Applikationsdaten abhängt, festgestellt werden. Auffällig ist zudem, dass zum Versenden gleich langer Nachrichten Matrix etwa 5× größere Applikationsdaten versendet, als WhatsApp. Zu den weiteren feststellbaren Daten über die versandte Nachricht gehören analog zu WhatsApp (6.4.1):

- **Größe:** Wie zuvor besprochen und stark schwankend.
- **Zeit:** Erneut ist die relative Zeit seit Eintreffen des ersten Datenpaketes angegeben. Auch dies ist eine Information, welche zu den Metadaten der Nachrichtenübertragung zählt.
- **MAC-Adresse:** Ist in allen durchgeführten Test für den Server konstant geblieben, auch wenn sich die IP-Adresse des Servers geändert hat.
- **IP-Adresse:** Diese änderte sich im Verlauf der durchgeführten Testdurchläufe für Server, beispielsweise zu sehen in *"...\el_mobNew-to-ubuntuDesk_06"*. Innerhalb einer Sitzung blieb die IP-Adresse konstant. Dennoch wurde dadurch gezeigt, dass für die Kommunikation mit Matrix verschiedene IP-Adressen verwendet werden, anders als bei WhatsApp.
- **Port:** Zuletzt erhält man Informationen über die verwendeten Ports. Der Server verwendete dafür stets den Port "443". Dieser wird für eine https-Internetverbindung genutzt.

Die Daten können bei jedem Mitschnitt einer Paketübertragung erhoben werden und bleiben wie zuvor beschrieben für sämtliche durchgeführten Testdurchläufe unter diesem Szenario gleich. Dadurch ergeben sich bereits zwei wesentliche Unterschiede im Vergleich zu WhatsApp:

- **Größe der Pakete/ Applikationsdaten:** lässt nicht auf die Größe der versendeten Nachrichten schließen, schwankt stark und ist bei gleicher Nachrichtenlänge etwa 5× größer, als derer von WhatsApp-Paketen.
- **IP des Servers:** ist je nach Sitzung unterschiedlich.

Beide Punkte sorgen dafür, dass Außenstehende (MitM-Angreifer) weniger Informationen über den Nutzer und dessen versandte Daten erhalten. Gerade der Punkt der wechselnden IP-Adressen des Servers ist besonders wichtig. Es genügt nicht nach nur einer einzigen Adresse zu filtern. Für die selbe Verbindung zwischen zwei Nutzern können die IP-Adressen auf der Verbindung zwischen diesen variieren. Ebenso kann aus Sicht eines nicht-vertrauenswürdigen Serverbetreibers, welcher zudem Zugang zu den verschlüsselten TLS-Daten hat, eine Kommunikation mehrere Pfade nehmen. Geht man davon aus, dass jede neue IP-Adresse der Server, zu welchem sich die Nutzer verbinden, ein anderer unabhängiger Server ist, so genügt es nicht die Kommunikation eines Nutzers auf einem Server zu überwachen, um ein vollständiges Bild und alle Informationen über dessen Kommunikation zu erhalten.

7.4.2. Verwenden eines anderen Handys als Absender

Folgender Testdurchlauf soll die Frage beantworten, ob bei der Verwendung eines anderen Mobiltelefones zum Versenden der Nachrichten Unterschiede in den aufgezeichneten Daten erkennbar sind. Konkret wurde dafür nicht wie im vorherigen Test das Umidigi F2 (Testsystem C (5.1.3)), zum Versenden von Nachrichten, sondern das F1 (Testsystem D (5.1.4)) verwendet. Ansonsten blieb der Testaufbau gleich. Der vollständige Test kann unter *"...\el_mobOld-to-ubuntuDesk_01"* - *"...04"* nachvollzogen werden, sowie im Anhang unter NetCap 14.

Es konnten keine Unterschiede von außen und nur anhand der aufgezeichneten Daten der Paketübertragung festgestellt werden. Ebenso können keine Rückschlüsse auf das verwendete System zum Versenden der Nachrichten getroffen werden. Zwar unterscheiden sich wie bei WhatsApp (6.4.1) die Größen der Nachrichten, da diese aber bereits im Referenztest (7.4.1) stark variierten, genügt diese Tatsache allein nicht, um Informationen herzuleiten. Der durchgeführte Test kann nicht ausschließen, dass die unverschlüsselten Applikationsdaten derartige Metadaten enthalten.

7.4.3. Verwenden eines Browsers (Chrome & Firefox)

Als nächstes soll überprüft werden, ob das Verwenden von Element über den Browser Änderungen in den Metadaten der Paketübertragung beim Versenden einer Nachricht verursacht. Dafür wird auf dem Testsystem B (5.1.2) der Browser Chrome verwendet und in diesem die Webseite von Element `https://app.element.io/` (abgerufen am 4. September 2022) aufgerufen. Die Ergebnisse des Test werden durch NetCap 15 zusammengefasst.

Erneut konnten keine Unterschiede in den mitgeschnittenen Paketen festgestellt werden. Die Änderung der Paketgröße des TLS-Pakets allein genügt nicht, um Aussagen über den verwendeten Browser des Empfängers zu tätigen. Es kann auch nicht gesagt werden, ob der Empfängern einen Browser, oder die Desktop-Applikation unter Ubuntu nutzt. Selbiges gilt für die Verwendung von Firefox als Browser (NetCap 16). Dies spricht für die Sicherheit der Metadaten über die Kommunikation via Matrix Element. Einziger nicht-trivialer Unterschied zum Referenztest ist die neue IP-Adresse des Servers. Diese entspricht auch der von *"...\el_mobNew-to-ubuntuDesk_06"* und wird somit auch für die Übertragung der Pakete der Desktop-Applikation verwendet. Dadurch lässt sie auch keine Rückschlüsse zu, ob ein und welcher Browser verwendet wurde.

Beide Tests konnten Aufgrund der Tatsache keine Beziehung zwischen Metadaten und dem verwendeten System oder Browser von außen nachweisen, dass sowohl Paketgrößen, als auch IP-Adressen von Servern für ein und das selbe Testszenario variieren. Dies verdeutlicht, dass bereits ein geringer zusätzlicher Overhead, welcher zur Varianz der Paketgröße beiträgt, genügt, um den Inhalt von Paketen zusätzlich zu verschleiern. In Kombination mit wechselnden IP-Adressen können so wesentlich weniger Metadaten der Paketübertragung für Außenstehende MitM-Angreifer sichtbar sein. Dies schützt die Privatsphäre der Nutzer jedoch nicht vor den Serverbetreibern selbst.

7.5. Möglichkeiten zur Deanonymisierung von Matrix

Folgend werden die über Nutzer gesammelten Daten zusammengetragen, welche bei der Verwendung des Matrix-Protokolls oder dessen kompatiblen Applikationen anfallen und zur Deanonymisierung derer Nutzer führen können. Matrix, deren Systeme und Server sind offen und in Verbindung mit den Datenschutzrichtlinien, sowie Informationen über die verwendete Architektur, erfährt man die Menge über die Nutzer anfallenden Daten.

7.5.1. Matrix & dessen Metadaten

Die grundlegende Funktionalität von Matrix sieht vor, dass die Kommunikation der Nutzer an sämtliche beteiligte Server gesendet wird. Trotz Verschlüsselung der Nachrichten erhalten somit sämtliche Server Zugang zu den Metadaten der Kommunikation. Obgleich wie in den Tests gezeigt verschiedene Server an der Kommunikation beteiligt sein können, so erhalten zumindest die Homeserver der Kommunikationspartner sämtlichen Netzwerkverkehr. Zwar bietet Matrix die Möglichkeit, die eigenen Nachrichten auf dem eigenen Homeserver zu löschen, es kann jedoch aufgrund der Architektur keine Garantie dafür geben, dass sämtliche Kopien der Nachrichten ebenfalls gelöscht werden. Zudem speichert der eigene Homeserver noch weitere Daten: Kontaktlisten, Raum- und Gruppenmitgliedschaften, persönliche Informationen (Matrix-ID, optionale Identifikatoren wie E-Mail oder Telefonnummer [Matd]) und alle Räume des Nutzers mit dessen (verschlüsselten) Nachrichten [Kuk20a]. Die Nutzer müssen dabei nicht nur ihrem Homeserver vertrauen, sondern auch dem der Nutzer, mit welchen sie kommunizieren. Das verwenden eines eigenen Homeservers ändert an dieser Tatsache deshalb nichts [Kuk20a]. Hinzu kommen Metadaten, welche bei der Kommunikation anfallen, selbst bei eigens gehosteten Heimservern [Dor19]: Nutzungsmuster des Benutzers, IP-Adressen, Geräte- und Systeminformationen, sowie korrespondierende Server und Raum-IDs. Unter Standardeinstellungen sind Matrix-IDs, welchen E-Mail-Adressen/Telefonnummern zugeordnet sind, in Verbindung mit den Einstellungen eines Benutzers, sowie Profilname und -bild des Nutzers einsehbar.

Der Beitrag von Dor, Notes on privacy and data collection of Matrix.org, 2019 [Dor19] kritisiert scharf die Privatsphäre von Matrix. Zu den zuvor genannten Problemen werden einige weitere Bedenken geäußert. Nach heutigem Stand sind viele gerade der grundlegenden Privatsphäreprobleme weiterhin ungelöst. Die Referenzimplementierungen von Matrix; *Riot* und *Synapse* resultieren derzeit in einem nahezu zentralisiert genutzten Netzwerk, was die Befürchtung von Signals Mitgründer Moxie Marlinspike [Mar16] bestätigt, auf welche am Ende im Abschnitt 7.6 ausführlicher eingegangen wird. Hinzu kommen Sicherheitslücken, welche in der Vergangenheit und aufgrund der unzureichenden Absicherung der Daten über Nutzer zu Privatsphäreverletzungen und Offenlegung von Identifikationsdaten dieser geführt haben [Matc],[Tre19]. Matrix ist sich vielen der Probleme bewusst [Hod19b] und arbeitet neben der Entwicklung selbst an der Lösung dieser Probleme. Zudem wurden einige der kritisierten Punkte aus [Dor19] von Matthew Hodgson angesprochen [Hod19a], wobei die widersprochenen Punkte nicht in dieser Arbeit angeführt wurden.

Ungeachtet der internen Datenschutzprobleme fallen von außen einsehbare Metadaten an. Dieses Problem hat seinen Ursprung in dem IP/TCP-Protokoll und betrifft Matrix ebenso wie WhatsApp und andere Messenger. Ein außenstehender MitM-Angreifer erhält Informationen über die übertragenen Pakete, ihre Art, die Zeiten des Eintreffens, Größe, IP-Adressen und damit an der Verbindung beteiligte Kommunikationspartner. Der Vorteil von Matrix gegenüber anderen Applikationen ist nun, dass die Nachrichten von Nutzern an deren Heimserver gesendet und von diesen verwaltet werden. Die Kommunikation zwischen diesen Heimservern ist dezentral und eine Überwachung dieser allein genügt nicht, um Rückschlüsse über die Kommunikation von Nutzern zu ziehen. Aufgrund der Föderalisierung müssen wesentlich mehr Server überwacht werden, um Informationen über Nutzer zu erhalten.

7.5.2. Identifikatoren & Identifikationsserver

Wie schon bevor besprochen (Abschnitt 7.2) nutzt Matrix zur Identifikation von Nutzern die sogenannte *Matrix-ID*, welche sich aus dem Alias des Nutzers und dessen Matrix-Server zusammensetzt. Diese genügt in Kombination mit einem Passwort, um sich als neuer Nutzer zu registrieren. Optional kann man einen *Identifikator*, meist Mail-Adresse oder Telefonnummer mit dem Account verknüpfen. Diese werden dann auf einem Identifikationsserver mit dem Nutzer verbunden, ähnlich der Funktionalität eines DNS-Servers, welcher folgende Funktionen bietet [Matb]: Verifikation des Identifikators, Herausfinden der Matrix-ID von Nutzern über deren Identifikator (aber nicht umgekehrt), Kontaktfindung und Registrierung für einige Server. Der Identifikationsserver ist von der restlichen Serverinfrastruktur des Matrix-Netzes unabhängig und dessen Nutzung ist rein optional. Das große Problem ist hierbei, dass der Identifikationsserver von Matrix (vector.im und matrix.org) zentral ist, welches enormes Potential für Privatsphäreverletzungen der Nutzer bietet. Der föderale Identifikationsserver *mxisd*[18] soll dieses Problem lösen. Das dezentrale System bietet die Möglichkeit, seinen eigenen Identitätsserver zu hosten, welcher zu den bestehenden Identitätsservern von Matrix verlinkt werden kann.

7.5.3. Datenschutz von Matrix-Brücken

Die Interoperabilität von Matrix ist Kernbestandteil der Funktionalität und konzipiert die Verbindung zu bestehenden Messengern, sowie SMS und Mail. Dies bietet zwar für Nutzer einen leichteren Umstieg auf Matrix, oder theoretisch auch die Möglichkeit der Kommunikation zwischen verschiedenen Messengern, resultiert jedoch je nach Anwendung in einem datenschutzrechtlichen Problem [Kuk20b]: unterschiedliche Messenger haben unterschiedliche Datenschutzrichtlinien. Die Tatsache, ob eine Brücke zu einem anderen Messenger-Ökosystem besteht, ist nicht zwingend transparent und die eigenen Daten können so in Systeme gelangen, dessen Nutzung man nie eingestimmt hat. Zudem stimmen Benutzer zwar den Datenschutzrichtlinien [Mata] von Matrix bei der Benutzung des *Matrix.org*-Servers zu, der Datenschutz beim hosten eines eigenen Servers ist jedoch ungewiss.

[18] `https://github.com/kamax-matrix/mxisd` (abgerufen am 4. September 2022)

7.6. Erkenntnisse

Matrix und dessen föderativer Ansatz unterscheiden sich grundlegend von bisherigen, zentralisierten Messengern. Es ermöglicht enorme Freiheit und Selbstbestimmung der Nutzer und theoretisch freie Wahl, welche und wie viele Daten Nutzer von sich selber preisgeben. Zur vollständigen Einordnung und Bewertung der Sicherheit der Daten und Metadaten der Nutzer genügt es nicht, Matrix geschlossen zu betrachten. Die unterschiedlichen Philosophien von zentralen oder föderativen Systemen resultieren in grundlegenden Unterschieden für die Nutzung, Datensammlung und Weiterentwicklung beider Ansätze. Somit führt die Frage zur Sicherheit der Metadaten der Nutzer unumgänglich zur Frage, ob zentrale oder föderative diese Daten besser schützen. Gleichzeitig hilft ein datenschützendes System wenig, wenn es nicht von genügend Anwendern genutzt wird. Daher soll folgend diskutiert werden, welche Vor- und Nachteile zentrale und föderative Systeme haben.

Zentral oder föderativ/dezentral?

Hauptunterschied von Matrix zu anderen Messengern wie WhatsApp ist der föderative Ansatz, welcher folglich zentral für die Untersuchung der Sicherheit der Metadaten von Element und Matrix sein soll. Anders als bei zentralisierten Messengern haben Nutzer von Matrix die Wahl, bei welchem Server sie ihr Konto anlegen und welchen Client sie zur Kommunikation nutzen. Das Prinzip entstammt wie zuvor erwähnt dem der E-Mail und der föderative Ansatz sorgt für mehr Freiheit der Nutzer. Gleichzeitig läuft nicht sämtliche Kommunikation über einen zentralen Punkt, welcher stets das Risiko bietet kompromittiert oder durch die Betreiber selbst überwacht zu werden. Der Entwickler des Signal-Protokolls und Mitgründer des zentralisierten Dienstes Signal Moxie Marlinspike betrachtete 2016 in einem Blogartikel diese dezentralisierte Freiheit kritisch [Mar16]:

> ”I no longer believe that it is possible to build a competitive federated messenger at all.”

Zunächst führt er die langsamere Möglichkeit zur Einbettung neuer Funktionen an, welche im Kontrast stehen zu rapiden Geschwindigkeit der Industrie. Als Beispiel werden dafür IP und HTTP angeführt, welche etwa 20 Jahre bis zur heutigen Weiterentwicklung benötigt haben. Zudem seien Slack mit IRC, Facebook mit E-Mail und WhatsApp mit XMPP erfolgreich gewesen, ein föderatives Anwendungsprotokoll in einen zentralen Dienst umzusetzen [Mar16]. Hingegen sei die Föderation von Mail und die Möglichkeit einen eigenen Mail-Server hosten zu können gleichzeitig der Grund, weshalb diese noch heute nicht E2E-verschlüsselt sind, ”and probably never will be”[Mar16]. Besonders relevant sind die hier angesprochenen Punkte über Föderation und Metadaten. Sie bietet die Auswahl des Servers zwar theoretisch die Möglichkeit für Nutzer auszuwählen, wer dessen Metadaten sehen kann, in der Praxis sammeln sich nach einiger Zeit jedoch ein Großteil der Nutzer und deren Daten an einem zentralen Server, was nach Marlinspike das Schlechteste aus beiden Welten sei. Abschließend soll die Sicherheit von Metadaten aus der Weiterentwicklung von Protokollen und Software entstehen, was in zentralisierten Diensten wesentlich besser kontrolliert werden können soll [Mar16].

Als Antwort darauf verfasste 2020 der Mitgründer von Matrix Matthew Hodgson einen Blogartikel [Hod20], in welchem er das Prinzip der Föderation verteidigt. So stimmt er Marlinspike zwar zu, dass dezentralisierte Dienste länger benötigen, um neue Funktionen durchzusetzen, gleichzeitig heiße dies nicht, dass es nicht möglich sei und Matrix ginge dieses Problem bewusst an. Ebenso sammeln sich viele Nutzer auf dem Matrix-Homeserver (etwa 35% des sichtbaren Netzwerkverkehrs [Hod20]), was ein potentielles Risiko für Metadaten darstellt. Gleichzeitig können diese Probleme behoben werden, indem Nutzer die Möglichkeit gegeben wird, ihre Metadaten selbst zu speichern und zu entscheiden, wer diese einsehen kann, sowie Verschleierung von Kommunikationspfaden durch "mixnets of blinded store & forward servers"[Hod20]. Nicht zuletzt sollen Nomadenkonten [Hof19] den "matrix.org"-Server obsolet machen, wodurch der Pseudo-Zentralisierung entgegengewirkt wird und der Standardserver auf dem Client des Nutzers laufen würde [Hod20]. Schlussendlich sticht jedoch ein Kernvorteil des föderativen Ansatzes heraus; die Freiheit der Nutzer, welche in folgenden Punkten gezeigt wird [Hod20]:

- Auswahl des Servers
- Möglichkeit, einen eigenen Server zu betreiben
- Wahl des Landes, in welchem der Server läuft
- Einschränkung der Metadaten und Verläufe/gespeicherten Daten
- Auswahl der verwendeten App
- Wahl der Funktionalität - Bots, Bridges, Integration und weitere
- Auswahl ob und welche Identifikatoren zur Authentifizierung genutzt werden
- freie Erweiterbarkeit des Protokolls

Matrix bietet eine enorme Freiheit im Vergleich zu zentralen Messengern und damit ein enormes Potential. Wie dies konkret genutzt wird, wird sich in der Zukunft zeigen. Im besten Fall überkommt Matrix seine aktuellen Probleme im Bezug auf die Privatsphäre der Nutzer und entwickelt ein freies, sicheres und datensparendes Protokoll, welches die Abgrenzung der aktuellen Messengerlandschaft aufweicht und damit sogar Konkurrenz unter diesen belebt. WhatsApp wird aktuell auch deshalb so verbreitet genutzt, weil viele es Nutzen: der Umstieg auf eine neue Plattform ist wesentlich einfacher, wenn Nutzer dabei nicht ihre alten Kontakte und Chatverläufe verlieren. Im schlechtesten Fall bleiben die aktuellen Probleme im Bereich der Daten von Nutzern, oder Matrix wird von zu wenig Anwendern genutzt. Die Verwendung von Matrix in Frankreichs Behörden und der Bundeswehr [Gru20] deutet zumindest auf eine positive Entwicklung hin. Obgleich sich Matrix in einer datensparenderen Version oder ein sich ebenfalls der Privatsphäre verschriebener Messenger wie Signal durchsetzen, so kann diese Konkurrenz auch einen ähnlichen Ausgang nehmen, wie die der beiden Unternehmen Microsoft und Apple: mit zwei konkurrierenden Firmen an der Spitze und obgleich unterschiedlicher Architektur ein durch Konkurrenz belebtes Wettrüsten. Den Metadaten der Nutzer soll es nicht schaden.

8. Corona-Warn-App

Das Jahr 2020 ist wohl von vielen Katastrophen geprägt. Jedoch veränderte keine unser bisheriges Leben so sehr, wie die Ausbreitung des Covid-19 Virus, auch Corona genannt. Zur Eindämmung des Virus hat die Bundesregierung eine App in Auftrag gegeben, welche potentiell von Corona erkrankte Personen warnen soll, die in der Nähe einer infizierten Person oder einer Kontaktperson zu dieser waren. Dadurch sollen Infektionsketten gebrochen und Bürger über ihr mögliches Risiko einer Erkrankung gewarnt werden. Bei dieser App handelt es sich um die Corona-Warn-App (CWA) des Robert Koch-Instituts [Cwaa],[Bun], welche im folgenden auf ihre Sicherheit, insbesondere in Bezug auf deren Metadaten und die damit verbundenen Möglichkeiten der Deanonymisierung der Nutzer untersucht werden soll.

8.1. Relevanz der Sicherheitsanalyse der Corona-Warn-App

Die CWA soll ein wesentlicher Faktor in der Reduktion der Ausbreitung des Corona-Virus sein. Damit dies gelingt, müssen möglichst viele Nutzer diese App auf ihrem Handy installieren. Die App wurde von der Bundesregierung selbst in Auftrag gegeben und obgleich es keine Verpflichtung gibt diese zu Nutzen, so wurden doch Empfehlungen zur Nutzung der App ausgesprochen und diese wird auf der offiziellen Webseite der Bundesregierung beworben [Bun]. Dabei sehen Kritiker in der App ein potentielles Risiko für die Privatsphäre der Nutzer, da eine von der Regierung verbreitete App zur Überwachung der Bevölkerung genutzt werden könnte, wie es aktuell in China und der in WeChat und Alibaba integrierten Corona-App der Fall ist [Dor20]. Umso wichtiger ist die Sicherheit, Datensparsamkeit und der Privatsphäreschutz durch die CWA, um ein Vertrauen der App in der Bevölkerung aufzubauen und die Nutzung dieser zu erhöhen. Denn einen tatsächlichen Nutzen und Einfluss auf das Infektionsgeschehen kann die CWA nur dann haben, wenn diese auch von einem hinreichend großen Teil der Bevölkerung genutzt wird.

8.2. Überblick und Funktion der Corona-Warn-App

Im Zentrum der Funktionalität steht das anonyme Austauschen von Kontaktpersonen, sowie eine Risikoeinschätzung und das schnelle Erfahren von Testergebnissen bei Test auf COVID-19. Die folgende Liste stammt von der offiziellen Webseite der CWA [Cwaa]:

- **RPI in der Nähe sammeln:** Senden und Empfangen von zufälligen RPI durch das *Exposure Notification System (ENF)* mittels *Bluetooth Low Energy*. Die RPI sind 10-20min gültig und von alle 24h geänderten Schlüsseln abgeleitet.
- **Testergebnis mitteilen:** Abrufen von Testergebnissen mittels QR-Code.
- **Liste von Schlüsseln infizierter Personen teilen:** Bei bestätigter Infektion freiwilliges Teilen der Testergebnisse und damit hochladen der temporären Schlüssel der letzten 14 Tage. Bestätigte Liste wird regelmäßig an alle Apps gesendet.
- **Prüfung auf Kontakt mit infizierten Personen:** Abgleichen der Liste infizierter Personen IDs und Risikoabschätzung.

8.3. Installation der Corona-Warn-App

Die App kann im App-Store für alle unterstützten Geräte für Google/Android unter `https://play.google.com/store/apps/details?id=de.rki.coronawarnapp` (abgerufen am 4. September 2022) und Apple/iOS unter `https://apps.apple.com/de/app/corona-warn-app/id1512595757` (abgerufen am 4. September 2022) heruntergeladen werden. Auf iOS läuft die App ab dem iPhone 6s und iOS 13.6. Für Android ist die CWA ab der Version Android 6 funktionsfähig[19].

8.4. Sicherheit der Corona-Warn-App

Aufgrund der vorher genannten Ursachen wurde ein wesentlicher Fokus auf die Sicherheit und den Datenschutz der CWA gelegt. Die App basiert auf einer dezentralen Serverarchitektur und wurde von der *Deutschen Telekom* und *SAP* entwickelt. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) und der Bundesbeauftragte für den Datenschutz und Informationsfreiheit arbeiteten ebenfalls an der App mit, um die Sicherheitsanforderungen zu gewährleisten [bun20]: Der Zentrale Server der CWA hat lediglich die Funktion, die anonymen Listen mit positiv getestet und gemeldeten Nutzer-IDs an die jeweiligen Nutzer der App zu verteilen. Die Bewertung des Risikos und der Austausch der Daten erfolgt dezentral auf den Mobiltelefonen der Nutzer. Zudem ist eine Echtzeitwarnung bei infizierten Personen im eigenen Radius aus datenschutzrechtlichen Gründen nicht möglich. Die Anonymität (Abschnitt 2.2.6) der Nutzer ist ebenfalls gewährleistet. Es werden keine personenbezogenen Daten wie E-Mail oder Telefonnummer erhoben, die App-eigenen Daten werden dezentral auf dem eigenen Gerät gespeichert und die versandten IDs pseudonymisiert. Ebenso werden die Begegnungen mit anderen Nutzern und deren IDs nur auf dem eigenen gerät gespeichert und verschlüsselt und nach 14 Tagen gelöscht. Zudem soll es nur der lokalen App, nicht aber dem Server möglich sein, Kontakt zu einer infizierten Person zu erkennen und einzuschätzen [bun20]. Zudem erfolgt die Meldung eines positiven Tests freiwillig und anonym. Kontakte erfahren nicht, dass man positiv getestet wurde, als auch umgekehrt erfährt man nach einer Positivmeldung nicht, wer informiert wird. Zuletzt ist die CWA und deren Serverarchitektur open source[20], wodurch sämtliche zuvor genannten Punkte selbst überprüft werden können. Diese Offenheit steigert das Vertrauen in die CWA enorm.

Für diese Arbeit besonders relevant ist die Betrachtung der Metadaten. Durch HTTPS und der Entfernung der Metadaten vor dem Versenden der Positivmeldung sollen Nutzerdaten nicht deanonymisierbar sein [Cwab]. Zudem sorgt ein Hintergrundrauschen von unechten Meldungen dafür, die Überwachung des Netzwerkverkehrs weiter zu erschweren [Kre20]. Ausführliche Informationen zum Datenschutz und der Sicherheit der CWA finden sich in den zuvor zitierten Quellen und unter Robert Koch-Institut, Bericht zur Datenschutz-Folgenabschätzung für die Corona-Warn-App der Bundesrepublik Deutschland, 2020 [RI20]. Die hohen Sicherheits- und Privatsphäreanforderungen der CWA wurden in diversen Artikeln [Kre20],[Kre],[Ver] und von IT-Experten [Kli20],[Rö20] gelobt.

[19] `https://www.coronawarn.app/de/faq/#minimum_requirements` (aufgerufen am 4. September 2022)
[20] `https://github.com/corona-warn-app` (abgerufen am 4. September 2022)

8.5. Prüfen der Sicherheit der Metadaten der Corona-Warn-App

Folgend werden die mit der Corona-Warn-App durchgeführten Testdurchläufe besprochen und ausgewertet. Die App lief auf dem Testsystem E (5.1.5). Es konnten aufgrund der zuvor besprochenen technischen Limitierungen nicht alle Bestandteile der App – insbesondere die Übertragung der IDs per Bluetooth – getestet werden. Insgesamt waren die Möglichkeiten sehr begrenzt, Daten aus der Paketübertragung zu erhalten, was wiederum für die Sicherheit der Corona-Warn-App spricht. Daher lag ein größerer Aspekt dieser Arbeit in dem Zusammentragen von Sicherheitsbewertungen und schließlich dem Aufstellen einer fundierten Aussage über die Sicherheit der CWA. Die durchgeführten Tests spielten dabei dennoch eine zentrale Rolle und werden nun im Folgenden besprochen.

8.5.1. Ablauf der Tests

Wichtig ist hierbei noch anzumerken, dass nicht wie in den zuvor durchgeführten Tests der Messenger gezielt Datenpakete versandt wurden konnten. Konkret wurde keine Möglichkeit gefunden, durch Interaktion mit der App durch den Nutzer gezielt Anfragen an einen Server zu senden. Somit blieb für die Testdurchläufe nur die Möglichkeit, den gesamten eingehenden und ausgehenden Datenverkehr des virtuellen Handys zu überwachen, nach eventuellen Paketen der CWA zu suchen und diese anschließend auszuwerten. Der Ablauf der Tests war daher wie folgt:
Zunächst wurde Wireshark und die virtuelle Maschine auf dem Testsystem A (5.1.1) gestartet. Die VM des Testsystem E (5.1.5) ist durch eine Netzwerkbrücke mit dem Internet verbunden und hat daher ihre eigene IP-Adresse. Sämtlicher Datenverkehr des Systems E läuft über diese IP. Nach dieser IP wird nun in Wireshark gefiltert mittels des bekannten "host"-Filters:

```
host 192.168.0.14
```

Code 3: Wireshark Filter: IP des Testsystems E

Als nächstes wird nach dem Systemstart der VM einige Zeit abgewartet, damit der Hintergunddatenverkehr minimiert wird. Nun startet der eigentliche Test. Mittels Wireshark beginnt die Aufzeichnung der Pakete und im Testsystem wird die Corona-Warn-App gestartet. In beiden Testdurchläufen konnten eine menge von Paketen der CWA zugeordnet werden. Dabei war zuvor bekannt, dass die CWA ihre Daten über Telekomserver verwaltet und diese an die Server der Telekom sendet. Da sonst keine anderen Applikationen auf dem Testsystem E mit der Telekom kommunizieren, kann sämtlicher an die Telekom gesendeter Datenverkehr sehr sicher der CWA zugeordnet werden. Das Beenden des Mitzeichnens der Datenpakete beendet ebenso den Testdurchlauf. Anschließend werden alle aufgezeichneten Pakete nach den relevanten, mit Telekom-Servern interagierenden Pakete gefiltert. Aufgrund der Größe der Aufzeichnung befinden sich die diese nicht vollständig im Anhang, dafür aber auf der mitgelieferten CD unter *"Bachelorarbeit\Wireshark Capture\Corona-Warn-App\…"*.

8.5.2. Erster Testdurchlauf: Corona-Warn-App Wireshark Mitschnitt

Es werden nun chronologisch die wichtigsten Pakete der ersten Aufzeichnung besprochen. Dabei wird ausführlich auf die Paketinformationen, die Art der Pakete und die aus der Paketübertragung erkennbaren Daten eingegangen. Der Test erfolgte am 20.11.2020, was später noch relevant sein wird. Abbildung 16 zeigt einen Ausschnitt des Mitschnitts der Datenpakete.

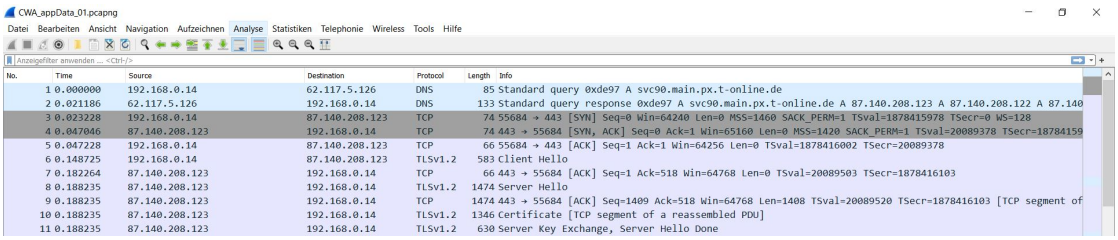


Abb. 16: CWA Wireshark Mitschnitt 01 - Übersicht (vom 20.11.2020)

Der vollständige Mitschnitt der Pakete befindet sich auf der mitgelieferten CD unter *"...\Corona-Warn-App\CWA_appData_01.pcapng"*. Die Datei *"\CWA_appData_01-filtered.pcapng"* enthält die für diese Arbeit relevanten Daten, ohne die als Hintergrunddaten identifizierten Pakete. Die Nummerierung entspricht den Paketen dieser Datei und wird zur Übersichtlichkeit mit aufgeführt.

No. 1-2: Domain Name System-Pakete

Die ersten beiden aufgezeichneten Pakete sind Domain Name System (DNS)-Pakete. Im ersten Paket wird eine Anfrage an den Nameserver gesendet, welche IP der Name "svc90.main.px.t-online.de" hat. Das zweite Paket ist die Antwort des Servers an die DNS-Anfrage, zu sehen in NetCap 4:

```
1  No.     Time           Source                Destination           Protocol Length Info
2  2       0.021186       62.117.5.126          192.168.0.14          DNS      133    Standard query response
               0xde97 A svc90.main.px.t-online.de A 87.140.208.123 A 87.140.208.122 A 87.140.208.121

4  Frame 2: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on interface \Device\NPF_{72F1DC27-4D81-4
       B0B-B7D1-00F842F88621}, id 0
5  Ethernet II, Src: 34:2c:c4:8c:1a:5b, Dst: 08:00:27:53:ba:32
6  Internet Protocol Version 4, Src: 62.117.5.126, Dst: 192.168.0.14
7  User Datagram Protocol, Src Port: 53, Dst Port: 50069
8  Domain Name System (response)
9      Transaction ID: 0xde97
10     Flags: 0x8180 Standard query response, No error
11     Questions: 1
12     Answer RRs: 3
13     Authority RRs: 0
14     Additional RRs: 0
15     Queries
16         svc90.main.px.t-online.de: type A, class IN
17     Answers
18         svc90.main.px.t-online.de: type A, class IN, addr 87.140.208.123
19         svc90.main.px.t-online.de: type A, class IN, addr 87.140.208.122
20         svc90.main.px.t-online.de: type A, class IN, addr 87.140.208.121
21     [Request In: 1]
22     [Time: 0.021186000 seconds]
```

Code 4: CWA Wireshark Mitschnitt 01 - No. 1-2: DNS-Pakete (vom 20.11.2020)

Relevant ist in der DNS-Abfrage hauptsächlich die Antwort, welche die möglichen IP-Adressen zeigt, zu welchen sich die App verbinden kann (Zeile 18-20 in NetCap 4):

- 87.140.208.123
- 87.140.208.122
- 87.140.208.121

Damit ist es nicht nur ein einziger Server, welcher mit der CWA kommuniziert und durch welchen die Daten verarbeitet werden.

Die restlichen 20 Pakete des Mitschnitts beinhalten den TCP-Verbindungsaufbau, sowie die TLS-Kommunikation, zu sehen in Abbildung 17:

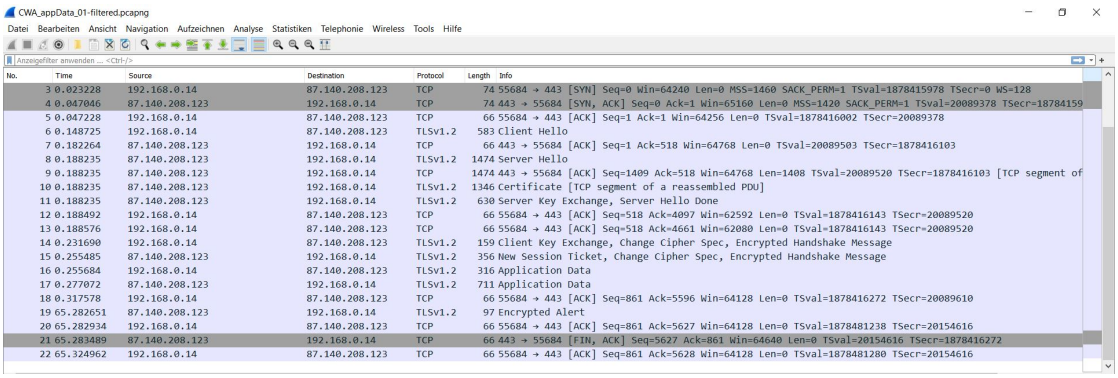


Abb. 17: CWA Wireshark Mitschnitt 01 - TCP- & TLS-Pakete (vom 20.11.2020)

Es folgt eine chronologische Ausarbeitung der wichtigsten erfassten Informationen aus den mitgeschnittenen Paketen. Dabei wird nicht ausführlich auf jedes einzelne Paket eingegangen, da dies den Rahmen dieser Arbeit sprengen würde.

No. 3-5: TCP-Verbindungsaufbau

Die Pakete Nummer 3-5 zeigen den Verbindungsaufbau nach dem Transmission Control Protocol (TCP). Der Client (192.168.0.14, das Handy mit der CWA) signalisiert dem Server (87.140.208.123), dass er eine Verbindung aufbauen möchte. Dieser bestätigt den Verbindungsaufbau, sowie den Erhalt des Paketes und sendet eine Bestätigung an den Client. Der Client wiederum bestätigt den Erhalt der Antwort des Servers und die TCP-Verbindung ist aufgebaut. Bei den gezeigten und allen folgenden Paketen sind TCP-Optionen (engl. "Options") angegeben, konkret TCP-Zeitstempel (engl. "Time-stamps"). Diese können für die Bestimmung der Reihenfolge der Pakete oder den Protection Against Wrapped Sequence Numbers (PAWS)-Algorithmus [RFC1323] genutzt werden. Paket Nummer 3 und 4 haben aber noch weitere gesetzte Optionen, da es sich bei diesen um SYN-Pakete handelt, welche womöglich von Interesse sind und daher nun betrachtet werden.

```
1  No. Time        Source          Destination
       Protocol
2    3 0.023228    192.168.0.14    87.140.208.123 TCP
3  Length Info
4  74     55684 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS
       =1460 SACK_PERM=1 TSval=1878415978 TSecr=0 WS
       =128
5  ...
6  Options: (20 bytes), Maximum segment size, SACK
       permitted, Timestamps, No-Operation (NOP),
       Window scale
7      TCP Option - Maximum segment size: 1460 bytes
8      TCP Option - SACK permitted
9      TCP Option - Timestamps: TSval 1878415978, TSecr
           0
10     TCP Option - No-Operation (NOP)
11     TCP Option - Window scale: 7 (multiply by 128)
```

NetCap: CWA Wireshark Mitschnitt 01 - No. 3: Optionen (vom 20.11.2020)

```
1  No. Time        Source          Destination
       Protocol
2    4 0.047046    87.140.208.123 192.168.0.14   TCP
3  Length Info
4  443 → 55684 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
       MSS=1420 SACK_PERM=1 TSval=20089378 TSecr
       =1878415978 WS=128
5  ...
6  Options: (20 bytes), Maximum segment size, SACK
       permitted, Timestamps, No-Operation (NOP),
       Window scale
7      TCP Option - Maximum segment size: 1420 bytes
8      TCP Option - SACK permitted
9      TCP Option - Timestamps: TSval 20089378, TSecr
           1878415978
10     TCP Option - No-Operation (NOP)
11     TCP Option - Window scale: 7 (multiply by 128)
```

NetCap: CWA Wireshark Mitschnitt 01 - No. 4: Optionen (vom 20.11.2020)

Die Maximum Segment Size (MSS) in Zeile 7 gibt von beiden Kommunikationspartnern an, wie viele Applikationsdaten maximal in einem Paket (Segment) versandt werden können [RFC879]. Damit soll die Fragmentierung der IP-Pakete verhindert werden. Selective Acknowledgment (SACK) ist auf beiden Seiten gestattet und beschreibt ein erweitertes Verfahren zur Bestätigung oder den Verlust von Paketen [RFC2018]. Zuletzt wurde in beiden Optionen für die Fensterskalierung (engl. "Window Scale") der Wert 7 angegeben, wodurch das Segmentfenster um den Faktor 7 multipliziert wird [RFC1323]. Diese Optionen wurden ebenso in allen Testdurchläufen mit der CWA festgestellt.

No. 6-15: TLS-Verbindungsaufbau

Ab Paket Nummer 6 beginnt der Client mit dem Aufbau einer verschlüsselten TLS- Verbindung gemäß dem Handshake-Protokoll. Dies ist nach zwei "round trips" mit dem Paket Nummer 15 abgeschlossen und ab dann kommunizieren beide Partner über einen sicheren, verschlüsselten Kanal, in welcher keinen Einblick gewährt. Der Client schickt im TLS-Paket Nummer 6 dem Server eine Liste von möglichen Cipher Suites zur Auswahl. Der Server entscheidet sich im TLS-Paket Nummer 8 für die folgende Cipher Suite: "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384" (0xc030). Daraus erschließen sich folgende Informationen über die verwendete Cipher Suite [RFC5289]:

- **Protokoll:** Transport Layer Security (TLS)-Protokoll
- **Schlüsselaustausch:** Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)
- **Authentifikation:** Rivest Shamir Adleman (RSA)-Algorithmus
- **Verschlüsselung:** Advanced Encryption Standard (AES) mit 256Bit Schlüssel im Galois/Counter-Modus
- **Hash:** Secure Hash Algorithm (SHA) 384

Die Verbindung zum Server ist somit abgesichert und MitM-Angreifer können keinen Einblick in die versandten Daten erhalten.

Beim Schlüsselaustausch wurde ECDHE verwendet. Dies bietet *PFS* (4.3.1), wobei die ausgetauschten Schlüssel temporär und nicht zwingend autorisiert sind. Eine Autorisierung des Servers mittels Zertifikates erfolgt im Paket No. 10. Das Telekom-Server Zertifikat beinhaltet folgende Informationen (NetCap 17, Zeile 20):

- id-at-countryName = DE
- id-at-organizationName = Deutsche Telekom AG
- id-at-organizationalUnitName = NSO-DS
- id-at-stateOrProvinceName = Hessen
- id-at-localityName = Darmstadt
- id-at-commonName = svc90.main.px.t-online.de

Ausgestellt wurde dieses Zertifikat von folgender Institution (NetCap 17, Zeile 26):

- id-at-countryName = DE
- id-at-organizationName = T-Systems International GmbH
- id-at-organizationalUnitName = T-Systems Trust Center
- id-at-stateOrProvinceName = Nordrhein Westfalen
- id-at-postalCode = 57250
- id-at-localityName = Netphen
- id-at-streetAddress = Untere Industriestr. 20
- id-at-commonName = TeleSec ServerPass Class 2 CA

Das Paket Nummer 10 enthält für diese Organisation ebenfalls ein Zertifikat, welches von "T-TeleSec GlobalRoot Class 2" ausgestellt wurde. Dadurch erfolgte eine Zertifizierung des Telekom-Servers gegenüber der CWA und damit dem Nutzer.

No. 16-18: TLS-Applikationsdaten

In dem TLS-Kanal senden sich Server und CWA zunächst verschlüsselte Daten (Paket No. 16). Dabei kann es sich durchaus um die Liste der vom Benutzer begegneten IDs handeln, sowie die zur Risikoeinschätzung notwendige Liste des Servers. Dafür spricht auch, dass die Applikationsdaten des Clients kleiner sind als die des Servers. Währenddessen werden TCP-Pakete gegenseitig versendet, um den Erhalt der Nachrichten zu bestätigen. Dies alles geschieht in knapp 0,3 Sekunden.

No. 19-22: TLS-Alert & TCP-Verbindungsabbau

Die letzten vier aufgezeichneten Pakete (No. 19-22) bauen die Verbindung nach 65 Sekunden ohne versandte Daten ab. Der Server sendet einen über TLS verschlüsselten Alarm (engl. "Encrypted Alert"). Dies ist üblich beim Sitzungsabbau, um ein automatisches Wiederverbinden des Clients zu unterbinden. In dem Fall handelt es sich um einen "Close notify Alert". Dennoch wird dem meisten Alarmen – insbesondere den fatalen – die Sitzung anschließend geschlossen. Aufgrund der Inaktivität kann hier davon ausgegangen werden, dass dies wahrscheinlich der Grund für das Beenden der Sitzung war. Der Client bestätigt den Erhalt mit dem folgenden Paket und der Server sendet ein TCP-Paket mit gesetzter FIN-Flag. Dadurch ist die Verbindung als geschlossen angesehen und die CWA bestätigt den Erhalt im letzten Paket.

8.5.3. Zweiter Testdurchlauf: Corona-Warn-App Wireshark Mitschnitt

Dieser Testdurchlauf folgt derselben Struktur des vorherigen. Es sollen gewonnene Erkenntnisse bestätigt und Unterschiede zum vorherigen Test herausgearbeitet werden. Zu finden ist dieser Testdurchlauf vollständig auf der mitgelieferten CD unter *"...\Corona-Warn-App\CWA_appData_02.pcapng"*. Der Mitschnitt wurde nach den für diese Arbeit relevanten Paketen gefiltert und unter *"...\CWA_appData_02-filtered.pcapng"* gespeichert. Abbildung 18 zeigt einen Ausschnitt der Aufzeichnung der Datenpakete:

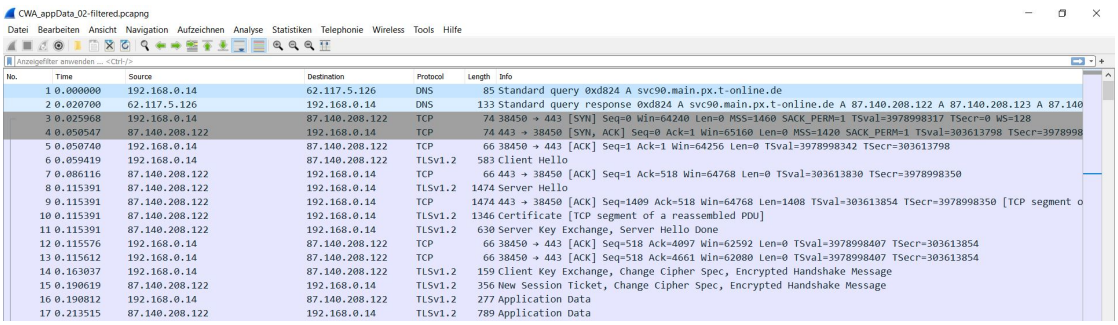


Abb. 18: CWA Wireshark Mitschnitt 02 - Übersicht (vom 01.12.2020)

No. 1-2: DNS-Pakete

Die DNS-Anfrage entspricht der des ersten Testdurchlaufs. Abermals werden für den Namen "svc90.main.px.t-online.de" die folgenden drei IP-Adressen zurückgegeben:

- 87.140.208.122
- 87.140.208.123
- 87.140.208.121

Lediglich die Reihenfolge der IP-Adressen änderte sich zum vorherigen Test.

No. 3-5: TCP-Verbindungsaufbau

Der TCP-Verbindungsaufbau ist ebenfalls fast identisch zu dem vorherigen Test, bis auf die IP-Adresse, zu welchen sich der Client verbindet. Diese wechselt von "87.140.208.123" auf "87.140.208.122". In beiden Fällen war die zur Verbindung zum Server genutzte IP-Adresse die erste, welche bei der DNS-Anfrage zurückgesendet wurde. Damit konnte gleichzeitig gezeigt werden, dass in der Praxis verschiedene Server zur Kommunikation genutzt werden und nicht nur ein zentraler oder konstanter Server.

No. 6-15: TLS-Verbindungsaufbau

Hier gibt es keinerlei Änderungen zum vorherigen Testdurchlauf, weshalb die erhaltenen Informationen im Rahmen des Tests als konstant angesehen werden können. Als Cipher Suite wird erneut "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384" (0xc030) verwendet und die gesendeten Zertifikate sind identisch mit dem ersten Test.

No. 16-160: TLS-Applikationsdaten

Der Ablauf des Versendens der Pakete entsprechen dem der Applikationsdaten des vorherigen Tests. Bemerkenswert ist hierbei, dass deutlich mehr Applikationsdaten versandt wurden, als beim ersten Test. Waren es im ersten Test mit TCP-Acknowledgement noch 3 Pakete, so wurden nun insgesamt 145 Pakete aufgezeichnet. 0,37 Sekunden nach Aufzeichnen des ersten Paketes wurden sämtliche Applikationsdaten versandt. Im ersten Test sendete der Client nur ein Paket mit Applikationsdaten. Im zweiten waren es zwei Pakete, wobei die Größe bei allen Paketen variierte. Bis auf Paket Nummer 17 wurden alle folgenden vom Server versendeten Pakete nach den Paketen des Clients gesendet. Die Ursache, weshalb im zweiten Testdurchlauf deutlich mehr Applikationsdaten versandt wurden, kann allein aus den Aufzeichnungen nicht geschlussfolgert werden.

8.6. Möglichkeiten zur Deanonymisierung der Corona-Warn-App

Die CWA wurde von Beginn der Entwicklung so entworfen, dass möglichst wenig Metadaten anfallen und eine Deanonymisierung der Nutzer nicht möglich ist. Dabei konnten in den Test lediglich die Metadaten der Paketübertragung überprüft werden.

Metadaten der Paketübertragung

In dem Bericht zur Datenschutz-Folgeabschätzung des Robert Koch-Instituts heißt es dazu [RI20, S. 60]: ”Nur IP-Adressen, die als Angreifer erkannt wurden, werden zur Gefahrenabwehr bis zu eine Stunde lang gespeichert”. Ansonsten sollen IP-Adressen direkt nach der Verarbeitung der Anfrage gelöscht werden. Des Weiteren werden die IP-Adressen der Nutzer durch einen ”Load Balancer” [RI20, S. 46] maskiert und danach nicht mehr zur Beantwortung der Anfrage an den Server verarbeitet. Zu den Zugriffsdaten kommen Zeitstempel, Größe der Pakete und Meldungen über den Erfolg der Anfrage hinzu, welche in den Test ebenfalls festgestellt werden konnten. Insgesamt sorgen die vorgenommenen Maßnahmen dafür, dass die Deanonymisierung von Nutzern durch Metadaten der Kommunikation nach aktuellem Stand nicht möglich ist.

Metadaten der RPI

Die zufällig generierten Rolling-Proximity-Identfier (RPI) werden per Bluetooth ausgetauscht und sind Hauptinformation der Begegnungsaufzeichnung der CWA. Diese hat keinen Zugriff auf eigene oder fremde RPI [RI20, S. 47]. Die Verwaltung erfolgt über das ENF. In den RPI sind die Metadaten Datum, Dauer und die Signalstärke der Begegnung gespeichert, welche zur Risikoermittlung der CWA genutzt werden. Somit ist auch hier nach aktuellem Wissen keine Deanonymisierung von Nutzern möglich.

Die Testergebnisse zeigen keine Möglichkeit zur Deanonymisierung der Nutzer auf. Die Offenheit des Quellcodes, dezentrale Umsetzung, datensparende Protokolle und sichere Übertragung der anonymen Nutzerdaten tragen zum Schutz der Privatsphäre bei, sowohl vor den Betreibern der CWA, als auch vor Außenstehenden MitM-Angreifern.

8.7. Hilft die CWA die Ausbreitung des Virus zu verringern?

Der Fokus dieser Arbeit liegt auf der sicherheitstechnischen Betrachtung der App. Dennoch sollen abschließend auch einige Worte darüber verloren werden, wie hilfreich die CWA in der Eindämmung der Pandemie ist. Dies trägt zu einer vollständigeren Betrachtung der CWA bei.

Obgleich die Sicherheit der App wenig kritisiert wurde, so doch deren Beitrag zur Pandemiebekämpfung. Zum einen hat nicht jeder Bürger ein Handy, dies auch nicht immer bei sich und die App ist auch nicht auf allen Geräten lauffähig (Abschnitt 8.3). Zudem ist für die Funktionalität der CWA Bluetooth notwendig, welches sowohl Sicherheitslücken aufweist [Xu+19], als auch Übertragungsprobleme haben kann [Sch20]. Des Weiteren kann die App nur dann hilfreich sein bei der Bekämpfung der Ausbreitung des Virus sein, wenn hinreichend viele Bürger die CWA nutzen. Bis zum 08.01.2021 haben etwa 24.9 Millionen Nutzer die App heruntergeladen [Rki]. Einen wesentlichen Beitrag hat die App ebenfalls bei der schnellen Übermittlung der Testergebnisse geliefert. Je schneller eine von Corona infizierte Person ihre Infektion erfährt, desto schneller können Maßnahmen zur Isolation getroffen und mögliche weitere Ansteckungen verhindert werden. Aktuell können rund 90 Prozent der Labore ihre Ergebnisse digital übermitteln und bis heute wurden 6,9 Millionen Testergebnisse übermittelt [Rki]. Davon waren 320.000 Personen positiv auf das Corona-Virus getestet wurden und 185.00 (58%) entschieden sich dafür, ihr Testergebnis zu teilen und vorherigen Begegnungen zu warnen. Der tatsächliche Nutzen der App ist schwer zu bemessen, da aufgrund des dezentralen Ansatzes keine Informationen vorliegen, wie viele Personen bislang von der App gewarnt wurden. Eine App zur Kontaktverfolgung ist umso wirkungsvoller, je mehr Bürger diese nutzen und bietet gleichzeitig mit mehr Nutzern ein geringeres Risiko für ein falsches Gefühl von Sicherheit [Fer+20],[SD20]. Daher wird die Nutzung der CWA auch weiterhin von der Bundesregierung beworben [Bun] und auch wenn die App nur eines von vielen Tools zur Bekämpfung Corona-Virus ist, so trägt sie doch ihren Teil zur Eindämmung bei.

8.8. Erkenntnisse

Diese Arbeit fokussierte sich vor allem auf den Bereich der Metadaten der Paketübertragung der CWA. Dabei konnten einige Erkenntnisse gewonnen und belegt werden. Bei der Verbindung mit den Servern der CWA werden die Daten verschlüsselt und insbesondere die IP-Adresse von Nutzern und deren Metadaten durch Verwendung von verschiedenen Servern zur Kommunikation und Verschleierung der eigenen IP-Adresse geschützt. Die App ist in ihrer derzeitigen Version und nach aktuellem Wissen datenschutzrechtlich unbedenklich. Es wurden von Beginn der Entwicklung diverse Vorkehrungen zum Schutz der Daten der Nutzer getroffen, wobei der die dezentrale Architektur und Anonymisierung von Nutzerdaten einen wesentlichen Beitrag dafür leisten. Und auch wenn der tatsächliche Nutzen der CWA nicht abschätzbar ist, so schadet es zumindest nicht, diese zu nutzen.

9. Erkenntnisse & abschließende Worte

Abschließend sollen die wichtigsten Erkenntnisse dieser Arbeit zusammengetragen und reflektiert werden.

9.1. Zusammenfassung

Zu Beginn wurde der Aspekt der sicheren und anonymen Netzwerkübertragung theoretisch betrachtet und mit der Thematik der Metadaten verknüpft. Bereits hier zeigte sich, dass die Protokolle, welche in der Praxis genutzt werden (IP, TCP,…) in ihrer Konzeption nicht von Anfang an darauf ausgelegt waren, Datensicherheit zu realisieren oder möglichst wenig Daten der Kommunikationspartner in Form von Metadaten zu offenbaren. Heutige Protokolle und Verfahren zum Schutz der Daten von Netzwerkkommunikation setzen auf diesen Protokollen auf und verschlüsseln den Kanal auf der Ebene der Anwendungsschicht (SSL/TLS, HTTPS,…). Dadurch können dank *state-of-the-art* Kryptographieverfahren die versendeten Daten verschlüsselt werden. Die grundlegenden Metadaten der Kommunikation wie IP-Adressen, Zeitstempel und Paketgrößen- oder art sind dennoch von außen und von potentiellen MitM-Angreifern einsehbar.

Der Hauptteil dieser Arbeit thematisierte die Sicherheit der Daten und Metadaten von Instant Messengern, sowie die Überprüfung der in der Praxis verwendeten Messengern WhatsApp und Matrix (Element). Kernpunkte heutiger Sicherheitsanforderungen an Messengern sind Verfahren zur Wahrung von PFS, PCS und Authentizität. Die gesonderte Betrachtung der Sicherheit von Gruppenchats ist ebenfalls relevant, da die Verfahren zur Sicherung von Point-to-Point (P2P)-Verbindungen nicht direkt auf Gruppenverschlüsselung übertragbar sind. In Testverfahren wurden für die Anwendungen WhatsApp, Matrix (Element) und die Corona-Warn-App die anhand der Paketübertragung von außen einsehbare Menge an Metadaten ermittelt. Gleichzeitig wurde gezeigt, dass die Daten und Metadaten der Applikationen bei der Kommunikation mit dem Server durch TLS geschützt und für MitM-Angreifer nicht einsehbar sind. Es erfolgte eine Differenzierung zwischen nach außen verschlüsselten und gleichzeitig für Serverbetreiber unverschlüsselten Metadaten. Letztere konnten durch die durchgeführten Tests nicht betrachtet werden und bieten womöglich zusätzliches Potential zur Deanonymisierung von Nutzern.

Zuletzt wurde die Thematik der zentralen und föderativen Ansätze von Applikationen betrachtet und im Kontext der Metadaten, insbesondere von WhatsApp und Matrix verglichen. Hier gliedert sich die Corona-Warn-App zwar nicht als Messenger, aber als dezentrales System ein. Für diese wurden die anfallenden Metadaten der Kommunikation mit dem Server praktisch und zwischen Nutzern per Bluetooth theoretisch betrachtet.

9.2. Erkenntnisse

Nun sollen die während der Arbeit gewonnenen Erkenntnisse zusammengetragen und die am Anfang formulierten Fragen beantwortet werden. Zunächst war es spannend zu sehen, dass und wie die in der Theorie bekannten kryptografischen Verfahren und Protokolle praktisch umgesetzt werden. Die jeden Tag von unzähligen Menschen unbewusst genutzten Protokolle konnten aufgezeichnet und den einzelnen Bits und Bytes der Paketübertragung zugeordnet werden.

Wie sicher sind aktuelle Messenger, insbesondere im Bezug auf Metadaten?

Die Sicherheit aktuell verbreiteter Messenger muss differenziert betrachtet werden: auf der einen Seite sind die vom Nutzer übertragenden Daten (Nachrichten, Dateien,…) per E2E-Verschlüsselung sowohl vor MitM-Angreifer, als auch den Serverbetreibern selbst geschützt. Auf der anderen Seite trifft dies für die Metadaten der Nachrichten und Kommunikation nicht automatisch zu. Die von außen (nicht durch TLS) verschlüsselten Metadaten wie IP-Adresse, Zeitstempel und Paketgrößen sind vor allem aufgrund der Funktionsweise von TCP/IP-Protokollen einsehbar. Ohne die Nutzung zusätzlicher Verschleierungsmethoden wie Hintergrundrauschen durch Dummy-Übertragungen oder zufällige Overheads bei Nachrichten, welche beide erhöhte Netzwerklasten zur Folge haben, ist eine Änderung der zugrunde liegenden Protokolle überlegenswert. Zumindest ist zu erkennen, dass eine Verschlüsselung der Daten allein nicht genügt, um die Anonymität der Nutzer zu wahren.

Wie sicher ist WhatsApp (Web) und dessen Metadaten?

WhatsApp zeigt die zuvor erläuterten Punkte beispielhaft: die Verschlüsselung der Nutzerdaten durch das Signal-Protokoll schützt die übertragenden Daten, während gleichzeitig durch die App selbst eine Vielzahl von Daten und Metadaten über die Nutzer gesammelt und übertragen werden, welche potentiell eine Deanonymisierung der Nutzer zulassen. Der zentrale Ansatz von WhatsApp, die Verschlüsselung der zusätzlich zur Nachricht übertragenen Metadaten durch TLS, sowie der nicht einsehbare Quellcode forcieren den Nutzer, WhatsApp mit dem Umgang der eigenen Daten nicht-nachprüfbar zu vertrauen. Ob man WhatsApp, sowie der Muttergesellschaft Facebook, an welche ein Teil der Daten weitergeleitet werden, vertrauen kann, muss letztendlich jeder Nutzer für sich selbst entscheiden.

Wie sicher ist Matrix und dessen Metadaten?

Matrix verfolgt mit dessen föderativem System einen grundsätzlich anderen Ansatz, als klassische Messenger. Die zusätzliche Freiheit der Nutzer soll zu mehr Kontrolle im Umgang und Verbreitung der eigenen Daten führen, was wiederum der Definition der Privatsphäre nahekommt. Die Verwendung des Signal-Protokolls bietet dieselben zuvor besprochenen sicherheitstechnischen Eigenschaften, während die Offenheit des Quellcodes

die Menge der verarbeiteten und versandten Daten überprüfbar macht. Theoretisch bietet Matrix Interoperabilität mittels sogenannter Bridges, welche jedoch aktuell noch in Entwicklung sind und u.a. aufgrund der Komplexität weit davon entfernt, von einer breiten Masse genutzt zu werden. Mittels föderativen Systemen soll die Kommunikation der Nutzer verteilt werden, was eine Deanonymisierung der Nutzer durch Überwachung eines Zentralen Servers erschwert. Die in der Theorie beschriebene Freiheit resultiert praktisch in einer Großzahl an zentral gebündelten Nutzern, wodurch eine Pseudo-Zentralisierung erfolgt. Gleichzeitig werden die zur Kommunikation gespeicherten Daten und Metadaten an sämtliche Kommunikationspartner und deren Homeserver gesendet. Hieran ist zu erkennen, dass eine Dezentralisierung allein nicht genügt, um Metadaten effektiv zu schützen. Wenn dezentral bedeutet, dass die eigenen Daten nicht nur beim Server des Nutzers, sondern auch bei jedem für die Kommunikation relevanten Server gespeichert werden, ist der Privatsphäre der Nutzer nicht geholfen.

Wie sicher ist die Corona-Warn-App und deren Metadaten?

Wie dezentrale Systeme, welche von Anbeginn der Entwicklung zur Wahrung der Privatsphäre konzipiert wurden, praktisch eine Deanonymisierung ihrer Nutzer verhindern können, zeigt die Corona-Warn-App. Eine von der Regierung in Auftrag gegebene und verbreitete App, welche zur effektiven Eindämmung der Pandemie von möglichst vielen Nutzern verwendet werden muss, benötigt das Vertrauen der Bevölkerung. Dies begründet die CWA durch die Offenheit des Quellcodes und die dadurch überprüfbar kleine Menge an sicher übertragenen anonymen Daten. Der dezentrale Ansatz ist hierbei zentral für die Anonymität der Nutzer. Ob und wie groß der Einfluss der CWA bei der Bekämpfung des Virus ist, ist zwar umstritten, zumindest bietet die App nach aktuellem Wissen keine Möglichkeit deren Nutzer zu deanonymisieren.

9.3. Einordnung & Ausblick

Die Ergebnisse dieser Arbeit zeigen, welche Metadaten durch aktuelle Protokolle wie IP, TCP und TLS bei der Paketübertragung anfallen. Die Bestimmung der Menge an von außen einsehbaren Metadaten ist grundlegend für die Überlegung und Betrachtung von Anwendungsprotokollen zur Reduktion oder Verschleierung dieser Menge an Daten. Zugleich zeigte sich in den letzten Jahren der Wert der Metadaten von Nutzern für Unternehmen, Regierungen, Geheimdienste und andere Akteure. Aufgrund der enormen Verbreitung von Messengern ist eine sicherheitstechnische Betrachtung dieser ebenso notwendig wie die Überprüfung der Möglichkeiten zur Deanonymisierung der Nutzer durch diese und weitere Anwendungen. In beide zuvor beschriebenen Bereiche ordnet sich die Arbeit ein, wobei sie als Grundlage zur weiteren Forschung und Überblick über den aktuellen Stand der Sicherheit von den betrachteten Netzwerkprotokollen und Anwendungen genutzt werden kann.

Dabei konnten aufgrund von TLS lediglich die unverschlüsselten Metadaten der Paketübertragung betrachtet werden. Die hinter TLS verschlüsselten Daten sind ebenso relevant für die Betrachtung der Privatsphäre von Nutzern wie die Metadaten der Kommu-

nikation. Diese konnten in den Testverfahren nicht unverschlüsselt aufgezeichnet werden. Ein Gesamtbild der Menge an übertragenen Metadaten konnte somit nicht zusammengetragen und überprüft werden. Lediglich die Datenschutzbestimmungen der Applikationen, sowie tiefgreifendere Testverfahren anderer Arbeiten gaben Ausschluss über die von der Anwendung verwendeten Metadaten. Dies ist ein Ausblick für folgende Arbeiten, gezielt die durch TLS verschlüsselten Metadaten zu betrachten und weitere Rückschlüsse über die Deanonymisierung von Nutzern zu treffen.

Des Weiteren hätten die betrachteten Applikationen auch ohne das Umgehen der TLS-Verschlüsselung umfangreicher analysiert werden können. So wurde bei der Betrachtung von WhatsApp nur der Web- und diesem identische Desktop-Client geprüft und aufgrund von technischen Problemen, zu großem Hintergrundrauschen und daher nicht eindeutig bestimmbaren Applikationsdaten nicht die mobile Version von WhatsApp selbst. Zumindest für Matrix ist es praktisch gelungen, einen Server zu erstellen und dessen Metadaten auch unverschlüsselt zu empfangen. Da auch hier technische Probleme auftraten, welche in der vorgegebenen Zeit nicht überwunden werden konnten, konnte keine Verbindung zu anderen Servern oder per Bridges zu anderen Anwendungen hergestellt werden. Auch andere Messenger von Matrix als Element, können im Rahmen einer Analyse der anfallenden Metadaten betrachtet werden. Zuletzt wurde bei der Corona-Warn-App lediglich die Verbindung zum Server getestet. Dabei konnte das Übertragen von Testergebnissen nicht getestet werden, da diese einem Authentifizierungsprozess durchlaufen. Ebenso konnte aufgrund von fehlendem technischem Equipment die Bluetooth-Übertragung der RPI nicht aufgezeichnet und ausgewertet werden. Die Analyse lediglich einer der drei Applikationen hätte eine fokussiertere Betrachtung ermöglicht. Alle zuvor genannten Punkte sind Möglichkeiten zur Analyse von Metadaten der jeweiligen Anwendungen und potentielle Bereiche, welche zusätzlich überprüft werden können.

Ebenso wichtig ist die Betrachtung der Verschlüsselung und Menge an übertragenen Metadaten von Gruppenkonversationen. Diese unterscheiden sich jedoch in vielen Aspekten gerade in Bezug auf Sicherheit von P2P-Konversationen und eröffnen ganz neue Fragestellungen, weshalb sie in dieser Arbeit lediglich kurz betrachtet wurden. Vor allem im Hinblick auf die Deanonymisierung von Nutzern sind (gerade politische) Gruppen gefährdet, weshalb aktuell viel auf dem Gebiet der Anonymisierung von Gruppenchats geforscht wird.

Zuletzt wurde in dieser Arbeit vermehrt das Problem der dem Internet zugrunde liegenden Protokolle TCP und IP im Bezug auf Metadaten besprochen. Ein spannender Forschungsaspekt wäre die Betrachtung möglicher Verbesserungen oder Alternativen zu diesen Protokollen, ungeachtet derer enormen Verbreitung und daher resultierenden Umstiegskosten. Es sollte die Frage gestellt werden, ob Verfahren zur Sicherung von Daten und Metadaten, welche auf diesen Protokollen aufsetzen, genügen, oder man die Protokolle selbst nicht grundlegend überdenken sollte, welche nie für diesen Zweck konzipiert waren. Den zumindest zwei Punkte wurden hoffentlich in dieser Arbeit deutlich: nachträglich Daten und Metadaten zu schützen ist schwieriger, als dies von Beginn an zu konzipieren und die Verschlüsselung von Daten allein genügt nicht, um Metadaten der Nutzer und diese vor einer Deanonymisierung zu schützen.

Anhang

A. Testumgebung Systeminformationen, Programme & Versionen

Betriebsystemname	Microsoft Windows 10 Home
Version	10.0.19041 Build 19041
Betriebsystemhersteller	Microsoft Corporation
Systemhersteller	LENOVO
Systemmodell	3109
Systemtyp	x64-basierter PC
System-SKU	LENOVO_MT_3109
Prozessor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 3401 MHz, 4 Kern(e), 8 logische(r) Prozessor(en)
BIOS-Version/-Datum	LENOVO EQKT20AUS, 29.10.2012
SMBIOS-Version	2.7
Version des eingebetteten Controllers	255255
BIOS-Modus	UEFI
BaseBoard-Hersteller	LENOVO
BaseBoard-Version	Win8 STD MM DPK IPG
Plattformrolle	Desktop
Sicherer Startzustand	Ein
Hardwareabstraktionsebene	Version = "10.0.19041.488"
Zeitzone	Mitteleuropäische Sommerzeit
Installierter physischer Speicher (RAM)	16,0 GB
Gesamter physischer Speicher	16,0 GB
Verfügbarer physischer Speicher	11,2 GB
Gesamter virtueller Speicher	18,3 GB
Verfügbarer virtueller Speicher	11,8 GB
Größe der Auslagerungsdatei	2,38 GB

Abb. 19: Testsystem A - Computer, Windows 10 (5.1.1)

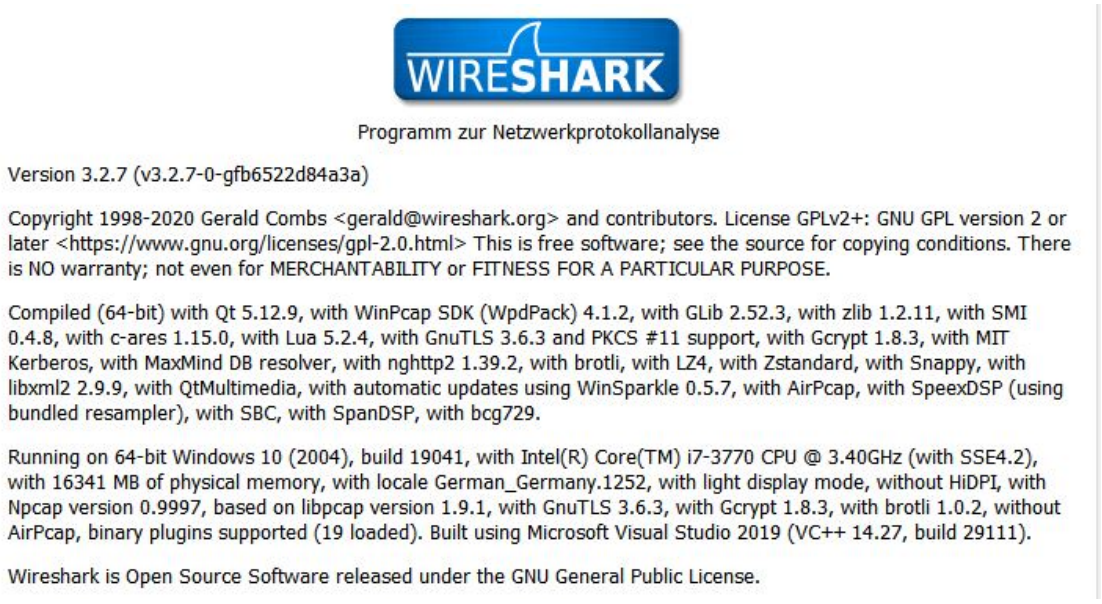
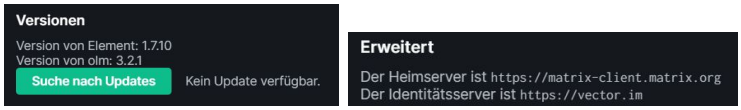


Abb. 20: Wireshark - Testsystem A (5.1.1)

(a) Chrome **(b)** Firefox **(c)** Microsoft Edge

Abb. 21: Browser - Testsystem A (5.1.1)



(a) Chrome **(b)** Firefox

Abb. 22: Matrix Elements - Testsystem A (5.1.1)

Abb. 23: Oracle VM VirtualBox auf Testsystem A (5.1.1)

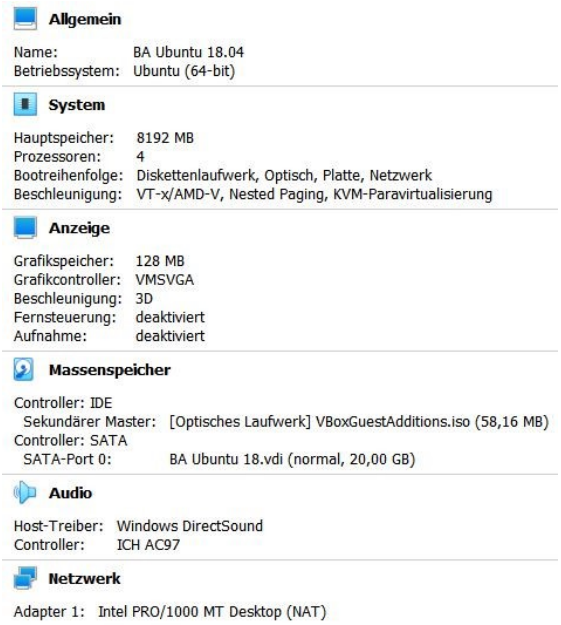


Abb. 24: Einstellungen VM VirtualBox Ubuntu 18.04 - Testsystem B (5.1.2)

Abb. 25: Systeminformationen Ubuntu 18.04 - Testsystem B (5.1.2)

Systeminformationen durch Ausführen des Kommandos "inxi -F" in der Console (installiert mit dem Befehl "sudo apt-get install inxi"):

```
System:    Host: dominik-VirtualBox Kernel: 5.4.0-52-generic x86_64 bits: 64 Desktop: Gnome 3.28.4
           Distro: Ubuntu 18.04.5 LTS
Machine:   Device: oracle System: innotek product: VirtualBox v: 1.2 serial: N/A
           Mobo: Oracle model: VirtualBox v: 1.2 serial: N/A BIOS: innotek v: VirtualBox date: 12/01/2006
CPU:       Quad core Intel Core i7-3770 (-MCP-) cache: 8192 KB
           clock speeds: max: 3392 MHz 1: 3392 MHz 2: 3392 MHz 3: 3392 MHz 4: 3392 MHz
Graphics:  Card: VMware SVGA II Adapter
           Display Server: x11 (X.Org 1.20.8 ) driver: vmwgfx Resolution: 1920x950@60.00hz
           OpenGL: renderer: llvmpipe (LLVM 10.0.0, 256 bits) version: 3.3 Mesa 20.0.8
Audio:     Card Intel 82801AA AC'97 Audio Controller driver: snd_intel8x0 Sound: ALSA v: k5.4.0-52-generic
Network:   Card: Intel 82540EM Gigabit Ethernet Controller driver: e1000
           IF: enp0s3 state: up speed: 1000 Mbps duplex: full mac: 08:00:27:4b:aa:7e
Drives:    HDD Total Size: 21.5GB (31.2% used)
           ID-1: /dev/sda model: VBOX_HARDDISK size: 21.5GB
Partition: ID-1: / size: 20G used: 6.3G (34%) fs: ext4 dev: /dev/sda1
RAID:      No RAID devices: /proc/mdstat, md_mod kernel module present
Sensors:   None detected - is lm-sensors installed and configured?
Info:      Processes: 227 Uptime: 8 min Memory: 1364.2/7961.8MB Client: Shell (bash) inxi: 2.3.56
```

Abb. 26: Systeminformationen Ubuntu 18.04 mittels "inxi -F" - Testsystem B (5.1.2)

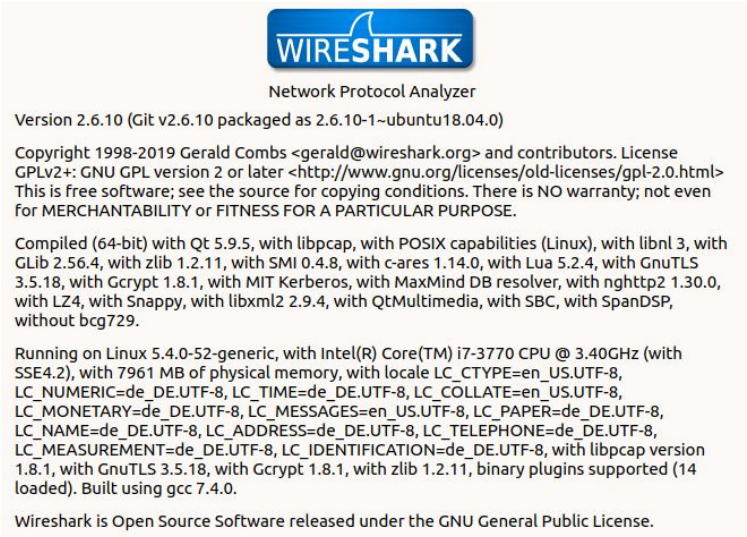


Abb. 27: Wireshark - Testsystem B (5.1.2)

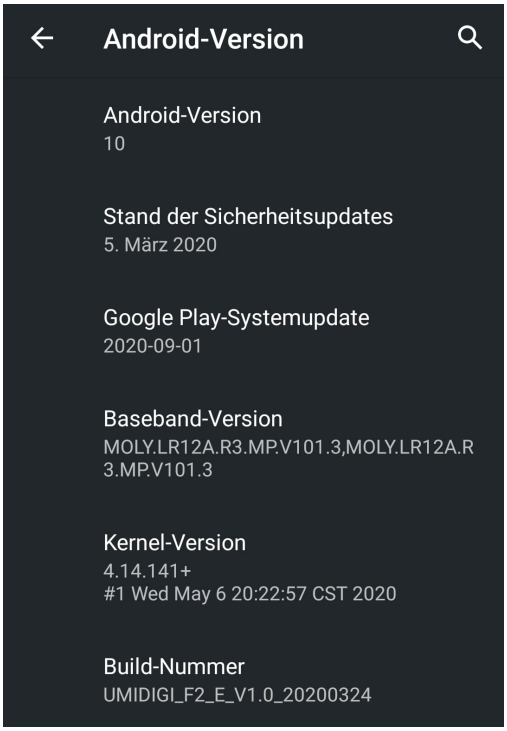


Abb. 28: Android - Testsystem C (5.1.3)

Model : F2
RAM : 6GB, LPDDR4X dual-channel
ROM : 128GB, UFS 2.1 flash storage
Battery : 5150mAh(Typ)
CPU : MediaTek, Helio P70, 4xCortex-A73, 2.1GHz & 4xCortex-A53, 2.0GHz
GPU : ARM Mali G72 MP3 900MHz
IEEE802.11 a/b/g/n/ac
Supports 5G Wi-Fi Signal / Wi-Fi Direct / Supports Wi-Fi Display
Supported network bands:
2G: GSM: B2 /B3 /B5 /B8
2G: CDMA1X: BC0 /BC1
3G: WCDMA: B1 /2 /4 /5 /6 /8
3G: TD-SCDMA: B34 /39
3G: CDMA EVDO: BC0 /BC1
4G: FDD-LTE: B1 /2 /3 /4 /5 /6 /7 /8 /12 /13 /17 /18 /19 /20 /26 /28A /28B /66 /71
4G: TDD-LTE: 34 /38 /39 /40 /41
https://www.umidigi.com/page-umidigi_f2_specification.html
(abgerufen am 4. September 2022)

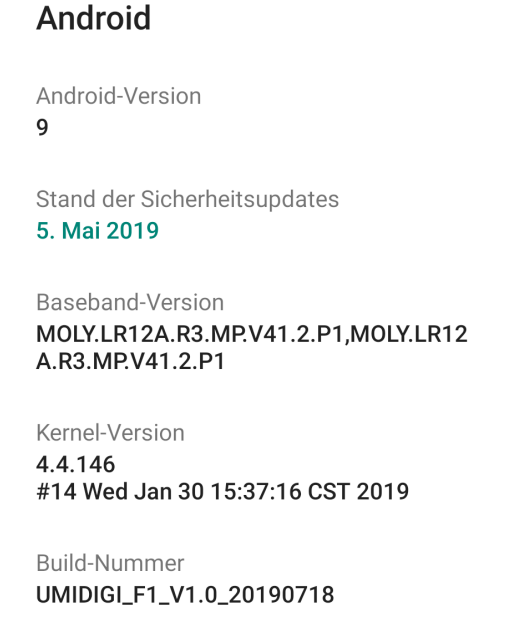


Abb. 29: Android - Testsystem D (5.1.4)

Model : F1
RAM : 4GB
ROM : 128GB
Battery : 5150mAh(Typ)
CPU : MTK, Helio P60, 4xCortex-A73, 2.0GHz & 4xCortex-A53, 2.0GHz
GPU : ARM Mali G72 MP3 800MHz
IEEE802.11 a/b/g/n/ac
Supports 5G Wi-Fi Signal / Wi-Fi Direct / Supports Wi-Fi Display
Supported network bands:
2G: GSM 2 /3 /5 /8 /CDMA1X BC0,BC1
3G: EVDO BC0,BC1 /WCDMA 1 /2 /4 /5 /6 /8 /19 /TD-SCDMA 34 /39
4G: TDD-LTE 34 /38 /39 /40 /41
4G: FDD-LTE 1 /2 /3 /4 /5 /7 /8 /12 /13 /17 /18 /19 /20 /25 /26 /28A /28B
https://www.umidigi.com/page-umidigi_f1_specification.html
(abgerufen am 4. September 2022)

B. Wireshark Netzwerkverkehr

Es folgt der in der Arbeit referenzierte aufgezeichnete Netzwerkdatenverkehr (engl. Network Capture (NetCap)). Dieser befindet sich ebenfalls auf den Dateien der mitgelieferten CD unter *"Bachelorarbeit\Wireshark Capture\..."*.

B.1. WhatsApp Web Netzwerkverkehr

WhatsApp Web: Mobile (Umidigi F2) nach Ubuntu Firefox

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  271    Application Data

Frame 1: 271 bytes on wire (2168 bits), 271 bytes captured (2168 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 40588, Seq: 1, Ack: 1, Len: 217
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000034184    10.0.2.15             185.60.217.53         TCP      54     40588 → 443 [ACK] Seq=1 Ack
           =218 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 40588, Dst Port: 443, Seq: 1, Ack: 218, Len: 0
```

NetCap 1: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu 01 (vom 15.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  271    Application Data

Frame 1: 271 bytes on wire (2168 bits), 271 bytes captured (2168 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 33520, Seq: 1, Ack: 1, Len: 217
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000039159    10.0.2.15             185.60.217.53         TCP      54     33520 → 443 [ACK] Seq=1 Ack
           =218 Win=63900 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 33520, Dst Port: 443, Seq: 1, Ack: 218, Len: 0
```

NetCap 2: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu 02 (vom 22.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  270    Application Data

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 37560, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000060031    10.0.2.15             185.60.217.53         TCP      54     37560 → 443 [ACK] Seq=1 Ack
           =217 Win=63900 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 37560, Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 3: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu 05 (vom 30.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  262    Application Data

Frame 1: 262 bytes on wire (2096 bits), 262 bytes captured (2096 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 37670, Seq: 1, Ack: 1, Len: 208
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000050771    10.0.2.15             185.60.217.53         TCP      54     37670 → 443 [ACK] Seq=1 Ack
          =209 Win=63900 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 37670, Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 4: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu 06 (vom 30.11.2020)

WhatsApp Web: Mobile (Umidigi F1) nach Ubuntu Firefox

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  270    Application Data

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 50510, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000035486    10.0.2.15             185.60.217.53         TCP      54     50510 → 443 [ACK] Seq=1 Ack
          =217 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 50510, Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 5: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F1) nach Ubuntu 01 (vom 23.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  270    Application Data

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 50510, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000034857    10.0.2.15             185.60.217.53         TCP      54     50510 → 443 [ACK] Seq=1 Ack
          =217 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 50510, Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 6: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F1) nach Ubuntu 02 (vom 23.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  270    Application Data

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 50510, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000060069    10.0.2.15             185.60.217.53         TCP      54     50510 → 443 [ACK] Seq=1 Ack
          =217 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 50510, Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 7: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F1) nach Ubuntu 03 (vom 23.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  270    Application Data

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 50510, Seq: 1, Ack: 1, Len: 216
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000037217    10.0.2.15             185.60.217.53         TCP      54     50510 → 443 [ACK] Seq=1 Ack
          =217 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 50510, Dst Port: 443, Seq: 1, Ack: 217, Len: 0
```

NetCap 8: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F1) nach Ubuntu 04 (vom 23.11.2020)

WhatsApp Web: Mobile (Umidigi F2) nach Ubuntu Chrome

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  262    Application Data

Frame 1: 262 bytes on wire (2096  bits), 262 bytes captured (2096  bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 41342, Seq: 1, Ack: 1, Len: 208
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000064072    10.0.2.15             185.60.217.53         TCP      54     41342 → 443 [ACK] Seq=1 Ack
          =209 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 41342, Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 9: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu Chrome 01 (vom 29.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  262    Application Data

Frame 1: 262 bytes on wire (2096  bits), 262 bytes captured (2096  bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 47508, Seq: 1, Ack: 1, Len: 208
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000041493    10.0.2.15             185.60.217.53         TCP      54     47508 → 443 [ACK] Seq=1 Ack
            =209 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 47508, Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 10: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu Chrome 02 (vom 29.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  262    Application Data

Frame 1: 262 bytes on wire (2096  bits), 262 bytes captured (2096  bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 47512, Seq: 1, Ack: 1, Len: 208
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000063453    10.0.2.15             185.60.217.53         TCP      54     47512 → 443 [ACK] Seq=1 Ack
            =209 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 47512, Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 11: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu Chrome 03 (vom 29.11.2020)

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000000    185.60.217.53         10.0.2.15             TLSv1.2  262    Application Data

Frame 1: 262 bytes on wire (2096  bits), 262 bytes captured (2096  bits) on interface enp0s3, id 0
Ethernet II, Src: 52:54:00:12:35:02, Dst: 08:00:27:4b:aa:7e
Internet Protocol Version 4, Src: 185.60.217.53, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 47512, Seq: 1, Ack: 1, Len: 208
Transport Layer Security

No.     Time           Source                Destination           Protocol Length Info
      2 0.000058780    10.0.2.15             185.60.217.53         TCP      54     47512 → 443 [ACK] Seq=1 Ack
            =209 Win=65535 Len=0

Frame 2: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface enp0s3, id 0
Ethernet II, Src: 08:00:27:4b:aa:7e, Dst: 52:54:00:12:35:02
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 185.60.217.53
Transmission Control Protocol, Src Port: 47512, Dst Port: 443, Seq: 1, Ack: 209, Len: 0
```

NetCap 12: Netzwerkverkehr WhatsApp Web - Nachrichten: Handy (F2) nach Ubuntu Chrome 04 (vom 29.11.2020)

B.2. Matrix (Element) Netzwerkverkehr

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.20.20.236	10.0.2.15	TLSv1.2	1093	Application Data
2	0.000377210	104.20.20.236	10.0.2.15	TLSv1.2	85	Application Data
3	0.001648749	10.0.2.15	104.20.20.236	TCP	54	47216 → 443 [ACK] Seq=1 Ack=1071 Win=63900 Len=0
4	0.052021946	10.0.2.15	104.20.20.236	TLSv1.2	211	Application Data
5	0.052366944	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1071 Ack=158 Win=65535 Len=0
6	0.104706885	104.20.20.236	10.0.2.15	TLSv1.2	242	Application Data
7	0.106381360	10.0.2.15	104.20.20.236	TLSv1.2	202	Application Data
8	0.106615086	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1259 Ack=306 Win=65535 Len=0

NetCap 13: Netzwerkverkehr Matrix (Element) - Nachrichten: Handy (F2) nach Ubuntu Desktop 01 (vom 06.12.2020)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.20.20.236	10.0.2.15	TLSv1.2	1093	Application Data
2	0.000377210	104.20.20.236	10.0.2.15	TLSv1.2	85	Application Data
3	0.001648749	10.0.2.15	104.20.20.236	TCP	54	47216 → 443 [ACK] Seq=1 Ack=1071 Win=63900 Len=0
4	0.052021946	10.0.2.15	104.20.20.236	TLSv1.2	211	Application Data
5	0.052366944	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1071 Ack=158 Win=65535 Len=0
6	0.104706885	104.20.20.236	10.0.2.15	TLSv1.2	242	Application Data
7	0.106381360	10.0.2.15	104.20.20.236	TLSv1.2	202	Application Data
8	0.106615086	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1259 Ack=306 Win=65535 Len=0

NetCap 14: Netzwerkverkehr Matrix (Element) - Nachrichten: Handy (F1) nach Ubuntu Desktop 01 (vom 06.12.2020)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.20.20.236	10.0.2.15	TLSv1.2	1093	Application Data
2	0.000377210	104.20.20.236	10.0.2.15	TLSv1.2	85	Application Data
3	0.001648749	10.0.2.15	104.20.20.236	TCP	54	47216 → 443 [ACK] Seq=1 Ack=1071 Win=63900 Len=0
4	0.052021946	10.0.2.15	104.20.20.236	TLSv1.2	211	Application Data
5	0.052366944	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1071 Ack=158 Win=65535 Len=0
6	0.104706885	104.20.20.236	10.0.2.15	TLSv1.2	242	Application Data
7	0.106381360	10.0.2.15	104.20.20.236	TLSv1.2	202	Application Data
8	0.106615086	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1259 Ack=306 Win=65535 Len=0

NetCap 15: Netzwerkverkehr Matrix (Element) - Nachrichten: Handy (F2) nach Ubuntu Web Chrome 01 (vom 06.12.2020)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.20.20.236	10.0.2.15	TLSv1.2	1093	Application Data
2	0.000377210	104.20.20.236	10.0.2.15	TLSv1.2	85	Application Data
3	0.001648749	10.0.2.15	104.20.20.236	TCP	54	47216 → 443 [ACK] Seq=1 Ack=1071 Win=63900 Len=0
4	0.052021946	10.0.2.15	104.20.20.236	TLSv1.2	211	Application Data
5	0.052366944	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1071 Ack=158 Win=65535 Len=0
6	0.104706885	104.20.20.236	10.0.2.15	TLSv1.2	242	Application Data
7	0.106381360	10.0.2.15	104.20.20.236	TLSv1.2	202	Application Data
8	0.106615086	104.20.20.236	10.0.2.15	TCP	60	443 → 47216 [ACK] Seq=1259 Ack=306 Win=65535 Len=0

NetCap 16: Netzwerkverkehr Matrix (Element) - Nachrichten: Handy (F2) nach Ubuntu Web Firefox 01 (vom 06.12.2020)

B.3. Corona-Warn-App Netzwerkverkehr

zur Übersichtlichkeit wurden hier nicht die kompletten Paketdetails jedes Mitschnittes dargestellt, sondern jeweils nur eine Übersicht. Die vollständigen Informationen zu den Paketen sind in der mitgelierferten CD unter *"Bachelorarbeit\Wireshark Capture\Matrix-Element\..."* zu finden.

```
No.     Time           Source                Destination           Protocol Length Info
10      0.188235       87.140.208.123        192.168.0.14          TLSv1.2  1346   Certificate [TCP segment of a reassembled PDU]

Frame 10: 1346 bytes on wire (10768 bits), 1346 bytes captured (10768 bits) on interface \Device\NPF_{72F1DC27-4D81-4B0B-B7D1-00F842F88621}, id 0
Ethernet II, Src: 34:2c:c4:8c:1a:5b, Dst: 08:00:27:53:ba:32
Internet Protocol Version 4, Src: 87.140.208.123, Dst: 192.168.0.14
Transmission Control Protocol, Src Port: 443, Dst Port: 55684, Seq: 2817, Ack: 518, Len: 1280
[3 Reassembled TCP Segments (4001 bytes): #8(1319), #9(1408), #10(1274)]
Transport Layer Security
    TLSv1.2 Record Layer: Handshake Protocol: Certificate
        Content Type: Handshake (22)
        Version: TLS 1.2 (0x0303)
        Length: 3996
        Handshake Protocol: Certificate
            Handshake Type: Certificate (11)
            Length: 3992
            Certificates Length: 3989
            Certificates (3989 bytes)
                Certificate Length: 2507
                Certificate: 308209…c7308208afa003020102021025733 3b160bef697a1dc86233dd8e017300d06092a (id-at-commonName=svc90.main.px.t-online.de,id-at-localityName=Darmstadt,id-at-stateOrProvinceName=Hessen,id-at-organizationalUnitName=NSO-DS,id-at-organizationalUnitName=NSO-DS,id-at-organizationName=Deutsche Telekom AG,id-at-countryName=DE)
                    signedCertificate
                        version: v3 (2)
                        serialNumber: 0x257333b160bef697a1dc86233dd8e017
                        signature (sha256WithRSAEncryption)
                        issuer: rdnSequence (0)
                            rdnSequence: 8 items (id-at-commonName=TeleSec ServerPass Class 2 CA,id-at-streetAddress=Untere Industriestr. 20,id-at-localityName=Netphen,id-at-postalCode=57250,id-at-stateOrProvinceName=Nordrhein Westfalen,id-at-organizationalUnitName=T
                        validity
                        subject: rdnSequence (0)
                        subjectPublicKeyInfo
                        extensions: 10 items
                    algorithmIdentifier (sha256WithRSAEncryption)
                    Padding: 0
                    encrypted: 3006562…ae7b0a590bc83adf97e868b6036648c916cb96e2bc0970a9130c98af75f8e40ae
                Certificate Length: 1476
                Certificate: 308205…c0308204a8a00302010202087e39c7ad1dd9f043300d06092a864886f70d01010b (id-at-commonName=TeleSec ServerPass Class 2 CA,id-at-streetAddress=Untere Industriestr. 20,id-at-localityName=Netphen,id-at-postalCode=57250,id-at-stateOrProvinceName=Nordrhein Westfalen,id-at-organizationalUnitName=T-Systems Trust Center,id-at-organizationName=T-Systems International GmbH,id-at-countryName=DE)
```

NetCap 17: CWA Wireshark Mitschnitt 01 - No. 10: Zertifikate (vom 20.11.2020)

Literaturquellen

Sämtliche angegebenen URLs waren am 4. September 2022 abrufbar.

Fachbücher

[BDSG] Robert Schwarz. *Bundesdatenschutzgesetz (BDSG)01*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, S. 103–108. ISBN: 978-3-658-20797-7. DOI: 10.1007/978-3-658-20797-7_8. URL: https://doi.org/10.1007/978-3-658-20797-7_8 (siehe S. 6).

[BM19] Alexandra Boldyreva und Daniele Micciancio. *Advances in Cryptology – CRYPTO 2019 : 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III / edited by Alexandra Boldyreva, Daniele Micciancio*. Security and Cryptology: 11694. Springer, 2019. ISBN: 9783030269548. URL: http://search.ebscohost.com/login.aspx?direct=true\&db=cat06365a\&AN=ulb.1676308482\&lang=de\&site=eds-live\&scope=site (siehe S. 1).

[Eck14] Claudia Eckert. *IT-Sicherheit Konzepte - Verfahren - Protokolle*. De Gruyter Oldenbourg, 2014. ISBN: 978-3-486-77848-9. URL: http://search.ebscohost.com/login.aspx?direct=true&db=edsgvk&AN=edsgvk.81812718X&lang=de&site=eds-live&scope=site (siehe S. 2, 4–14, 21).

[Sch15] Bruce Schneier. *Data and Goliath: The Hidden Battles to Capture Your Data and Control Your World*. 1st. W. W. Norton & Company, 2015. ISBN: 0393244814 (siehe S. 1).

[Tan03] Andrew S. Tanenbaum. *Computer networks*. 4th ed. Delhi: Pearson education (Singapore), 2003 (siehe S. 12, 13).

Artikel

[Arn18] Davis Mike Arndt. „DeadDrop: Message Passing Without Metadata Leakage“. In: (2018). URL: http://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.14AEFBAD&lang=de&site=eds-live&scope=site (siehe S. 18).

[AS16] Sebastian Angel und Srinath Setty. „Unobservable Communication over Fully Untrusted Infrastructure“. In: (Nov. 2016), S. 551–569. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/angel (siehe S. 18).

[Bah+20] Alireza Bahramali u. a. „Practical Traffic Analysis Attacks on Secure Messaging Applications“. In: *Proceedings 2020 Network and Distributed System Security Symposium* (2020). DOI: 10.14722/ndss.2020.24347. URL: http://dx.doi.org/10.14722/ndss.2020.24347 (siehe S. 2, 15).

[BB16] Gennie Gebhart Bill Budington. „Where WhatsApp Went Wrong: EFFs Four Biggest Security Concerns“. In: *Electronic Frontier Foundation* (2016-10-13). URL: https://www.eff.org/deeplinks/2016/10/where-whatsapp-went-wrong-effs-four-biggest-security-concerns (besucht am 26. 12. 2020) (siehe S. 37).

[CCG16] K. Cohn-Gordon, C. Cremers und L. Garratt. „On Post-compromise Security“. In: (2016), S. 164–178. DOI: 10.1109/CSF.2016.19 (siehe S. 21).

[CG+18] Katriel Cohn-Gordon u. a. „On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees“. In: CCS '18 (2018). Toronto, Canada, S. 1802–1819. DOI: 10.1145/3243734.3243747. URL: https://doi.org/10.1145/3243734.3243747 (siehe S. 23).

[CL19] Tom Carpay und Pavlos Lontorfos. „WhatsApp End-to-End Encryption: Are Our Messages Private?“ In: *SnE masters programme* (5. Feb. 2019) (siehe S. 1).

[Fer+20] Luca Ferretti u. a. „Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing“. In: *Science* 368.6491 (2020). ISSN: 0036-8075. DOI: 10.1126/science.abb6936. eprint: https://science.sciencemag.org/content/368/6491/eabb6936.full.pdf. URL: https://science.sciencemag.org/content/368/6491/eabb6936 (siehe S. 70).

[Geo09] Fanny Georges. „Self-representation and digital identity. A semiotic and quali-quantitative approach to the cultural empowerment of the Web 2.0“. In: *Cairn International Edition* No 154 (2009), S. 165–193. URL: https://www.cairn-int.info/article-E_RES_154_0165--self-representation-and-digital-identity.htm# (siehe S. 16).

[GIL19] YOSSI GILAD. „Metadata-Private Communication for the 99%.“ In: *Communications of the ACM* 62.9 (2019), S. 86. ISSN: 00010782. URL: http://search.ebscohost.com/login.aspx?direct=true&db=edb&AN=138293527&lang=de&site=eds-live&scope=site (siehe S. 13, 15, 17, 18).

[Hoo+15] Jelle van den Hooff u. a. „Vuvuzela: Scalable Private Messaging Resistant to Traffic Analysis“. In: SOSP '15 (2015), S. 137–152. DOI: 10.1145/2815400.2815417. URL: https://doi.org/10.1145/2815400.2815417 (siehe S. 18).

[JO15] Jakob Jakobsen und Claudio Orlandi. „On the CCA (in)security of MTProto“. In: (2015). URL: https://eprint.iacr.org/2015/1177 (siehe S. 23).

[JSH20] Kee Jefferys, Maxim Shishmarev und Simon Harman. „Session: A Model for End-To-End Encrypted Conversations With Minimal Metadata Leakage.“ In: (2020). URL: http://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.B2E84F92&lang=de&site=eds-live&scope=site (siehe S. 18).

[Jä18] Prof. Dr. Thomas Jäschke. „Für immer anonym: Wie kann De-Anonymisierung verhindert werden?“ In: *Abida - Assessing Big Data* (2018). Förderkennzeichen: 01IS15016A - F. URL: https://www.abida.de/sites/default/files/ABIDA\%20Gutachten\%20F\%c3\%9cR\%20IMMER\%20ANONYM.pdf (besucht am 29. 12. 2020) (siehe S. 10).

[LGZ18] David Lazar, Yossi Gilad und Nickolai Zeldovich. „Karaoke: Distributed Private Messaging Immune to Passive Traffic Analysis“. In: (Okt. 2018), S. 711–725. URL: https://www.usenix.org/conference/osdi18/presentation/lazar (siehe S. 18).

[MM16a] Trevor Perrin Moxie Marlinspike. „The Double Ratchet Algorithm“. In: *Signal* (20. Nov. 2016). URL: https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf (besucht am 26. 12. 2020) (siehe S. 22).

[MM16b] Trevor Perrin Moxie Marlinspike. „The X3DH key agreement protocol“. In: *Signal* (4. Nov. 2016). URL: https://signal.org/docs/specifications/x3dh/x3dh.pdf (besucht am 26. 12. 2020) (siehe S. 22).

[Puj+19] Alexandre Pujol u. a. „Spying on Instant Messaging Servers: Potential Privacy Leaks through Metadata“. In: *Transactions on Data Privacy* 12(2) (2019), S. 175–206. URL: https://hal.archives-ouvertes.fr/hal-02493276 (siehe S. 16, 17).

[Reh12] Christian Rehn. „Warum man Krypto nicht selbst machen sollte“. In: *christian-rehn.de* (12. Juni 2012). URL: http://www.christian-rehn.de/2012/06/12/warum-man-krypto-nicht-selbst-machen-sollte/ (besucht am 26. 12. 2020) (siehe S. 23).

[RH17] Nidhi Rastogi und James Hendler. „WhatsApp security and role of metadata in preserving privacy.“ In: (2017). URL: http://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.52627A02&lang=de&site=eds-live&scope=site (siehe S. 47).

[RMS18] Paul Rösler, Christian Mainka und Jörg Schwenk. „More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema“. In: *IEEE European Symposium on Security and Privacy, EuroS&P 2018* (2018) (siehe S. 23).

[Rü14] Stefanie Rühle. „Kleines Handbuch Metadaten“. In: *Yumpu* (2014). URL: https://www.yumpu.com/de/document/view/23832049/kleines-handbuch-metadaten (besucht am 20. 12. 2020) (siehe S. 17).

[SCS05] Kim Sangkyun und Leem Choon Seong. „Security of the internet-based instant messenger: Risks and safeguards“. In: *Internet Research* 15.1 (2005), S. 88 –98. ISSN: 1066-2243. URL: http://search.ebscohost.com/login.aspx?direct=true\&db=edsemr\&AN=edsemr.10.1108.10662240510577086\&lang=de\&site=eds-live\&scope=site (siehe S. 1).

[SD20] GERT WAGNER SUSANNE DEHMEL PETER KENNING. „Die Wirksamkeit der Corona-Warn-App wird sich nur im Praxistest zeigen. Der Datenschutz ist nur eine von vielen Herausforderungen“. In: (2020). Berlin: Sachverständigenrat für Verbraucherfragen. URL: https://www.svr-verbraucherfragen.de/wp-content/uploads/Policy_Brief_Corona_Tracing_App.pdf (siehe S. 70).

[SZ20] Ilya Sukhodolskiy und Sergey Zapechnikov. „Analysis of Secure Protocols nd Authentication Methods for Messaging“. In: *Procedia Computer Science* 169 (2020), S. 407 –411. ISSN: 1877-0509. URL: http://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S1877050920303616&lang=de&site=eds-live&scope=site (siehe S. 1, 19, 20, 22, 23, 37).

[Wha16] WhatsApp. „WhatsApp Encryption Overview - Technical white paper“. In: (5. Apr. 2016). Version 3 Updated October 22, 2020. URL: http://www.cdn.whatsapp.net/security/WhatsApp-Security-Whitepaper.pdf (besucht am 04. 01. 2021) (siehe S. 49).

[Xu+19] Fenghao Xu u. a. „BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals.“ In: (2019) (siehe S. 70).

Abschlussarbeiten

[Kac17] Fabian Kaczmarczyck. „Metadata Reduction for the Signal Protocol“. Masterarbeit. Humboldt-Universität zu Berlin, 2017 (siehe S. 3).

Online Quellen

[BSI19] *Mindeststandard des BSI zur Verwendung von Transport Layer Security (TLS)*. Bundesamt für Sicherheit in der Informationstechnik. 9. Apr. 2019. URL: https://www.bsi.bund.de/DE/Themen/StandardsKriterien/Mindeststandards_Bund/TLS-Protokoll/TLS-Protokoll_node.html (besucht am 25. 12. 2020) (siehe S. 21).

[Bun] *Bundesregierung Corona-Warn-App*. bundesregierung.de. 2021. URL: https://www.bundesregierung.de/breg-de/themen/corona-warn-app (besucht am 07. 01. 2021) (siehe S. 60, 70).

[bun20] bundesregierung.de. *Corona-Warn-App F.A.Q.* 29. Dez. 2020. URL: https://www.bundesregierung.de/resource/blob/1753814/1760022/c18fa6fac3fed5d68807c08ba0849bfa/2020-06-12-faq-langfassung-de-data.pdf?download=1 (besucht am 07. 01. 2021) (siehe S. 61).

[Coc19] Isobel Cockerell. *Inside China's Massive Surveillance Operation*. Wired. 5. Sep. 2019. URL: https://www.wired.com/story/inside-chinas-massive-surveillance-operation/ (besucht am 02. 01. 2021) (siehe S. 24).

[Cwaa] *Corona-Warn-App Webseite*. Corona-Warn-App. 2021. URL: https://www.coronawarn.app/de/ (besucht am 07. 01. 2021) (siehe S. 60).

[Cwab] *Häufig gestellte Fragen zur Corona-Warn-App - Kann ich durch Abhören des Netzwerkverkehrs als Corona-positiv identifiziert werden?* coronawarnapp.de. 2020. URL: https://www.coronawarn.app/de/faq/#traffic_monitoring (besucht am 07. 01. 2021) (siehe S. 61).

[Dof21] Zak Doffman. *WhatsApp Beaten By Apple's New iMessage Privacy Update*. Forbes. 3. Jan. 2021. URL: https://www.forbes.com/sites/zakdoffman/2021/01/03/whatsapp-beaten-by-apples-new-imessage-update-for-iphone-users/?sh=2ee546d83623 (besucht am 12. 01. 2021) (siehe S. 24).

[Dor19] Max Dor. *Notes on privacy and data collection of Matrix.org*. Kamax.io. Libre Monde ASBL. 12. Aug. 2019. URL: https://gitlab.com/libremonde-org/papers/research/privacy-matrix.org/-/blob/3fdc6db2d132c91d06a2a60bb6a384b3cb16873a/part1/README.md (besucht am 06. 01. 2021) (siehe S. 56).

[Dor20] Axel Dorloff. *Gekommen, um zu bleiben: die Corona-App in China*. BR24. 22. Aug. 2020. URL: https://www.br.de/nachrichten/deutschland-welt/gekommen-um-zu-bleiben-die-corona-app-in-china,S8Jq9Nj (besucht am 07. 01. 2021) (siehe S. 60).

[Dou17] Eva Dou. *Jailed for a Text: China's Censors Are Spying on Mobile Chat Groups*. The Wall Street Journal. 8. Dez. 2017. URL: https://www.wsj.com/articles/jailed-for-a-text-chinas-censors-are-spying-on-mobile-chat-groups-1512665007 (besucht am 02. 01. 2021) (siehe S. 24).

[Dtk] *Chinas Sozialkredit-System - Auf dem Weg in die IT-Diktatur*. Deutschlandfunk Kultur. 5. Sep. 2017. URL: https://www.deutschlandfunkkultur.de/chinas-sozialkredit-system-auf-dem-weg-in-die-it-diktatur.979.de.html?dram:article_id=395126 (besucht am 02. 01. 2021) (siehe S. 24).

[Ele] *Riot Web 1.6, RiotX Android 0.19 & Riot iOS 0.11 — E2E Encryption by Default & Cross-signing is here!!* Element.io. 6. Mai 2020. URL: https://element.io/blog/e2e-encryption-by-default-cross-signing-is-here/ (besucht am 06. 01. 2021) (siehe S. 52).

[EM13] Gabriel Dance Ewen Macaskill. *NSA files: decoded - What the revelations mean for you.* The Guardian. 1. Nov. 2013. URL: https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded (besucht am 23. 12. 2020) (siehe S. 17).

[EPP20] CPJ Middle East, North Africa Program und CPJ Technology Program. *Why Telegram's security flaws may put Iran's journalists at risk.* Committee to Protect Journalists. 31. Mai 2020. URL: https://cpj.org/2016/05/why-telegrams-security-flaws-may-put-irans-journal/ (besucht am 26. 12. 2020) (siehe S. 23).

[Eum] *Europäische Menschenrechtskonvention.* 19/05 Menschenrechte. BGBl. Nr. 210/1958 zuletzt geändert durch BGBl. III Nr. 30/1998. 1. Nov. 1998. URL: https://www.ris.bka.gv.at/Dokument.wxe?Abfrage=Bundesnormen&Dokumentnummer=NOR12016939 (besucht am 12. 01. 2021) (siehe S. 11).

[Gre13] Glenn Greenwald. *NSA collecting phone records of millions of Verizon customers daily.* The Guardian. 6. Juni 2013. URL: https://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order (besucht am 23. 12. 2020) (siehe S. 1, 17, 18).

[Gru20] Sebastian Gruener. *Bundeswehr will komplett auf Matrix-Chat wechseln.* Golem.de. 12. Mai 2020. URL: https://www.golem.de/news/messenger-bundeswehr-will-komplett-auf-matrix-chat-wechseln-2005-148407.html (besucht am 05. 01. 2021) (siehe S. 50, 59).

[Heg17] Lisa Hegemann. *Überwachung auf allen Kanälen? Wechat-Konto soll mit chinesischer ID verknüpft werden.* 28. Dez. 2017. URL: https://t3n.de/news/china-wechat-id-karte-895656/ (besucht am 02. 01. 2021) (siehe S. 24).

[Heg19] Lisa Hegemann. *How private are your favourite messaging apps?* Amnesty International. 21. Okt. 2019. URL: https://www.amnesty.org/en/latest/campaigns/2016/10/which-messaging-apps-best-protect-your-privacy/ (besucht am 02. 01. 2021) (siehe S. 24).

[Hil05] Diane Hillman. *Using Dublin Core.* Dublin Core. 7. Nov. 2005. URL: https://www.dublincore.org/specifications/dublin-core/usageguide/ (besucht am 23. 12. 2020) (siehe S. 17).

[Hod16] Matthew Hodgson. *Matrix's 'Olm' End-to-end Encryption security assessment released - and implemented cross-platform on Riot at last!* Matrix.org. 21. Nov. 2016. URL: https://matrix.org/blog/2016/11/21/matrixs-olm-end-to-end-encryption-security-assessment-released-and-implemented-cross-platform-on-riot-at-last (besucht am 06. 01. 2021) (siehe S. 52).

[Hod19a] Matthew Hodgson. *response to: Notes on privacy and data collection of Matrix.org*. Matrix.org. 14. Juni 2019. URL: https://matrix.org/~matthew/Response_to_-_Notes_on_privacy_and_data_collection_of_Matrix.pdf (besucht am 06. 01. 2021) (siehe S. 56).

[Hod19b] Matthew Hodgson. *Tightening up privacy in Matrix*. Matrix.org. 30. Juni 2019. URL: https://matrix.org/blog/2019/06/30/tightening-up-privacy-in-matrix (besucht am 06. 01. 2021) (siehe S. 56).

[Hod20] Matthew Hodgson. *On Privacy versus Freedom*. Matrix.org. 2. Jan. 2020. URL: https://matrix.org/blog/2020/01/02/on-privacy-versus-freedom (besucht am 06. 01. 2021) (siehe S. 52, 59).

[Hof19] Richard van der Hoff. *MSC1228: Removing MXIDs from events*. 14. Okt. 2019. URL: https://github.com/matrix-org/matrix-doc/blob/rav/proposal/remove_mxids_from_events/proposals/1228-removing-mxids-from-events.md (besucht am 07. 01. 2021) (siehe S. 59).

[Kli20] Thomas Klingbeil. *Corona-Warn-App - Behind the scenes: Invisible, yet important*. Chaos Computer Club e.V. 30. Dez. 2020. URL: https://media.ccc.de/v/rc3-11573-corona-warn-app#t=9 (besucht am 08. 01. 2021) (siehe S. 62).

[Kre] Dirk Kretzschmar. *IT-Sicherheit & Datenschutz der Corona-Warn-App*. TÜViT. URL: https://www.tuvit.de/de/tuevit/corona-warn-app/ (besucht am 08. 01. 2021) (siehe S. 62).

[Kre20] Stefan Krempl. *rC3: Wie die Corona-Warn-App vor Hackerattacken geschützt wird*. Heise online. 31. Dez. 2020. URL: https://www.heise.de/news/rC3-Wie-die-Corona-Warn-App-vor-Hackerattacken-geschuetzt-wird-5001233.html (besucht am 08. 01. 2021) (siehe S. 61, 62).

[Kuk20a] Mike Kuketz. *Element: Messaging über die Matrix – Messenger Teil7*. Kuketz-Blog. 29. Okt. 2020. URL: https://www.kuketz-blog.de/element-messaging-ueber-die-matrix-messenger-teil7/ (besucht am 06. 01. 2021) (siehe S. 56).

[Kuk20b] Mike Kuketz. *Messenger-Brücken sind datenschutzrechtlich bedenklich*. Kuketz-Blog. 11. Aug. 2020. URL: https://www.kuketz-blog.de/element-messaging-ueber-die-matrix-messenger-teil7/ (besucht am 07. 01. 2021) (siehe S. 57).

[Mar14] Moxie Marlinspike. *Free, Worldwide, Encrypted Phone Calls for iPhone*. Signal. 29. Juli 2014. URL: https://signal.org/blog/signal/ (besucht am 01. 01. 2021) (siehe S. 24).

[Mar16] Moxie Marlinspike. *Reflections: The ecosystem is moving*. Signal. 10. Mai 2016. URL: https://signal.org/blog/the-ecosystem-is-moving/ (besucht am 06. 01. 2021) (siehe S. 56, 58).

[Mata] *Matrix.org Homeserver Privacy Notice*. Matrix.org. 10. Aug. 2020. URL: https://matrix.org/legal/privacy-notice (besucht am 07. 01. 2021) (siehe S. 57).

[Matb] *Vector.im and Matrix.org Identity Servers Privacy Notice*. Matrix.org. 2020. URL: https://matrix.org/legal/identity-server-privacy-notice-1 (besucht am 07. 01. 2021) (siehe S. 57).

[Matc] *We have discovered and addressed a security breach*. Matrix.org Team. 11. Apr. 2019. URL: https://matrix.org/blog/2019/04/11/we-have-discovered-and-addressed-a-security-breach-updated-2019-04-12 (besucht am 07. 01. 2021) (siehe S. 56).

[Matd] *What Information Do You Collect About Me and Why?* Matrix.org. 2020. URL: https://matrix.org/legal/identity-server-privacy-notice-1#32-what-information-do-you-collect-about-me-and-why (besucht am 07. 01. 2021) (siehe S. 56).

[ME2E] *End-to-End Encryption implementation guide*. Matrix.org. 2020. URL: https://matrix.org/docs/guides/end-to-end-encryption-implementation-guide (besucht am 06. 01. 2021) (siehe S. 52).

[Meh20a] Matthias Mehner. *Nutzerzahlen Messenger Apps Deutschland und Weltweit*. 13. Aug. 2020. URL: https://www.messengerpeople.com/de/weltweite-nutzer-statistik-fuer-whatsapp-wechat-und-andere-messenger/ (besucht am 28. 09. 2020) (siehe S. 19).

[Meh20b] Matthias Mehner. *Studie zu Messenger Nutzung "OTT" 2020 in Deutschland*. Messenger People. 25. Mai 2020. URL: https://www.messengerpeople.com/de/studie-messenger-nutzung-2020-deutschland/ (besucht am 01. 01. 2021) (siehe S. 19).

[Mon] *WeChat confirms that it makes all private user data available to the Chinese government*. Moneycontrol. 19. Sep. 2017. URL: https://www.moneycontrol.com/news/business/companies/wechat-confirms-that-it-makes-all-private-user-data-available-to-the-chinese-government-2391847.html (besucht am 02. 01. 2021) (siehe S. 24).

[MSpec] *Matrix Specification*. Matrix.org. 2014. URL: https://matrix.org/docs/spec/ (besucht am 06. 01. 2021) (siehe S. 50, 51).

[MSyn] *Synapse*. Matrix.org. 2020. URL: https://matrix.org/docs/projects/server/synapse (besucht am 06. 01. 2021) (siehe S. 50).

[ORP20] K.G ORPHANIDES. *Why everyone should be using Signal instead of WhatsApp*. Wired. 16. Apr. 2020. URL: https://www.wired.co.uk/article/signal-vs-whatsapp (besucht am 19. 12. 2020) (siehe S. 3, 24, 49).

[PM20] Tim Pritlove Pavel Mayer. *UKW032 Corona Weekly: Sensornetze und Wurmlöcher*. Podcast. Unsere kleine Welt. 20. Juni 2020. URL: https://ukw.fm/ukw032-corona-weekly-sensornetze-und-wurmloecher/ (besucht am 13. 01. 2021) (siehe S. 3).

[RI20] Robert Koch-Institut. *Bericht zur Datenschutz-Folgenabschätzung für die Corona-Warn-App der Bundesrepublik Deutschland*. Öffentliche Version 1.1. 16. Okt. 2020. URL: https://www.coronawarn.app/assets/documents/cwa-datenschutz-folgenabschaetzung.pdf (besucht am 07. 01. 2021) (siehe S. 61, 69).

[Rki] *Kennzahlen zur Corona-Warn-App*. Robert Koch-Institut. 8. Jan. 2021. URL: https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/WarnApp/Archiv_Kennzahlen/Kennzahlen_08012021.pdf?__blob=publicationFile (besucht am 08. 01. 2021) (siehe S. 70).

[Rud16] Tomas Rudl. *Nun amtlich: Der Messenger Signal ist ziemlich sicher*. Netzpolitik.org. 4. Okt. 2016. URL: https://netzpolitik.org/2016/nun-amtlich-der-messenger-signal-ist-ziemlich-sicher/ (besucht am 01. 01. 2021) (siehe S. 24).

[Rö20] Thomas Röbke. *Interview - Wie sicher ist die Corona-Warn-App?* Helmholtz-Gemeinschaft 2021. 16. Juni 2020. URL: https://www.helmholtz.de/technologie/wie-sicher-ist-die-corona-warn-app/ (besucht am 08. 01. 2021) (siehe S. 62).

[Sch17] Fabian A. Scherschel. *Krypto-Messenger Signal schützt Kontaktdaten vor den Server-Betreibern*. Heise online. 27. Sep. 2017. URL: https://www.heise.de/security/meldung/Krypto-Messenger-Signal-schuetzt-Kontaktdaten-vor-den-Server-Betreibern-3844545.html (besucht am 01. 01. 2021) (siehe S. 24).

[Sch20] Christian Schiffer. *Corona-Warn-App: Wie zuverlässig ist Bluetooth?* BR24. 13. Nov. 2020. URL: https://www.br.de/nachrichten/netzwelt/corona-warn-app-wie-zuverlaessig-ist-bluetooth,SGDKkBx (besucht am 08. 01. 2021) (siehe S. 70).

[Sta] *Es gibt zu viele Messenger – und einen klar besten: Signal*. Der Standard. 6. Nov. 2017. URL: https://www.derstandard.at/story/2000067258484/es-gibt-zu-viele-messenger-und-einen-klar-besten-signal (besucht am 01. 01. 2021) (siehe S. 24).

[TP21] Linus Neumann Tim Pritlove. *LNP350 Der Gästeblock muss in Quarantäne*. Logbuch:Netzpolitik. 5. Jan. 2021. URL: https://logbuch-netzpolitik.de/lnp350-der-gaesteblock-muss-in-quarantaene (besucht am 13. 01. 2021) (siehe S. 3).

[Tre19] Moritz Tremmel. *Matrix.org-Server gehackt*. Golem.de. 12. Apr. 2019. URL: https://www.golem.de/news/messenger-matrix-org-server-gehackt-1904-140655.html (besucht am 07. 01. 2021) (siehe S. 56).

[Tur16] William Turton. *Why You Should Stop Using Telegram Right Now*. Gizmodo. 24. Juni 2016. URL: https://gizmodo.com/why-you-should-stop-using-telegram-right-now-1782557415 (besucht am 26. 12. 2020) (siehe S. 23).

[Ver] *Corona-Warn-App: Fragen und Antworten zur deutschen Tracing-App*. Verbraucherzentrale Bundesverband. 30. Dez. 2020. URL: https://www.verbraucherzentrale.de/wissen/digitale-welt/apps-und-software/coronawarnapp-fragen-und-antworten-zur-deutschen-tracingapp-47466 (besucht am 08. 01. 2021) (siehe S. 62).

[WaFAQ-FB] *So arbeiten wir mit den Facebook-Unternehmen zusammen*. WhatsApp. 2020. URL: https://faq.whatsapp.com/general/security-and-privacy/how-we-work-with-the-facebook-companies?eea=1 (besucht am 04. 01. 2021) (siehe S. 49).

[WaFAQ-St] *Informationen für Strafverfolgungsbehörden*. WhatsApp. 2020. URL: https://faq.whatsapp.com/general/security-and-privacy/information-for-law-enforcement-authorities?category=5245250&lang=de (besucht am 04. 01. 2021) (siehe S. 47).

[Was] *WeChat users outside China face surveillance while training censorship algorithms*. The Washington Post. 7. Mai 2020. URL: https://www.washingtonpost.com/opinions/2020/05/07/wechat-users-outside-china-face-surveillance-while-training-censorship-algorithms/ (besucht am 02. 01. 2021) (siehe S. 24).

[Wei14] Matt Weinberger. *Matrix wants to smash the walled gardens of messaging*. Computerworld. 16. Sep. 2014. URL: https://www.computerworld.com/article/2694500/matrix-wants-to-smash-the-walled-gardens-of-messaging.html (besucht am 06. 01. 2021) (siehe S. 50).

[Wha19] Knowledge Wharton. *Your Data Is Shared and Sold... What's Being Done About It?* The Wharton School, University of Pennsylvania. 28. Okt. 2019. URL: https://knowledge.wharton.upenn.edu/article/data-shared-sold-whats-done (besucht am 27. 12. 2020) (siehe S. 6).

RFCs

[RFC1323] David A. Borman, Robert T. Braden und Van Jacobson. *TCP Extensions for High Performance*. Techn. Ber. 1323. Mai 1992. 37 S. DOI: 10.17487/RFC1323. URL: https://rfc-editor.org/rfc/rfc1323.txt (siehe S. 65, 66).

[RFC2018] Sally Floyd u. a. *TCP Selective Acknowledgment Options*. Techn. Ber. 2018. Okt. 1996. 12 S. DOI: 10.17487/RFC2018. URL: https://rfc-editor.org/rfc/rfc2018.txt (siehe S. 66).

[RFC2818] Eric Rescorla. *HTTP Over TLS*. Techn. Ber. 2818. Mai 2000. 7 S. DOI: 10.17487/RFC2818. URL: https://rfc-editor.org/rfc/rfc2818.txt (siehe S. 14).

[RFC5246] Eric Rescorla und Tim Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. Techn. Ber. 5246. Aug. 2008. 104 S. DOI: 10.17487/RFC5246. URL: https://rfc-editor.org/rfc/rfc5246.txt (siehe S. 14, 21).

[RFC5289] Eric Rescorla. *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*. Techn. Ber. 5289. Aug. 2008. 6 S. DOI: 10.17487/RFC5289. URL: https://rfc-editor.org/rfc/rfc5289.txt (siehe S. 66).

[RFC6101] Alan O. Freier, Philip Karlton und Paul C. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. Techn. Ber. 6101. Aug. 2011. 67 S. DOI: 10.17487/RFC6101. URL: https://rfc-editor.org/rfc/rfc6101.txt (siehe S. 14).

[RFC791] J. Postel. *Internet Protocol*. RFC 791. RFC Editor, Sep. 1981. URL: https://www.rfc-editor.org/info/rfc791 (siehe S. 13).

[RFC793] *Transmission Control Protocol*. RFC 793. RFC Editor, Sep. 1981. URL: https://rfc-editor.org/rfc/rfc793.txt (siehe S. 14).

[RFC8446] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Techn. Ber. 8446. Aug. 2018. 160 S. DOI: 10.17487/RFC8446. URL: https://rfc-editor.org/rfc/rfc8446.txt (siehe S. 14).

[RFC879] *The TCP Maximum Segment Size and Related Topics*. Techn. Ber. 879. Nov. 1983. 11 S. DOI: 10.17487/RFC0879. URL: https://rfc-editor.org/rfc/rfc879.txt (siehe S. 66).