HAW HAMBURG

BACHELOR THESIS
Linus Kurz

Testbed für Flow-Korrelationsangriffe auf verschlüsselte Messenger-Anwendungen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Linus Kurz

Testbed für Flow-Korrelationsangriffe auf verschlüsselte Messenger-Anwendungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Martin Becke
Zweitgutachter: Prof. Dr. Bettina Buth

Eingereicht am: 10. August 2023

Linus Kurz

Thema der Arbeit

Testbed für Flow-Korrelationsangriffe auf verschlüsselte Messenger-Anwendungen

Stichworte

Flow-Korrelationsangriff, Verschlüsselte Kommunikation, Instant Messenger, Testumgebung, Seitenkanalangriff, IT-Sicherheit

Kurzzusammenfassung

Aktuelle politische Entwicklungen verdeutlichen die Relevanz verschlüsselter und anonymer Kommunikation für die Wahrung unserer Menschenrechte. Daher sind Angriffe auf Anwendungen, die solche Dienste bereitstellen, äußerst besorgniserregend.
Aktuelle Forschungsarbeiten stellen vollständig passive Angriffe auf verschlüsselte Messenger-Anwendungen vor, die es ermöglichen, die Anonymität der kommunizierenden Parteien offenzulegen. Mithilfe von Traffic-Analyse können Nutzer*innen von Messaging-Anwendungen mit einer Erkennungsgenauigkeit von 97 % als Mitglied eines IM-Kanals identifiziert werden. Diese Entwicklungen sind äußerst bedenklich und erfordern weitere Forschung zu diesen Techniken.
Um solche Angriffe hinsichtlich ihrer Anwendbarkeit und möglichen Gegenmaßnahmen zu untersuchen, wird im Rahmen dieser Arbeit ein Testbed zur Analyse und Erforschung dieses Angriffs entwickelt.
Das entwickelte Testbed zeichnet sich durch die Integration verschiedener physischer Geräte wie Desktop-PCs und Smartphones aus, um realitätsnahe Umgebungsbedingungen zu simulieren.
Um die Verwendung in einem wissenschaftlichen Kontext zu unterstützen und aussagekräftige Ergebnisse zu erzielen, wurde das Testbed mit klarem Fokus auf Reproduzierbarkeit und Fidelität konzipiert.
Einige zentrale Technologieentscheidungen wurden zudem im Hinblick auf kostengünstige und quelloffene Nutzung getroffen. Das Testmanagementsystem basiert daher auf der quelloffenen Lösung OpenTAP.
Das Testbed erwies sich als geeignetes Instrument, um verschiedene Aspekte und Auswirkungen auf den Erfolg des Angriffs offenzulegen. Es konnte gezeigt werden, dass die Konfiguration der Endgeräte einen starken Einfluss auf die Erkennungsrate und somit den Erfolg des Angriffs hat. Ein Endgerät mit gesperrtem Bildschirm ließ sich nicht mehr konsistent als Mitglied eines Kanals identifizieren.
Der modulare Aufbau des Testbeds ermöglichte auch die Analyse veröffentlichter Erkennungsalgorithmen und Datensätze. Es wurde festgestellt, dass die Datensätze eine Inkonsistenz aufweisen, die durch die Verwendung des Testbeds verbessert werden konnte. Darüber hinaus wurde ermittelt, dass die vorliegende Version des Erkennungsalgorithmus die angegebene Genauigkeit nicht erreicht. Durch den Einsatz des Testbeds konnte die Erkennungsgenauigkeit dieser Version um 30 % verbessert werden.
Die Entwicklung dieses Testbeds soll dazu beitragen, dass solche Angriffe weiterführend untersucht werden - nicht nur, um ein größeres Bewusstsein für derartige Bedrohungen zu schaffen, sondern auch, um die Forschung im Bereich der Gegenmaßnahmen voranzutreiben.

Linus Kurz

Title of Thesis

Testbed for Flow-Correlation Attacks on Encrypted Messenger-Applications

Keywords

Flow-correlation attack, encrypted communication, instant-messenger, testbed, side-channel attack, IT-Security.

Abstract

Current political developments highlight the importance of encrypted and anonymous communication for safeguarding our human rights. Therefore, attacks on applications that provide such services are extremely concerning.

Recent research has presented fully passive attacks on encrypted messenger applications, which disclose the anonymity of the communicating parties. By utilizing traffic analysis, Instant-Messaging (IM) users can be identified as members of an IM-channel with an accuracy of 97 %.

These developments are highly alarming and call for further research into these techniques. To investigate such attacks regarding their applicability and potential countermeasures in real-world scenarios, this work develops a testbed for analysis and exploration of this attack.

The developed testbed integrates various physical devices such as desktop PCs and smartphones to simulate realistic environmental conditions.

To support its use in a scientific context and achieve meaningful results, the testbed was designed with a clear focus on reproducibility and fidelity.

Several key technology decisions were made with cost-effective and open-source usage in mind. Thus, the testmanagement system is based on the open-source solution OpenTAP.

The testbed proved to be a suitable tool for exposing various aspects and impacts on the success of the attack. It demonstrated that the configuration of the end devices strongly influences the detection rate and, consequently, the success of the attack. An end device with a locked screen could no longer be consistently identified as a member of a channel.

The modular structure of the testbed also facilitated the analysis of published detection algorithms and datasets. It was found that the datasets exhibit inconsistencies, which could be improved through the use of the testbed. Furthermore, it was determined that the current version of the detection algorithm does not achieve the stated accuracy. With the deployment of the testbed, the detection accuracy of this version could be improved by 30 %.

The development of this testbed aims to further investigate such attacks — not only to raise greater awareness of such threats but also to advance research in the field of countermeasures.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

API Application Programming Interface.

CLI Command-line interface.

DUT Device Under Test.

IM Instant-Messaging.

IMD Inter Message Delay.

IPD Inter Packet Delay.

IRST Iran Standard Time.

ISP Internet service provider.

IXP Internet exchange point.

MTU Maximum Transmission Unit.

NAT Network address translation.

NI National Instruments.

OMF Control and Management Framework.

QoE Quality of experience.

1 Einleitung

1.1 Einführung

Mit der fortschreitenden Digitalisierung unserer Kommunikation haben sich verschlüsselte Instant-Messenger wie WhatsApp, Telegram oder Signal zu unverzichtbaren Basisdiensten für private und geschäftliche Kommunikation entwickelt. Insbesondere in repressiven Staaten kommt diesen Messenger-Anwendungen eine besonders sensible Rolle zu, indem sie den Menschen eine Möglichkeit bieten, sich außerhalb staatlicher Medien zu informieren, regierungskritische Diskurse zu führen und sich in Protesten zusammenzuschließen [1].
Die Zahl derer, die von Zensur betroffen sind und empfindliche Strafen zu befürchten haben, ist nicht unerheblich. So lebten bereits im Jahr 2015, 67 % aller Menschen in Ländern, in denen kritische Stimmen der Zensur unterliegen [31]. Verschlüsselte und anonyme Kommunikation ist für viele dieser Menschen eine Lebensversicherung und die einzige Möglichkeit, ihr Recht auf freie Meinungsäußerung auszuüben [31].
In diesem Zusammenhang können IM-Anwendungen mittlerweile als eigenständiger Bestandteil politischer Prozesse betrachtet werden [46].

Aufgrund der Verbreitung und Bedeutung solcher IM-Anwendungen gewinnen auch staatliche Akteure zunehmend Interesse an der Überwachung und Kontrolle dieser Dienste.
So wurden allein im Jahre 2022, 178 Internet-Shutdowns in 35 Ländern durchgeführt um die Kommunikation und den Informationsfluss während politischer Spannungen zu unterbinden [48].
Doch nicht nur in repressiven Staaten lassen sich zahlreiche solcher Versuche beobachten. Auch in demokratisch geführten Ländern gibt es intensive Bestrebungen, die Verwendung von verschlüsselter Kommunikation gesetzlich einzuschränken.
Bekannte Beispiele sind die geplanten Gesetze zur Einführung der „Chatkontrolle“ in der Europäischen Union [32], das „Online Safety Bill“ Gesetz in England [41] und der „Stop

CSAM Act“ in den Vereinigten Staaten. Diese Gesetze können Anbieter*innen von IM-Anwendungen dazu zwingen, Nachrichteninhalte zu scannen. Das könnte einem Verbot der Ende-zu-Ende-Verschlüsselung gleichkommen [40].

Dabei wird in zahlreichen Forschungsarbeiten die essenzielle Bedeutung einer sicheren und anonymen Kommunikation für die Ausübung unserer Menschenrechte und den Erfolg demokratischer Prozesse hervorgehoben [42] [46] [31].
Bereits 2015 forderte der Menschenrechtsrat der UN in einer Resolution, dass nationale Gesetze das Recht aller Menschen auf eine verschlüsselte und anonyme Kommunikation anerkennen sollen. Anstatt die Technologien und Werkzeuge, die dies ermöglichen, einzuschränken und zu schwächen, sollten die Staaten Menschenrechtsaktivist*innen und Journalist*innen dabei unterstützen, diese Kommunikationsmittel zu nutzen [30].

> *“States should promote strong encryption and anonymity. National laws should recognize that individuals are free to protect the privacy of their digital communications by using encryption technology and tools that allow anonymity online.”*
> [30]

Solche Entwicklungen werfen wichtige Fragen hinsichtlich unserer zukünftigen Privatsphäre und Freiheit in der Kommunikation auf [42].

Daher sind Forschungsarbeiten wie [38] [10] [8] einem besonderen Fokus zu widmen, denn sie beschreiben Techniken, die trotz sicherer Verschlüsselung eine Gefahr für die Anonymität der Nutzer*innen von IM-Anwendungen darstellen.
In der Forschungsarbeit „Practical Traffic Analysis Attacks on Secure Messaging Applications“ [10][1] wird eine Technik vorgestellt, die es ermöglicht, den verschlüsselten Netzwerkverkehr von IM-Anwendungen einem Nutzer*innenaccount zuzuordnen. Der Angriff erfolgt dabei vollständig passiv, lässt sich auf verschiedene Anwendungen übertragen und ermöglicht eine Identifizierung der Endgeräte über die verwendete IP-Adresse [10].
Solche Seitenkanalangriffe sind bereits seit geraumer Zeit ein herausforderndes Forschungsthema [14] [19]. Außerhalb der Fachkreise sind solche Angriffe jedoch weitgehend unbekannt [9]. Dies verdeutlicht die Notwendigkeit für weitere Forschung und Veröffentlichungen zu diesem Thema.

[1] Die Studie wurde, leicht verändert, zwei Jahre später unter einen anderen Namen von IEEE veröffentlicht ([11]). Da die Version zu Beginn der Arbeit noch nicht zur Verfügung stand, wird im Folgenden die ursprüngliche Veröffentlichung ([10]) verwendet

Die zitierte Publikation veranschaulicht die technische Realisierbarkeit solcher Angriffsmethoden. Die verwendeten Erkennungsalgorithmen werden teils in realitätsnahen Bedingungen, teils auf synthetischen Datensätzen, aber auch unter dem Einsatz verschiedener Gegenmaßnahmen evaluiert. Mit einer Treffergenauigkeit von 97 % soll sich ein Mitglied oder Administrator einer IM-Gruppe anhand seines Traffics identifizieren lassen. Selbst die Verwendung eines VPNs reduziert die Trefferrate nur geringfügig auf 85 % [10]. Die Ergebnisse sind besorgniserregend hinsichtlich der Anonymität der Nutzer*innen und unterstreichen die Notwendigkeit für weitere Forschungen in diesem Feld.

Aussagen über die praktische Anwendbarkeit dieser Techniken oder ob sie bereits effektiv eingesetzt werden können, werden in der vorgestellten Forschungsarbeit jedoch nicht getätigt. Offene Fragen betreffen beispielsweise die Anwendbarkeit solcher Angriffe auf echte Hardwaregeräte, die Auswirkung störanfälliger Netzwerke oder den Einfluss von Anwendungskonfigurationen und Endnutzer*innenaktivitäten. Darüber hinaus gibt es aktuelle Entwicklungen im Bereich der Gegenmaßnahmen, die auf Neuronalen-Netzen basieren.
So stellt die Veröffentlichung [50] eine auf maschinellem-Lernen basierende Gegenmaßnahme für Flow-Korrelationsangriffe vor. Mithilfe eines „Adversarial-Neural-Networks“ wird durch das gezielte Einfügen von Paketen eine hohe Schutzrate von 97 % bei einem verhältnismäßig geringen Traffic-Overhead von nur 20 % erreicht.

Um solche Faktoren und Gegenmaßnahmen gezielt und kontrolliert zu untersuchen, eignen sich Testumgebungen. Sie ermöglichen durch das Simulieren und Bereitstellen, der zur Untersuchung des Systems benötigten Komponenten, die gezielte Analyse einzelner Systemaspekte [22]. Auf Basis der dadurch erzeugten Daten lassen sich realitätsnahe Abschätzungen über die Effektivität oder Effizienz eines echten Systems treffen [3].

In der Forschung sind Testbeds ein beliebtes Instrument und es existieren einige namhafte Testumgebungen wie das „Triangle-Projekt“ [20] [37], GENI [24], OMF [39] oder die „ATAACK Cloud“ [45].
Diese großskalierten Testumgebungen sind mit teurer, aufwändiger Hardware ausgestattet und weisen daher oft einen restriktierten Zugriff auf. Häufig sind sie domänenübergreifend konzipiert und ermöglichen eine vielfältige Nutzung. Damit fehlt es ihnen jedoch typischerweise an Spezifität.
Aufgrund der hohen Entwicklungs- und Bereitstellungskosten werden solche Testbeds zudem oft von Interessensgruppen wie dem Militär oder einzelnen Industriezweigen gefördert [45] [20].

Neben diesen gibt es Testbeds im IoT-Bereich, die auf Angriffsszenarien für IoT-Geräte konzipiert sind und ähnliche Anforderungen an die Infrastruktur aufweisen. In den Studien [43] und [44] wird ein Ansatz verfolgt, der auf die Verwendung von kostengünstiger und leicht zugänglicher Hardware abzielt. In diesen Forschungsarbeiten werden die Testbeds jedoch zur Untersuchung klassischer Sicherheitslücken entwickelt, indem verschiedene Werkzeuge zur Analyse bekannter Schwachstellen in den Testumgebungen bereitgestellt werden.

Oftmals entstehen Testbeds jedoch als kleine, spezifische Testumgebungen im Kontext eines Forschungsprojekts. Sie dienen häufig bis zur Veröffentlichung der Ergebnisse. Im Anschluss werden diese aber in den seltensten Fällen gepflegt, weitergenutzt oder überhaupt veröffentlicht. Dieses Vorgehen erschwert eine unabhängige Validierung der Ergebnisse und damit auch die weiterführende Forschung [39].

Im Rahmen dieser Arbeit wird daher ein Testbed entwickelt, das es ermöglicht, die Auswirkungen und Rahmenbedingungen des vorgestellten Angriffsszenarios in einer realitätsnahen Umgebung zu untersuchen. Es soll flexibel verschiedene Netzwerkumgebungen (Wi-Fi und Ethernet) und physische Geräte unterstützen, um die Auswirkungen von Endgerät- und Netzwerkkonfigurationen auf die Anwendbarkeit des vorgestellten Angriffs analysieren zu können. Mit der Entwicklung wird beabsichtigt, auf den Ressourcen vorhandener Forschung aufzubauen, indem bestehende Datensätze und Algorithmen in das Testumfeld integriert werden können. Zur Unterstützung zukünftiger Forschungsarbeiten ist zudem die Erzeugung neuer Datensätze vorgesehen.
Im Gegensatz zu den bisherigen Ansätzen wird das Testbed auf frei verfügbaren Software Komponenten basieren, wodurch keine speziellen Fördermittel für die Entwicklung oder den Betrieb erforderlich sind.

Im Hinblick auf die wissenschaftliche Verwertbarkeit und Glaubwürdigkeit des Testbeds, wird in der Entwicklung ein besonderer Fokus auf die Gewährleistung von Fidelität und Reproduzierbarkeit gelegt. Diese beiden Qualitätsziele spielen eine zentrale Rolle, um sicherzustellen, dass die durchgeführten Experimente und Studien zuverlässig und konsistent sind und von anderen Forschenden unabhängig reproduziert werden können [3].
Die abschließende Evaluation des Testbeds erfolgt anhand dieser Kriterien, um sicherzustellen, dass die erzielten Ergebnisse valide und reproduzierbar sind.

1.2 Verwandte Arbeiten

Dieser Abschnitt gibt einen detaillierteren einblick in verwandte Arbeiten, die Technologien oder Architekturen nutzen, welche für den Verlauf dieser Ausarbeitung von Interesse sein können.

Die Forschungsarbeit von Bahramali et al. [10] dient als Orientierung für die Entwicklung dieses Testbeds. Die Autor*innen demonstrieren erfolgreich einen Angriff auf IM-Anwendungen durch die Analyse verschlüsselter Netzwerkdaten. Der Schwerpunkt liegt dabei auf der Entwicklung und Evaluation von Erkennungsalgorithmen. Die verwendeten Datensätze werden teilweise synthetisch generiert, teilweise basieren sie auf beobachteten Benutzer*innenaktivitäten.
Es wird gezeigt, dass der Angriff auf verschiedene Anwendungen übertragbar ist. Zur Demonstration werden die Endgeräte durch synthetische (Command-line interface (CLI)- oder Application Programming Interface (API)-basierte) Implementierungen simuliert. Nicht untersucht wird hingegen der Einfluss von physischen Endgeräten sowie Aspekte, die den Zustand der Anwendung oder Firmware-Einstellungen betreffen. Zudem fehlt eine klare Dokumentation der Umgebungsbedingungen, auf denen die veröffentlichten Datensätze und Ergebnisse basieren.

Unter ähnlichen Bedingungen stellen die Autor*innen von „Inferring user activities on KakaoTalk with traffic analysis“ [38] einen Flow-Korrelationsangriff auf den Messenger *KakaoTalk* vor.
Das Ziel dieser Arbeit bestand darin, die in der App durchgeführten Anwendungsinteraktionen (z.B. Senden einer Nachricht oder beitreten und blockieren eines Chats) mithilfe eines Random-Forest-Klassifikators zu erkennen. Die Forschenden nutzten einen als Network address translation (NAT)-Gateway konfigurierten PC, um vollständig passiv, den Netzwerkverkehr von *KakaoTalk* zu analysieren.
Innerhalb des beschriebenen Aufbaus wird eine Erkennungsrate von 99,7 % erreicht.
Für das Training und die Erkennung wird allerdings simulierter Netzwerkverkehr anstelle von echten Anwendungsaktivitäten verwendet. Es wird darauf hingewiesen, dass echte Aktivitäten, zu sehr unterschiedlichen Ergebnissen führen können. Darüber hinaus werden keine anderen Arten von Nachrichten wie Bilder, Texte oder Emoticons berücksichtigt [38].

Die vorliegenden Forschungsarbeiten [8] und [7], stellen eine Methode zur passiven Beobachtung von verschlüsseltem Wi-Fi-Verkehr vor, um Skype-Sprachverkehr zu erkennen

und zu identifizieren. Die Autor*innen nutzen einen kostengünstigen Testaufbau aus handelsüblichen Komponenten, der einen Raspberry-Pi mit einem Wi-Fi-Adapter umfasst. Dieser Aufbau ist für weniger als 100 Euro erhältlich und ermöglicht eine Erkennungsgenauigkeit von 97 % bei einer Falsch-Positiv-Rate (FPR) von lediglich 3 %. Die Autor*innen stellen fest, dass dieser Angriff auch auf andere Anwendungen und Endgeräte übertragen werden kann, wobei der Fokus auf der passiven Beobachtung des verschlüsselten Funknetzwerkverkehrs liegt.
Es wird jedoch nicht berücksichtigt, ob der Testaufbau in der Lage ist, den Netzwerkverkehr von IM-Anwendungen zu untersuchen. Zudem wird eine Technik verwendet die den Netzwerkverkehr durch gezielt eingefügte Pakete markiert, um beschriftete Daten für das Training der Algorithmen zu generieren.

Kostengünstige IoT-Testumgebungen

In anderen Veröffentlichungen wird die Entwicklung von leichtgewichtigen Testumgebungen für die Sicherheitsforschung an IoT-Geräten behandelt. Obwohl diese Umgebungen für einen anderen Kontext entwickelt wurden, weisen die verwendeten Testaufbauten Ähnlichkeiten zu den Testumgebungen für Flow-Korrelationsangriffe auf.

Die Veröffentlichung [44] beschreibt ein Testbed zur Untersuchung klassischer Sicherheitslücken und Datenschutzaspekte von IoT-Geräten. Der vorgestellte kostengünstige Testaufbau ermöglicht die Analyse des Netzwerkverkehrs von IoT-Geräten wie Smartwatches oder Drohnen.
Dabei werden leicht verfügbare Hardwarekomponenten und Open-Source-Software eingesetzt, um verschiedene Geräte über Bluetooth oder Wi-Fi einzubinden. Auch wenn der Aufbau ähnlichen Rahmenbedingungen entspricht, liegt der Fokus auf der Untersuchung von IoT-Geräten und der Analyse klassischer Sicherheitslücken, was einen anderen Anwendungsbereich aufzeigt. Instrumente und Komponente welche die Untersuchung klassischer IM-Anwendung ermöglichen werden nicht beschrieben.

Einen anderen Ansatz verfolgt die Forschungsarbeit [2]. Ebenfalls mit dem Ziel kostengünstige und leicht verfügbare Hardware zu verwenden, wird ein „Experiment-as-a-Service“ Modell [23] vorgestellt. Die Testumgebung wurde entwickelt, um Studierenden eine Lernumgebung zu bieten, in der sie mit einfachen Mitteln an IoT-ähnlichen Geräten experimentieren können. Dabei sind reale Anwendungen wie IM-Apps auf Smartphones

nicht untersuchbar. Zudem stellt eine Lernumgebung völlig andere Anforderung an die Fidelität und Reproduzierbarkeit der Ergebnisse.

Die Studie [43] stellt ein weiteres Testbed zur Sicherheitsforschung von IoT-Geräten vor. Die aufwändige Architektur zielt darauf ab, einen hochgradig modularen Aufbau zu schaffen, der verschiedene Sicherheitsanalysen und Angriffe simulieren kann. Dadurch sollen beliebige IoT-Geräte und Simulationskomponenten integriert werden können, um komplexe Sicherheitstests durchzuführen.
Obwohl der Aufbau und die Architektur grundsätzlich auch für dieses Angriffsszenario geeignet sein könnte, fehlt es aufgrund der generalisierten und domänenübergreifenden Ausrichtung an spezifischen Werkzeugen und Instrumenten, um eine detaillierte Analyse des vorgestellten Flow-Korrelationsangriffs durchzuführen. Zudem werden für zentrale Komponenten wie das Testmanagement, die sehr kostenintensive und proprietäre Software *NI-Teststand* verwendet.

Großskalierte Testumgebungen

Neben den erwähnten Studien, welche kleine und kostengünstige Testaufbauten präsentierten, existieren auch großskalierte Testumgebungen.

Im Rahmen des TRIANGLE-Projekt wurde ein groß angelegtes Testbed entwickelt, mit dem Ziel, die Quality of experience (QoE) von mobilen Endgeräten, Anwendungen und Netzwerken reproduzierbar zu evaluieren. Teile der Entwicklung wurden in den Veröffentlichungen [20] und [37] beschrieben. Unter anderem wurde das Testbed für die Untersuchung des 5G Netzes gefördert. Eine große Anzahl emulierter und physischer Geräte werden bereitgestellt, um domänenübergreifende Tests durchzuführen.
Beschrieben wird eine aufwändige Architektur, die auf dem Test und Automatisierungs Framework TAP basiert. Obwohl die vorgestellte Architektur einige inspirierende Aspekte bietet und das Testbed als Infrastruktur für eine Evaluation des Angriffsszenarios geeignet sein könnte, fehlt es an Detailtiefe und den nötigen spezifischen Komponenten um die angestrebten Untersuchungen durchzuführen. Darüber hinaus bedeuten die hohen Entwicklungskosten von über 3,5 Millionen Euro und die damit verbundene Einschränkungen bei der Nutzung, dass das Testbed nur einer sehr ausgewählten Forschungsgemeinschaft zugänglich ist.

Als Teil der „ATAACK Cloud“ wurde ein Cloud-basiertes Testbed entwickelt [45]. Dieses Testbed bietet Zugriff auf über 1000 Geräte, um die dynamischen Anforderungen in

Bezug auf Orts- und Kontextwechsel zu untersuchen, denen unsere heutigen Kommunikationssysteme ausgesetzt sind.
Die Infrastruktur besteht unter anderen aus 34 Blade-Servern, sowie weiterer kostenintensiver Hardware [27]. Zur Finanzierung der Betriebs- und Entwicklungskosten wurde das Projekt durch das amerikanische Militär gefördert.

Diese Tatsache, sowie die bereits beschriebene Abgrenzung zu großskalierten Testsystemen wie [24] [16] [20], verdeutlichen auch in dieser Kategorie, eine Lücke in der Forschung bezüglich der Entwicklung eines Testbeds zur Untersuchung von Flow-Korrelationsangriffen auf verschlüsselte IM-Anwendungen. Es existieren bisher keine umfassenden wissenschaftlichen Arbeiten, die sich eingehend mit diesem Thema befassen.

Zusammenfassend

Zusammenfassend lässt sich feststellen, dass in den untersuchten Arbeiten verschiedene Testumgebungen zur Erforschung von Sicherheitslücken und Angriffsszenarien verwendet wurden. Während einige dieser Testbeds auf kostengünstiger Hardware und Open-Source-Software basieren, konzentrieren sich andere auf groß angelegte, domänenübergreifende Testaufbauten.

Das in dieser Arbeit entwickelte Testbed, unterscheidet sich von den vorgestellten Forschungsarbeiten in mehreren Aspekten. Es zielt darauf ab, Flow-Korrelationsangriffe auf IM-Anwendungen zu untersuchen um die Auswirkungen von Hardwarekonfigurationen sowie spezifischen Anwendungs- und Firmware-Einstellungen zu analysieren.
Das Testbed ermöglicht die Analyse von Netzwerkverkehr, unter Verwendung realitätsnaher Umgebungsdaten und Benutzer*innenaktivitäten, um die Anwendbarkeit des Angriffs durch bestehende Erkennungsalgorithmen zu evaluieren. Im Gegensatz zu den bisherigen Arbeiten wurde in dieser Studie besonderer Wert auf die Reproduzierbarkeit und Fidelität der Ergebnisse gelegt, wobei die Verwendung kostengünstiger und quelloffener Lösungen angestrebt wird.

Die vorliegenden verwandten Arbeiten verdeutlichen, dass in der Forschung zu diesem Thema noch erhebliche Lücken bestehen. Daher kann die Entwicklung dieses Testbeds einen Beitrag zum aktuellen Forschungsstand leisten.

1.3 Kapitelübersicht

Das Grundlagen-Kapitel gibt zunächst einen Überblick über die theoretischen Grundkonzepte, die für das Verständnis der nachfolgenden Ausführungen relevant sind. Es werden die Hintergründe von Flow-Korrelationsangriffen und Testbeds eingehend erläutert, und einen einblick in den Stand der Forschung auf diesen Gebieten ermöglicht.

Im Kapitel Analyse und Design wird zunächst das beschriebene Ziel der Arbeit näher betrachtet. Basierend auf den Anforderungen wird anschließend die Systemarchitektur entworfen und wichtige Komponenten im Detail konzipiert.

Das Implementationskapitel stellt dar, wie die im Design-Kapitel entwickelten Lösungen in Programmcode umgesetzt werden. Es bietet zudem einen Einblick in Test- und Integrationsprozess

Im Anschluss werden in der Evaluierung die implementierten Lösungen anhand praxisnaher Anwendungsszenarien getestet.

In der übergreifenden Diskussion werden die im Evaluierungskapitel generierten Ergebnisse eingehend analysiert und in einen erweiterten Kontext eingeordnet. Anhand der zu Beginn definierten Qualitätskriterien erfolgt eine systematische Bewertung des entwickelten Testbeds. Dabei werden mögliche Erweiterungen und Verbesserungen aufgezeigt.

Abschließend gibt das Fazit eine Zusammenfassung der wichtigsten Erkenntnisse und einen Ausblick auf weiterführende Arbeiten in diesem Bereich.

2 Grundlagen

2.1 Funktionsweise von Flow-Korrelationsangriffen

Dieses Kapitel dient der Einführung in die Hintergründe und technischen Grundlagen von Flow-Korrelationsangriffen („Flow-Correlation Attacks“), auf denen die folgenden Kapitel aufbauen.

2.1.1 Hintergrund

Schon seit vielen Jahren wird an Verschlüsselung geforscht, um sichere Kommunikation unter den Gesichtspunkten der Vertraulichkeit, Authentizität und Integrität zu ermöglichen [9]. Diese hat zur Entwicklung einer Reihe an effizienten und effektiven Protokollen geführt, die mittlerweile als Mindeststandard empfohlen, erfolgreich eingesetzt werden. [12]

Auch das Bewusstsein für die Bedeutung und Notwendigkeit von Verschlüsselung, gerade im Umfeld von autoritär geführten Staaten und in Bezug auf unabhängige Berichterstattung und Journalismus, ist in Fachkreisen präsent [30] [46] [31].

Dass der erfolgreiche Einsatz von sicherer Verschlüsselung aber nicht gleichzeitig die Aktivitäten und damit auch die Identität der User verbirgt, ist außerhalb von Forschungskreisen größtenteils unbekannt [8].
In bestimmten Situationen kann z.B. die bloße Beobachtung des Nachrichtenaustauschs zwischen zwei Kommunikationsteilnehmer*innen genügen, um sensible und belastbare Informationen zu erhalten [9]. Die Anonymität der kommunizierenden Parteien ist daher eine weitere, sehr wichtige Eigenschaft die es zu schützen gilt [51]. Dieser Aspekt erhält in der Praxis jedoch nur wenig Aufmerksamkeit [9].

Der folgende Teil gibt daher einen Überblick über einige Angriffsarten, um anschließend konkreter auf die vorgestellten Algorithmen eingehen zu können, für die das Testbed entwickelt werden soll.

2.1.2 Übersicht

Traffic Analyse ist ein Feld der Kryptographie, welches in der frei zugänglichen Literatur wenig Beachtung findet [9]. Dennoch gibt es mittlerweile eine Reihe an Forschungsarbeiten, die ein grundsätzliches Problem beim Einsatz von Verschlüsselung zeigen [10].

Traffic Analyse ist ein Seitenkanalangriff, der es ermöglicht, vertrauliche Informationen zu erhalten, ohne Verschlüsselungen zu brechen oder Sicherheitslücken auszunutzen [10]. Die Angriffsmethoden können dabei in passive und aktive unterteilt werden [26]:

Aktive Traffic-Analyse Angriffe, wie beispielsweise *Flow Watermarking* [29], nutzen aktive Sondierungswerkzeuge um Informationen über das Ziel zu gewinnen. So können spezifische, aber unauffällige Verkehrsmuster generiert werden (durch z.B. verzögern von Paketen), die eine Verfolgung des Nachrichtenverkehrs über Netzwerkknoten hinweg, bis zum Ziel ermöglichen [26] [10].
Ein weiteres Verfahren ermöglicht es, durch statistische Analysen der Round-Trip-Time (RTT) mithilfe von Ping, Statusinformationen über die gesendeten Nutzdaten zu erhalten [26].
Solche aktive Angriffe lassen sich leichter aufsetzen, können allerdings auch einfacher erkannt werden.

Passive Traffic-Analyse Angriffe sind in der Regel reine Lauschangriffe, bei denen zunächst (verschlüsselter) Netzwerkverkehr abgehört wird. Anschließende Analysen ermöglichen daraufhin Rückschlüsse z.B. auf die verwendete Anwendung [7] [21], die Nutzungszeiten oder durchgeführten Aktivitäten [38], die aufgerufene Website [28], und können selbst in Anonymisierungsnetzwerken wie *TOR* Rückschlüsse auf die Nutzeridentität zulassen [33].

Eine Unterform dieser Passiven Angriffe ist die Flow-Korrelation.
Flow-Korrelation ist eine Technik, bei der der Angreifer versucht, verschlüsselte Netzwerkflüsse miteinander zu verknüpfen, indem er die Verkehrsmerkmale (d.h. die Ankunftszeiten und Größen von Paketen) untersucht. Flow-Korrelation wurde insbesondere als Angriff auf Anonymitätssysteme wie *TOR* untersucht.

Ein bekanntes Beispiel hierfür ist der Einsatz von *DeepCorr* [33]. DeepCorr nutzt ein Convolutional-Neural-Network (CNN), um Netzwerkströme zwischen den Eintritts- und Austrittsknoten des Tor-Netzwerks zuzuordnen und somit Anfragen rückverfolgbar zu machen.
Die für das Training verwendeten Merkmale bestehen aus dem Inter Packet Delay (IPD) und der Paketgröße, jeweils für eingehende und ausgehende Pakete. Das trainierte Netzwerk erreicht eine Genauigkeit von 96 %. In einer aktuellen Forschungsarbeit wurde zudem ein Flow-Korrelationsangriff auf IM-Anwendungen wie Telegram & WhatsApp vorgestellt [10].
Der folgende Teil soll einen detaillierteren Einblick in den Ablauf und Aufbau der dort verwendeten Technik liefern.

2.1.3 Funktionsweise

In dem Forschungsarbeit *„Practical Traffic Analysis Attacks on Secure Messaging Applications“* [10] wird eine der Flow-Korrelation sehr ähnlichen Technik angewendet: Die angreifende Partei (im folgenden: Eve) schneidet zunächst die Aktivitäten in einem Zielkanal mit (z.B. im Falle einer öffentlichen Messenger-Gruppe, indem Eve dieser Beitritt). Diese Aktivitäten werden anschließend versucht den Verkehrsmustern zuzuordnen, die an anderen Stellen im Netzwerk aufgezeichnet wurden, um die IP-Adressen der Mitglieder oder Administratoren des Zielkanals zu identifizieren. Der Ablauf lässt sich durch die folgenden Schritte beschreiben.

- **Ground-Truth:**
  Für jeden zu beobachtenden Zielkanal des Messengers muss Eve, „Ground-Truth“, also tatsächlich stattgefundene Ereignisse, über den Netzwerkverkehr sammeln. Dafür gibt es verschiedene Wege. Ist der Zielkanal des IM-Service öffentlich, kann Eve diesem Beitreten und dort gesendete Nachrichten, zusammen mit anfallenden Metadaten (Zeitpunkt, Typ der Nachricht, Größe), mitschneiden. Dies entspricht dem Flow A
- **Überwachen des Netzwerkverkehrs:**
  Zusätzlich wird der verschlüsselte Netzwerkverkehr von IM-Nutzer*innen überwacht (Flow B), um die Mitglieder oder Administratoren des Zielkanals anhand ihrer IP-Adressen zu identifizieren.

Das Mitschneiden des Traffics kann durch die ISPs oder IXPs erfolgen, wenn diese Eve's Kontrolle unterliegen.

- **Erkennung:**
  Eve nutzt im Anschluss einen Erkennungsalgorithmus, um den aufgezeichneten Flow A aus dem Zielkanal mit den Verkehrsmustern aus Flow B zu vergleichen.

2.1.4 Erkennungsalgorithmen

Für die Erkennung (Vergleich des Ground-Truth Nachrichten zu den beobachteten Verkehrsmustern) lassen sich klassische statistische Metriken wie die Pearson Korrelation, Kosinus-Ähnlichkeit, oder Spearman Korrelation einsetzen.
Ein anderer Ansatz ist die Verwendung von Machine Learning Verfahren, wie sie zur Klassifikation bei [33, DeepCorr] eingesetzt wurden. Andere wie [38] setzen zur Korrelation von Messenger Aktivitäten auf einen Random-Forrest-Classifier während [18] den Naive-Bayes-Classifier für die Analyse von „IMessage“ Nachrichten verwendeten.
Der im folgenden detaillierter vorgestellte *„Event-Based-Algorithm“*, wurde speziell für den Kontext von IM-Kommunikation entwickelt und nutzt „Hypothesis-Testing“.
Im Gegensatz zu anderen Flow-Korrelationsangriffen, die in beiden „Flows“ die Eigenschaften von Netzwerkpaketen verwenden, werden hier für den Ground-Truth, die Merkmale (Timing und Größen) von IM-Nachrichten zur Erkennung genutzt [10].

Event-Based-Algorithm

Jede der im Schritt 1 beobachteten Ground-Truth Nachrichten (z.B. ein versendetes Bild) erzeugt einen Burst (eine Reihe dicht aufeinander folgende Pakete) auf dem Netzwerkmedium. Die Größe der Pakete entspricht dabei annähernd der verwendeten Maximum Transmission Unit (MTU). Dieser Burst lässt sich bei korrelierenden Flows als verschlüsselter Verkehr auf dem überwachten Netzwerkknoten beobachten [10].

- **Event-Extraction:**
  Im ersten Schritt, dem Extraktionsprozess, werden daher aus dem beobachteten Netzwerkflow einzelne IM-Events extrahiert. Ein IM-Event besteht dabei aus dem Zeitpunkt und der Größe des beobachteten Burst. Das letzte Paket bestimmt dabei den Zeitpunkt, während die Größen der Pakete aufsummiert werden. Bei kleineren

und vereinzelt auftretenden Paketen hingegen handelt es sich um Protokollnachrichten, die für die Erkennung ignoriert werden [10].

- **Korrelations-Funktion:**
  Zur Korrelation werden zwei Hypothesen formuliert:
  - H0: Das beobachtete Endgerät hat keine Verbindung (Mitglied oder Admin) zu dem beobachteten Kanal. Damit besteht auch keine Korrelation zwischen den Ground-Truth Nachrichten und den Netzwerkflüssen.
  - H1: Das Endgerät ist Mitglied oder Administrator des Zielkanals und erzeugt daher ähnliche Verkehrsmuster.
- **Erkennungs-Algorithmus:** Um die auftretenden Netzwerkflüsse einzuordnen, werden die extrahierten SIM-Events mit den beobachteten Ereignissen des Kanals verglichen. Ein Treffer wird identifiziert, wenn die Abweichung der Größe und der Zeitstempel jeweils innerhalb eines Schwellenwerts liegt, was bedeutet, dass die beobachteten Ereignisse ähnliche Größen aufweisen und zeitlich eng beieinander liegen.
  Dabei können Netzwerkeigenschaften wie Jitter und Bandbreite, Einfluss auf die Extraktion und Erkennung haben.
  Anhand der Anzahl der Treffer lässt sich anschließend eine Trefferrate ableiten, die mit einem Schwellenwert verglichen wird, um eine der Hypothesen zu bestätigen [10].

Shape-Based Algorithmus

Ein weiterer vielversprechender Algorithmus ist der Shape-Based Algorithmus, der ähnlich dem Event-Based Algorithmus funktioniert.
Zunächst wird die bereits beschriebene Event-Extraction 2.1.4 durchgeführt. Die extrahierten Events werden im nächsten Schritt in normalisierte Vektoren umgewandelt. Diese Vektoren beschreiben die Form des Traffics und verwenden die Größe und die Zeitstempel des Events als Parameter.
Eine normalisierte Korrelationsmetrik bestimmt im Anschluss die Ähnlichkeit der beiden Traffic-Formen und ermöglicht damit das Bestätigen einer der Hypothesen.
Durch die Normalisierung und Überführung in Vektoren wird der Einfluss der Bandbreite entfernt und das Rauschverhalten verbessert.

Beide Algorithmen erreichen schon nach kurzen Beobachtungszeiträumen, sehr hohe Trefferquoten. Mit Richtig-Positiven (TP) von 0.94 und Falsch-Positiven (FP) von 10^{-3}.
Im direkten Vergleich zeigt sich, dass der Event-Based-Algorithm in der Ausführung doppelt so schnell ist, während der Shape-Based-Algorithm bessere FP-Werte erreicht.

2.1.5 Bedeutung und Auswirkungen

Ein wichtiger Aspekt ist, dass die vorgestellten Angriffe nicht durch die Ausnutzung von Sicherheitslücken oder Bugs möglich gemacht werden. Vielmehr wird eine fehlende Verschleierung der Verkehrscharakteristiken ausgenutzt.
Spätestens seit den 70ern sind solche Seitenkanalangriffe eine bekannte Bedrohung für die Vertraulichkeit von Systemen, und beschreiben damit ein fundamentales Problem [7].
So können die beschriebenen Angriffstechniken nicht nur auf klassische IM-Dienste angewendet werden. Flow-Korrelationsangriffe stellen eine Bedrohung für die Vertraulichkeit aller interaktiven Kommunikationssysteme dar, die eine anonyme Kommunikation anstreben [19].
Die Angriffe können ausgenutzt werden, um die identifizierten Ziele zu verfolgen und erlauben das gezielte Blockieren von Diensten oder kontroversen Nachrichten Kanälen [10].

2.1.6 Gegenmaßnahmen

Mit dem Bekanntwerden dieser Angriffe wurde auch an Gegenmaßnahmen geforscht, die später in sogenannten „Mix Networks“ Anwendung fanden. [14]

Eine einfache Gegenmaßnahme ist das „Constant-Link-Padding“ [38]. Dabei werden zufällige Füllbytes den Paketen angefügt um diese auf eine feste Größe (z.B. die größte beobachtete Paketlänge) zu fixieren.
So lässt sich die Verbindung der Paketgrößen zu den User-Interaktionen vollständig auflösen [18].
Dies erzeugt allerdings unvermeidbaren Traffic-Overhead. So kann das „Constant-Link-Padding“ einen Overhead von 300 % erzeugen, was diese Gegenmaßnahme in Produktiven und wettbewerbsfähigen Systemen schwer einsetzbar macht [38].
„Traffic Shaping“ ist eine andere Methode die verwendet werden kann, um der zeitlichen Korrelation entgegenzuwirken. Dabei werden Pakete zusammengefasst, verzögert oder

außerhalb der Reihenfolge versendet. Auch zusätzliche Dummy Pakete können die Form des Traffics zu verschleiern [9].

Jüngste Veröffentlichungen verwenden Adverserial-Networks, um Deep-Learning basierte Flow-Korrelationsangriffe zu kontern [50].
All diese Gegenmaßnahmen erzeugen Traffic Overhead und bringen Latenz Konsequenzen mit sich. Und auch wenn in Systemen wie TOR solche Techniken eingesetzt werden können[1], so lässt sich grundlegend schließen, dass mit keiner der bestehenden Methoden, eine hohe Schutzrate gegen Flow-Korrelationsangriffe, bei gleichzeitig geringem und akzeptablen Overhead erreichen lässt.

Die zentralen Methoden für effiziente Netzwerkübertragung: nur so viele Daten wie nötig, und nur so selten wie möglich, zu übertragen, steht diesen Verfahren entgegen.
Global einsetzbaren Gegenmaßnahmen, die eine sowohl effiziente als auch sichere Kommunikation ermöglichen, sind daher eine bestehende und ungelöste Herausforderung [7].

2.1.7 Abschließend

In diesem Abschnitt wurden Flow-Korrelationsangriffe eingeführt, die eine Bedrohung für die Anonymität in interaktiven Kommunikationssystemen darstellen. Die vorgestellten Erkennungsalgorithmen zeigen hohe Trefferquoten, während Gegenmaßnahmen oft mit Traffic-Overhead und Latenzkonsequenzen einhergehen. Die Entwicklung effizienter und global einsetzbarer Schutzmechanismen, die die Anonymität wahren, bleibt eine Herausforderung.

[1] https://gitlab.com/yawning/obfs4

2.2 Testbeds

Im folgenden Abschnitt wird das Thema Testumgebungen eingeführt und ihre Bedeutung sowie ihre Verwendung in Forschung und Entwicklung näher erläutert.

2.2.1 Hintergrund

Unter dem Begriff Testbed vereint, finden sich unterschiedlichste Test-, Trainings und Demonstrationsaufbauten [22].
Sie alle stellen eine Umgebung bereit, die es ermöglicht, ausgewählte Systemaspekte unter kontrollierten Bedingungen zu testen.
Besonders für die Forschung im IT-Sicherheits Kontext sind Testbeds ein essenzieller Bestandteil [3], denn dort ist es selten möglich, in einem produktiven- oder Live-System die angestrebten Experimente und Szenarien durchzuführen [22].
Zum einen können Richtlinien und Gesetze dies verbieten, zum anderen braucht es teilweise auch aus sicherheitstechnischer Sicht, abgeschottete Systeme, um die Auswirkungen der Tests zu begrenzen [22].
Für die Glaubwürdigkeit und Reproduzierbarkeit sind kontrollierbare Systeme zudem notwendig, um die zu untersuchenden Aspekte und Größen von nicht beeinflussbaren Faktoren zu isolieren. So lassen sich Abhängigkeiten und Auswirkungen genauer untersuchen [22].

Testbeds können damit eine Realitätsnahe Hard- & Software Umgebung darstellen, die es ermöglicht, Daten und Erkenntnisse über ein System zu sammeln, welches so noch nicht existiert.
Die gewonnenen Erkenntnisse können so als Basis für weitere Forschung und Design-Entscheidungen dienen [25].

Um einen Überblick über die Bestandteile und Eigenschaften eines Testbeds zu geben, folgt nun eine detaillierte Betrachtung.

2.2.2 Bestandteile & Entwicklung

Testbeds können verschiedene Bereiche eines Systems umfassen [25]. Häufig konzentrieren sie sich jedoch auf den Teil des Systems, der für die Untersuchung der ausgewählten

Systemaspekte relevant ist.Um diese Aspekte testbar zu machen, werden Prototypen eingesetzt, die mit den zu testenden Komponenten interagieren [39].

Die Hauptbestandteile eines Testbeds lassen sich wie folgt beschreiben [25]:

- **Experimentelles-Teilsystem**
  Dieses Teilsystem umfasst Komponenten wie Prototypen und ausgewählte reale Teile des Systems, die überwacht, kontrolliert und getestet werden sollen.
- **Überwachungs-Teilsystem**
  Das Überwachen und Auslesen von anfallenden Daten aus dem Experiment, findet im Überwachungs-Teilsystem statt und enthält dafür die nötigen Schnittstellen zum experimentellen-Teilsystem.
- **Simulations-Stimulations-Teilsystem**
  Um dem experimentellen-Teilsystem einen möglichst realitätsnahen Kontext zu bieten, können die Simulations- und Stimulationselemente auf Ereignisse reagieren und die erforderlichen Daten und Stimuli in die Experimentierkomponenten einspeisen.

Herausfordernd bei der Entwicklung, kann das Eingrenzen und Analysieren der zu untersuchenden Systemaspekte sein.
Ebenfalls aufwendig kann die Auswahl von passenden Eingabestimuli und die Definition der Schnittstellen zum experimentellen Teilsystem sein. Diese werden benötigt, um eine möglichst realitätsnahe Abbildung des Kontexts im Simulations-Stimulations-System zu erreichen, und bestimmen maßgeblich die Genauigkeit der Ergebnisse [25].

2.2.3 Attribute und Designziele

In der Literatur werden mehrere zentrale Eigenschaften von Testumgebungen identifiziert, die im Designprozess dazu beitragen können, einen strukturierten Ansatz zu verfolgen. Die Beschreibung von Testumgebungen kann anhand ihrer Attribute in den folgenden Bereichen erfolgen [22]:

Ressourcen

Die zur Verfügung stehenden Ressourcen bestimmen das Anwendungsgebiet und den Kontext, in dem das Testbed eingesetzt werden kann und können folgende Punkte beinhalten:

- Unterstützte Hardware und Architekturen.
- Einsetzbare und vorhandene Anwendungen.
- Vorhandenes Know-how, Erfahrung und Support durch Expert*innen.

Ressourcenmanagement

Das Aufsetzen und Konfigurieren aller infrastrukturellen Komponenten wird durch das Ressourcenmanagement bestimmt. Der Automatisierungsgrad ist ein fließender Übergang von vollautomatisiert über hybrid bis hin zu einem vollständig manuellen Setup.

Konfiguration und Initialisierung

Nach dem Setup der Infrastruktur werden die restlichen Komponenten initialisiert und aufgesetzt. Das bedeutet zum Beispiel die Initialisierung des Testequipments, das Aufsetzen von Benutzer*innen-konten und die Konfiguration von Firmware und Testanwendungen. Der Automatisierungsgrad kann hier ebenfalls variieren.

Experiment-Design

Der Ablauf des Experiments sowie die Konfigurationsschritte können durch den Nutzer beschrieben und definiert werden. Dies geschieht im Experiment-Design, welches fertige Ablaufpläne oder auch eine Beschreibungssprache bereitstellen kann, um nötige Setups und Konfigurationsschritte (teil)automatisiert durchzuführen. Die Wiederholbarkeit des Testbeds wird maßgeblich durch dieses Attribut bestimmt.

Experiment-Steuerung

Dieses Attribut beschreibt die Fähigkeiten des Testbeds, auf das laufende Experiment Einfluss zu nehmen. Dies kann zum Beispiel durch den Zugriff auf ein Gerät erfolgen, indem Umgebungsereignisse oder Nutzerverhalten simuliert werden oder Fremdverkehr im Netzwerk injiziert wird.

Instrumente und Datenerfassung

Das Zusammenstellen und Erfassen von Sensordaten, Logs und Ergebnissen kann durch den Nutzer erfolgen oder ebenfalls unterschiedliche Automatisierungsgrade erfüllen.

Anhand der betrachteten Attribute lassen sich die Fähigkeiten und die Rahmenbedingungen von Testbeds strukturiert beschreiben.

2.2.4 Entwicklungsaspekte

Um die Wirksamkeit von Testbeds zu gewährleisten, müssen in der Entwicklung verschiedene Aspekte berücksichtigt werden. Dies umfasst die klare Definition des zu untersuchenden Systemaspekts, den Verwendungszweck und den Simulationsgrad des Testbeds sowie die Festlegung von Designzielen und Charakteristiken. Ein Evaluationsprozess ermöglicht es, die korrekte Implementierung zu validieren und die Erfüllung der Anforderungen zu überprüfen.
In diesem Abschnitt werden diese verschiedenen Aspekte genauer betrachtet.

Systemaspekt

Die Entwicklung von Testbeds setzt zunächst voraus, dass der zu untersuchende Systemaspekt klar definiert ist [3].

Dabei existieren Testumgebungen, die versuchen, mehrere Systemaspekte zu vereinen oder sogar domänenübergreifend zu agieren. Diese weisen oft einen Mangel an Detailtiefe auf, ermöglichen jedoch im Gegenzug, dass verschiedene Kontexte und Szenarien ohne umfangreiche Rekonfiguration getestet werden können [3].

Häufiger werden jedoch spezifische Testbeds entwickelt, die sich nur auf einzelne ausgewählte Systemaspekte konzentrieren. Diese ermöglichen eine detaillierte Implementierung und sind auf das konkrete System zugeschnitten. Dies begrenzt nicht nur den Entwicklungsaufwand, sondern ermöglicht auch eine gezieltere Bereitstellung von Fachwissen durch einzelne domänenspezifische Experten. Gleichzeitig geht damit jedoch eine verringerte Flexibilität in der Anwendung einher. Daher ist der Grad der Spezifität eine wichtige Design-Entscheidung [3].

Verwendungszweck

Mit der Festlegung des Systemaspekts geht einher, dass auch der Verwendungszweck und der Simulationsgrad (z. B. von rein softwarebasierten Simulationen über semiphysikalische bis hin zu ausschließlich physikalischen Testumgebungen) des Testbeds ersichtlich werden müssen [3]. Diese können in folgende Kategorien eingeteilt werden und bestimmen maßgeblich die Architektur, Designziele und das Toolset [22]:

- Training und Lernumgebungen
- Übungs-Testbeds
- Experimentier-Testbeds
- Test- und Evaluationsumgebungen
- Demonstrations-Testbeds

Designziele

Hinzu kommen die zentralen Designziele und Charakteristiken, die angestrebt werden und in unterschiedlichem Maße für die Glaubwürdigkeit und Wertigkeit des Testbeds ausschlaggebend sind [3].

Einige zentrale Eigenschaften, die in [3] vorgestellt werden, beschreiben Verhaltenseigenschaften wie z.B. die *Fidelity* und *Modularität*, aber auch Performance-Metriken, wie die *Skalier- und Erweiterbarkeit* sowie *Flexibilität und Adaptierbarkeit*. Weitere wichtige Eigenschaften umfassen *Wiederholbarkeit (Reproduzierbarkeit)*, *Kosteneffektivität* und *Openness*. Die Priorität dieser Eigenschaften richtet sich dabei nach dem Anwendungsfall.

Neben den zentralen Eigenschaften wird in der erwähnten Publikation eine Mapping-Struktur vorgestellt, die ebenfalls domänenübergreifend die Auswahl wichtiger Designkriterien und Eigenschaften fördert.

Die Festlegung dieser „Core Operational Characteristics“ eines Testbeds unterstützt nachweislich die Zuverlässigkeit und Glaubwürdigkeit desselben. Zudem enthalten die vorgestellten Eigenschaften Evaluierungsansätze, die eine Validierung ermöglichen und das Vertrauen in das Testbed stärken können [3].

Evaluation

Der Evaluationsprozess beinhaltet die Ablaufbeschreibung, um die korrekte Implementierung des Testbeds zu validieren [3]. Dafür kann die Erfüllung der Designziele und angestrebten Charakteristiken ausgewertet werden. Der durchlaufene Prozess dient anschließend als Nachweis dafür, dass der Aufbau und die gewählten Szenarien den Anforderungen entsprechen. Das Evaluationsverfahren kann dabei in den Schritten Verifikation, Validation und Akkreditierung erfolgen.

2.2.5 Einsatz & Herausforderungen

Oftmals entstehen Testbeds im Kontext eines Forschungsprojekts. Dabei werden sie häufig als zugeschnittene Anwendungen entwickelt und dienen bis zur Veröffentlichung der Ergebnisse als Testumgebung. Im Anschluss werden diese aber in den seltensten Fällen gepflegt, weitergenutzt oder überhaupt veröffentlicht. Das stellt unabhängige Forschung zur Validierung der Ergebnisse vor eine Herausforderung und beschränkt somit eine grundlegende Forschungsmethodik [39].

Erschwerend kommt hinzu, dass die Eigenschaften und Anforderungen, unter denen die Testbeds entwickelt werden, in der Literatur zwar diskutiert werden, aber selten einem zentralen Entwicklungsansatz folgen. Oftmals sind ausgewählte Attribute in den verschiedenen Forschungsarbeiten lediglich verstreut zu finden [3].

Daher ist es nicht nur wichtig, dass Test- instrumente, Umgebungen und Daten veröffentlicht werden, sondern mit ihnen auch eine systematische Beschreibung der Experimente. Diese trägt dazu bei, dass die Forschungsergebnisse nachvollzogen und reproduziert werden können [39].

2.2.6 Abschließend

Zusammenfassend lässt sich sagen, dass Testbeds eine wichtige und viel genutzte Ressource in der Security Forschung darstellen [3].
Der Begriff „Testbed“ umfasst dabei eine Vielzahl unterschiedlicher Test- umgebungen und Strukturen. Dadurch variieren auch die Anforderungen, Einsatzgebiete und verwendeten Architekturen erheblich. Eine klare Beschreibung des Simulationsansatzes, die Darstellung der Architektur und die Verwendung etablierter zentraler Charakteristiken, sind daher wichtige Aspekte bei der Entwicklung von Testbeds. Dies trägt zur Wiederverwendbarkeit bei, ermöglicht die Validierung der Ergebnisse durch unabhängige Stellen und unterstützt die Glaubwürdigkeit des resultierenden Testbeds.

3 Analyse & Design

In diesem Kapitel wird die Aufgabenstellung und das Ziel des Testbed genauer analysiert. Darauf aufbauend, werden einige wichtige Anwendungsszenarien und Qualitätsziele herausgearbeitet. Anschließend wird die Architektur des Testbed beleuchtet und wichtige Technologieentscheidungen präsentiert. Diese Betrachtung dient als Grundlage für die darauf folgende Implementierung.

3.1 Einführung

Das **Testbed** stellt eine Testumgebung dar, die es ermöglichen soll, den vorgestellten Flow-Korrelationsangriff auf verschlüsselte IM-Anwendung in einem realitätsnahen Kontext zu untersuchen. Für die Durchführung des beschriebenen Angriffs sind im wesentlichen 3 Komponenten notwendig.

1. Das Gerät des Opfers, welches ein Smartphone, Laptop oder ähnliches sein kann. Auf diesem ist die IM-Anwendung installiert.
2. Das Angriffssystem, welches den Netzwerkverkehr des Opfers mitschneidet, und zudem parallel die Nachrichten aus einem Zielkanal der IM-Anwendung beobachtet.
3. Der Zielkanal, welcher eine öffentliche IM-Gruppe oder Kanal sein kann.

Aus diesem Aufbau lassen sich die Daten erzeugen, die anschließend durch eine Erkennungsroutine klassifiziert werden können.
Das Testbed soll eine Umgebung bereitstellen in welcher diese drei Komponenten mit möglichst realitätsnahen Stimuli, konfigurierbaren Parametern und austauschbaren Erkennungsalgorithmen simuliert werden können. Dabei sollen bestehende Datensätze verarbeitet werden können, sowie weitere Daten erzeugt werden.

In den folgenden Abschnitten werden konkrete Anwendungsszenarien und Qualitätsziele des Testbeds herausgearbeitet. Darüber hinaus werden zentrale Entscheidungen und Kontextabgrenzungen beschrieben, sowie ein Blick auf die Architektur geworfen.

3.2 Qualitätsziele

Die im Folgenden vorgestellten und motivierten Qualitätsziele dienen als Orientierung für die in der Entwicklung angestrebten Funktionen. Anhand dieser Ziele werden in den folgenden Kapiteln die Lösungsstrategien herausgearbeitet und das Testbed abschließend evaluiert.

1. Reproduzierbarkeit (Zuverlässigkeit)
Für die wissenschaftliche Auswertung und Glaubwürdigkeit sind reproduzierbare Ergebnisse notwendig [3]. Im Rahmen der Testbedingungen soll das Testbed, bei wiederholter Durchführung, konsistente Ergebnisse liefern.

2. Genauigkeit (Fidelität)
Neben der Reproduzierbarkeit ist die Genauigkeit oder auch Fidelität der Ergebnisse eine wichtige Eigenschaft für die Glaubwürdigkeit des Testbeds. Diese beschreibt wie getreu das Testbed wichtige Umgebungsaspekte repräsentiert und nachempfindet. Diese wird vor allem durch einen realitätsnahen Kontext bestimmt [3], weshalb die Verwendung realitätsnaher Daten und die Einbindung physischer Geräte angestrebt werden soll.

3. Automatisierung
Die manuelle Durchführung aufwendiger und mehrstufiger Setup-Prozesse sowie die Konfiguration von Testgeräten kann fehleranfällig sein und erfordern Kenntnisse und Einarbeitungszeit seitens der Benutzer*innen. Um das erforderliche Know-how und die Interaktionen zu reduzieren, soll das Testbed, Testpläne und Skripte zur automatisierten Konfiguration bereitstellen.

Die vorgestellten Ziele stellen eine Auswahl dar, anhand derer das Testbed entwickelt und evaluiert werden soll.
Neben diesen Zielen werden weitere Aspekte angestrebt, wie die Langlebigkeit, Interoperabilität und Flexibilität. Um das Testbed zudem einer möglichst breiten und unabhängigen Forschungs-Gemeinschaft zugänglich zu machen, soll der Einsatz kostengünstiger, handelsüblicher Hardware und quelloffener Software angestrebt werden.
Diese Ziele dienen als Leitlinien während der Entwicklung und werden nicht als Evaluationskriterien verwendet.

3.3 Testbed Stimuli

Dieser Abschnitt befasst sich mit einer detaillierten Analyse der benötigten und vorhandenen Simulationsdaten.

3.3.1 Simulationsdaten

Um aussagekräftige Ergebnisse zu erzielen, ist es erforderlich, Simulationsdaten in das Testbed einzuführen. Eine Möglichkeit besteht darin, diese synthetisch zu generieren, wie es in der Publikation [10] dargestellt wird. So werden neben beobachteten Endbenutzeraktivitäten auch Aktivitäten mithilfe eines stochastischen Modells beschrieben, auf deren Grundlage anschließend die Simulationsdaten generiert wurden.

Die Verwendung von synthetischen Datensätzen ermöglicht eine höhere Flexibilität und eine nahezu unbegrenzte Menge an verfügbaren Daten. Allerdings bieten echte Daten häufig eine höhere Glaubwürdigkeit, wie in [3] betont wird. Der Datenverkehr aus realen Anwendungen ermöglicht eine genauere Abbildung der tatsächlichen Systemeigenschaften und kann daher als vertrauenswürdiger angesehen werden [3].

Es ist jedoch wichtig zu berücksichtigen, dass bei der Verwendung echter Daten, der Datenschutz und die Privatsphäre der Nutzer*innen zu respektieren sind.

Eine Möglichkeit für die Verwendung echter Endbenutzer*innen-aktivitäten besteht darin, öffentliche Gruppen der IM-Anwendung zu nutzen und die Nachrichten innerhalb dieser Gruppen aufzuzeichnen. Da es sich um öffentliche Gruppen handelt, ist dies eine einfache Methode um realitätsnahe und authentische Daten zu erfassen, ohne die Privatsphäre der Nutzer*innen zu verletzen [10].

3.3.2 Analyse bestehender Datensätze

Wie bereits erwähnt, haben die verwendeten Datensätze einen großen Einfluss auf die Ergebnisse der Erkennungsalgorithmen und damit auf den Erfolg des Angriffs.

Zum Evaluieren der Erkennungsalgorithmen genügen die beobachteten Zeitstempel und Größen der Ground-Truth Nachrichten aus den Messenger-Kanälen, sowie die dazugehörigen Netzwerkpakete. Ein Datensatz, der diese beinhaltet wird von den Autor*innen der

Studie [10] als Download[1] zur Verfügung gestellt.
Der Datensatz umfasst (komprimiert) 76 GB an Nachrichten und Netzwerkverkehr von den Messengern Signal, Telegram und Wickr. Abhängig der IM-Anwendung, sind teils nur synthetische oder aufgezeichnete Daten vorhanden. Die Dateiformate und Inhalte variieren je nach Anwendung und Szenario. So stehen auch Daten von Experimenten mit VPN-Anwendungen oder limitierter Bandbreite zur Verfügung.

Die aufgezeichneten Ground-Truth-Daten bestehen aus einem Zeitstempel, dem Nachrichtentyp (Foto, Text, Audio, Video) und der Nachrichtengröße.

Neben diesen, im Folgenden als „Message-Traces“ bezeichneten Daten, stehen die dazugehörigen (also parallel aufgezeichneten) Verkehrsdaten zur Verfügung.

Mithilfe dieser Daten sollen die in der Studie [10] vorgestellten Algorithmen validiert werden können. Dabei lässt sich der Einfluss der eingesetzten Algorithmen, der Beobachtungsdauer, sowie der dazugehörigen Parameter, auf die Erkennungsgenauigkeit untersuchen.
Nicht beurteilen lassen sich dagegen die Auswirkungen von Eigenschaften, die aufseiten der Anwendung, der Endgeräte, oder des Netzwerks und Server variieren können.
Dies könnte z.B. die Firmware Version, der Zustand der App beim Nachrichtenempfang (geöffnet, geschlossen), eine störanfällige WLAN-Verbindung oder abweichendes Nachrichtenaufkommen sein (z.B. vorwiegend Textnachrichten oder gleichzeitig versendete Fotos). Die Konfiguration des Testaufbaus für die vorliegenden Datensätze wurde nicht weiter spezifiziert.
Um solche Aspekte zu untersuchen, muss eine Umgebung geschaffen werden, welche es erlaubt, diese Eigenschaften in die Tests miteinzubeziehen.
Hierfür eignet sich das Einbinden von physischen Geräten mit möglichst realitätsnahen Umgebungsbedingungen. Dies trägt zur Testbed Fidelität bei und unterstützt so die Glaubwürdigkeit des Testbeds [3]. So können auch Beobachtungen aus realen Umgebungen wie z.B. aus Heimnetzen, wichtige Daten liefern [8]. Hierfür ist ein tragbares, einfach zu installierendes Setup, vom Vorteil.

Zusammenfassend lässt sich daher schließen, dass die bestehenden Datensätze zwar eine Grundlage bieten können, um die vorgestellten Algorithmen zu untersuchen, aber nicht ausreichen, um weitere Aspekte der Anwendbarkeit in realitätsnahen Umgebungen zu beurteilen. Hierfür braucht es die Möglichkeit, die Daten auch unter anderen Bedingungen

[1] https://traces.cs.umass.edu/

und Konfigurationen zu erzeugen.
Im Abschnitt 3.4 werden daher drei Anwendungsszenarien vorgestellt, die die Integration physischer Geräte zur Datenerzeugung ermöglichen.

3.4 Anwendungsszenarien

Um die Anwendbarkeit des Angriffs detailliert untersuchen zu können, wurde der Ablauf in drei Teilszenarien unterteilt. Diese Unterteilung ermöglicht eine genauere Analyse und Evaluierung des Angriffs und wird im Folgenden diskutiert.

Um realitätsnahe Szenarien mit hoher Fidelität zu untersuchen, ist es wichtig, möglichst reale Eingabedaten zu verarbeiten. Allerdings können solche Daten eine Herausforderung für die Wiederholbarkeit darstellen, wenn beispielsweise Nachrichten über ein störanfälliges und latenz-behaftetes Medium (z.B. WLAN) übertragen werden. Nicht vorhersehbare Interferenzen und Netzwerkauslastungen können zu erheblicher Varianz im gesamten Experiment führen.

Um diesen Herausforderungen zu begegnen, wurde das Angriffsszenario in drei Teilszenarien unterteilt. Dies ermöglicht zum einen, die Schwankungen auf einzelne Teile des Systems zu begrenzen und isoliert zu analysieren. Zum anderen gewährleistet diese modulare Aufteilung eine höhere Flexibilität, sowohl bei der Entwicklung, als auch bei der Anwendung des Angriffs. Die Teilszenarien sind inhaltlich aufeinander aufbauend und dennoch unabhängig voneinander und ermöglichen so, eine separate Analyse verschiedener Aspekte des Angriffs.
Zusammenfassend bestehen die drei Teilszenarien aus:

1. Anwendung der Erkennungsalgorithmen auf vorhandene oder neu erzeugte Datensätze
2. Generierung neuer Datensätze durch Aufzeichnung von Nachrichten in Messenger-Gruppen
3. Abspielen aufgezeichneter Nachrichten und simultanes Mitschneiden des Netzwerkverkehrs im Testnetzwerk

Im Folgenden werden die Anwendungsszenarien detaillierter vorgestellt.

3.4.1 Szenario 1 - Anwendung der Erkennungsalgorithmen

Dieses Szenario ermöglicht die Anwendung der Erkennungsalgorithmen auf bestehende und neu erzeugte Datensätze, um die verwendeten Algorithmen und Parameter eingehend zu untersuchen. In diesem Zusammenhang können diverse Aspekte analysiert werden, darunter unterschiedliche Beobachtungsintervalle, Erkennungsschwellwerte oder das Filtern spezifischer Nachrichtentypen. Zudem lässt sich der Einfluss der Zusammensetzung und Beschaffenheit des Datensatzes auf die Erkennungsrate untersuchen.

Eingabe
• Netzwerkverkehr als `pcap` • Ein Nachrichtenprotokoll als `TXT` oder `JSON`

Ausgabe
• Ergebnis des eingesetzten Erkennungsalgorithmus • Zwischenergebnisse aus dem Extraktionsprozess • Logdateien des Erkennungsalgorithmus • Grafik zur Analyse des Erkennungsvorgangs

Benötigte Komponenten und Geräte

- Testbed
- Erkennungsalgorithmus (verschiedene Implementationen möglich)

Konfiguration & Variationen

- Verschiedene Extraktions- und Korrelations-Parameter
- Verschiedene Erkennungsalgorithmen
- Verschiedene Datensätze

3.4.2 Szenario 2 - Daten Generierung

Dieses Anwendungsszenario bietet die Möglichkeit, realitätsnahe Datensätze zu erzeugen, indem die Nachrichten und Anhänge, sowie Metadaten aus echten IM-Kanälen aufgezeichnet werden. Die erzeugten Daten können als `JSON` gespeichert werden und als

Eingabedaten für weitere Anwendungsszenarien dienen. Dabei können variierende Beobachtungsintervalle, IM-Kanäle und die Zeitpunkte der Aufzeichnungen untersucht und protokolliert werden.

Eingabe
• Eine Liste der zu beobachtenden (öffentlichen) Gruppen als `CSV`

Ausgabe
• Ein `JSON` Nachrichtenprotokoll welches alle beobachteten Nachrichten sowie Metadaten enthält. • Alle zugehörigen Artefakte (Bilder, Videos, Audio)

Benötigte Komponenten und Geräte

- IM-Account der angreifenden Partei
- Testbed

Konfiguration & Variationen

- Unterschiedliche Beobachtungsdauer
- Verschiedene Gruppen
- Verschiedene Beobachtungszeitpunkte

3.4.3 Szenario 3 - Abspielen aufgezeichneter Aktivitäten

Die Durchführung des Angriffs setzt voraus, dass neben den Nachrichten aus dem Zielkanal auch der Netzwerkverkehr der Device Under Test (DUT) beobachtet wird. Dieses Anwendungsszenario bietet daher die Möglichkeit die aus 3.4.2 aufgezeichneten Nachrichten und Dateien in einem isolierten IM-Testkanal wieder abzuspielen. Eingebundene DUT empfangen diese Nachrichten, während der Netzwerkverkehr des Testnetzes mitgeschnitten wird.
Dieses Anwendungsszenario ermöglicht somit wiederholbare Tests mit gleichen Nachrichten Stimuli bei variierenden Geräte und Netzwerk Konfigurationen.

Eingabe
• Ein Nachrichtenprotokoll welches die abspielbaren Nachrichten Stimuli bereitstellt • Die zugehörigen Dateiinhalte (Bilder, Videos, Audio)

Ausgabe
• Aufgezeichneter Netzwerkverkehr (`pcap`) • Nachrichtenprotokoll der empfangenen Nachrichten (Nachrichtenprotokoll) • Interne Testbed Protokolle

Benötigte Komponenten und Geräte

- IM-Account der angreifenden Partei
- Testkanal
- Einzelne oder mehrere DUT
- Mitschneidekomponente (Zum Aufzeichnen des Netzwerkverkehrs)
- Testnetzwerk
- Testbed

Konfiguration & Variationen

- Verschiedene Netzwerkeigenschaften (Wi-Fi, Ethernet, VPN)
- Verschiedene Geräte (Smartphone, Desktop-PC)
- App Einstellungen (Automatischer Download von Anhängen, mögliche Privatsphäre Einstellungen)
- Verschiedene Beobachtungszeitpunkte

3.4.4 Zusammenfassend

In Szenario 1 können die präsentierten Ergebnisse der Erkennungsalgorithmen untersucht und validiert werden.
Szenario 2 ermöglicht die Generierung neuer Datensätze durch den Beitritt zu realen Messenger-Gruppen. Dadurch können die Auswirkungen verschiedener Gruppencharakteristiken miteinbezogen werden.

In Szenario 3 können die aufgezeichneten Nachrichten aus Szenario 2 abgespielt werden, um die Auswirkungen der Konfiguration von Endgeräten, IM-Anwendungen und Netzwerken zu untersuchen.

Die Aufteilung in die beschriebenen Teilszenarien erlaubt eine modulare und flexible Untersuchung des Angriffsszenarios, was die Reproduzierbarkeit und Wiederverwendbarkeit unterstützt. Durch die Aufteilung lassen sich zudem verschiedene Systemaspekte detaillierter untersuchen, was wichtig für die Testbed Fidelität ist.

3.5 Erkennungsalgorithmus

Im vorangegangenen Grundlagenkapitel wurden die Erkennungsalgorithmen, *Shape-Based* und *Event-based*, hinsichtlich ihrer prinzipiellen Funktionsweise erläutert. In diesem Abschnitt erfolgt eine Analyse aus Sicht der Anforderungen, um zu identifizieren, wie sich der *Event-Based* Algorithmus in das Testbed integrieren und anwenden lässt.

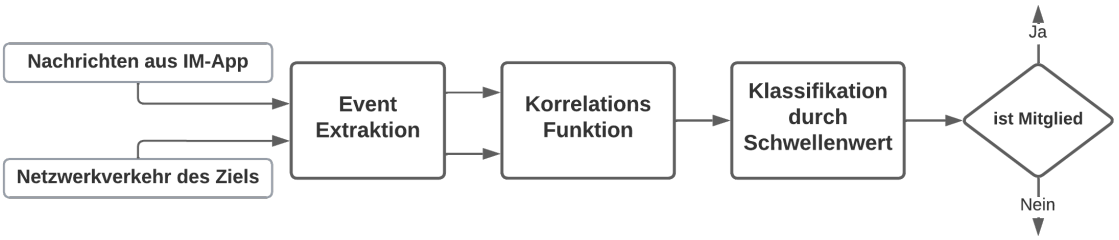


Abbildung 3.1: Bestandteile des *Event-Based* Algorithmus

Der *Event-Based* Algorithmus lässt sich in drei Hauptkomponenten zerlegen (Abbildung 3.1). Die erste Komponente übernimmt die Extraktion von Ereignissen aus den Netzwerkdaten, sowie die Zusammenfassung von Ground-Truth-Nachrichten mit einem kurzen Inter Message Delay (IMD). Hierbei wird ein Parameter t_e verwendet, der als Schwellenwert zur Identifikation der zusammenhängenden Pakete (oder Nachrichten) dient. Lässt sich ein IMD zwischen Pakete oder Nachrichten beobachten, welcher größer als dieser Schwellenwert ist, werden die darauf folgenden Pakete als neues Ereignis identifiziert.

Als zweite Komponente folgt die Korrelationsfunktion, bei welcher die extrahierten Ereignisse miteinander verglichen werden und ein Korrelationswert ermittelt wird. Hierbei sind zwei Parameter, ein Timing-Schwellenwert Δ und ein Größenschwellenwert Γ, erforderlich.

Im letzten Schritt wird zur Klassifikation, der Korrelationswert mit einem Schwellenwert η verglichen, um zu entscheiden, ob es sich um einen Treffer handelt und das Zielgerät Mitglied eines IM-Kanals ist.

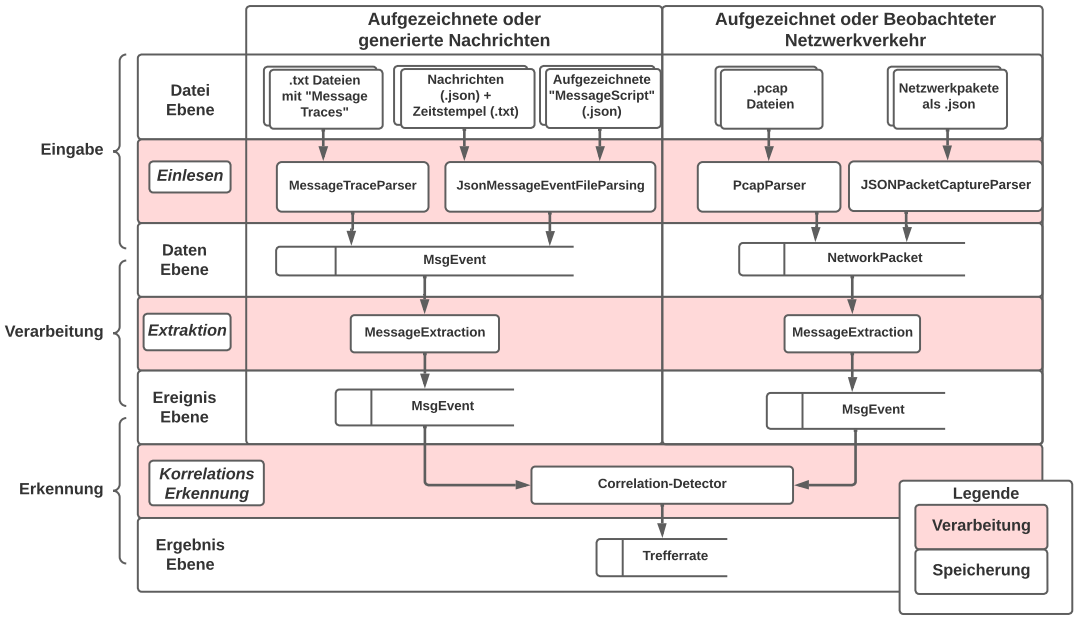


Abbildung 3.2: Datenfluss im System

Die Untersuchung des Datensatzes zeigt, dass die Eingabedaten in unterschiedlichen Dateiformaten vorliegen können. Da sich die Ground-Truth-Nachrichten in ihrer Form und benötigten Vorverarbeitung von den aufgezeichneten Netzwerkdaten unterscheiden, wurde das Einlesen und Verarbeiten in zwei Pipelines aufgeteilt (siehe Abbildung 3.2). Die verschiedenen Eingabeformate sollen dabei in ein internes Datenformat überführt werden, welches durch die in Abbildung 3.3 dargestellten Klassen repräsentiert wird.
Es sollen verschiedene Verarbeitungs- und Einlesekomponenten zur Verfügung stehen, die eine individuelle oder eine Stapelverarbeitung der Dateien ermöglichen. Eine parallele oder asynchrone Verarbeitung soll ebenfalls möglich sein.

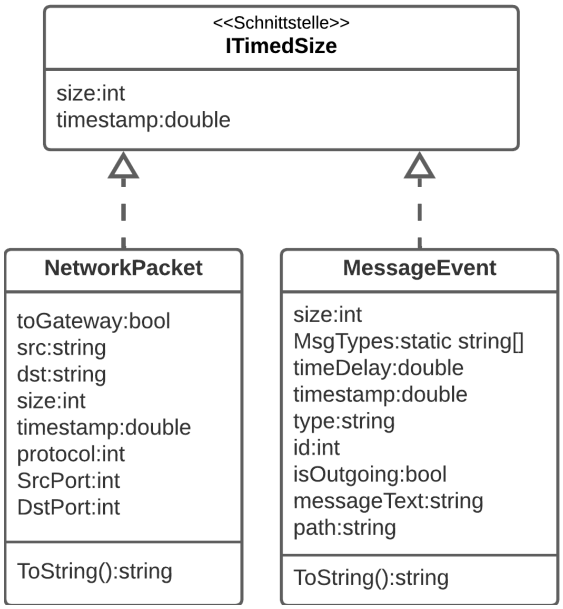


Abbildung 3.3: Internes Datenmodell

Abschließend:
Aus der Analyse des Erkennungsalgorithmus gehen 3 Hauptkomponenten hervor, die mit unterschiedlichen Parametern die Durchführung des beschriebenen Angriffs ermöglichen. Zur einheitlichen Verarbeitung unterschiedlicher Dateiformate wurde zudem ein Datenfluss Modell entwickelt, welches die Konvertierung der Eingangsdaten in ein internes Datenmodell zur Weiterverarbeitung vorschlägt.

Auf dieser Basis sollen nun die weiteren Schnittstellen und Randbedingungen des Systems analysiert werden

3.6 Randbedingungen

Im Rahmen der Entwicklung des Testbeds wurden verschiedene technische und organisatorische Randbedingungen ermittelt. Diese werden im folgenden kurz betrachtet.

Technische Randbedingungen

- Das Testbed soll unter einer liberalen OpenSource-Lizenz entwickelt werden und damit auch Quelloffene Bibliotheken und Programmteile verwenden. Zudem sollen offene Dateiformate unterstützt und verwendet werden.
- Beim Entwickeln und Testen muss die Privatsphäre von Benutzer*innen respektiert werden. Dies ist im Hinblick auf die zum Testen verwendeten Nachrichten aus IM-Kanälen wichtig.
- Die parallel verwendbaren Telegram Clients sind durch Telefonnummern-Bindung limitiert. Jeder Account erfordert eine Aktivierung per SMS oder Anruf [10].

Organisatorische Randbedingungen

- Veröffentlichung des Quelltextes:
  Der Quelltext sowie Testpläne und in der Entwicklung erzeugte Datensätze werden (soweit möglich) unter einer OpenSource Lizenz auf Github zur Verfügung gestellt
- Dokumentation:
  Zur Dokumentation der Architektur wird eine an ARC-42 angelehnte Struktur verwendet.
- Zeitplan:
  Das Projekt muss innerhalb der 6-monatigen Abgabefrist nach Anmeldung der Bachelorarbeit fertigstellt werden
- Vorgehensmodell:
  Die Entwicklung erfolgt iterativ, Inkrementell (agile)

Diese Randbedingungen sind entscheidend für den erfolgreichen Entwicklungsprozess und die Erreichung der angestrebten Ziele des Testbeds. Sie legen die Grundlagen für eine transparente und datenschutzkonforme Umsetzung der Forschungsarbeit.

3.7 Kontextabgrenzung

Der folgende Abschnitt dient zur Identifikation möglicher Systemgrenzen und Schnittstellen zur Außenwelt. Diese Abgrenzung ist ein wichtiger Schritt um den genauen Kontext des Testbeds zu definieren.

3.7.1 Fachlicher Kontext

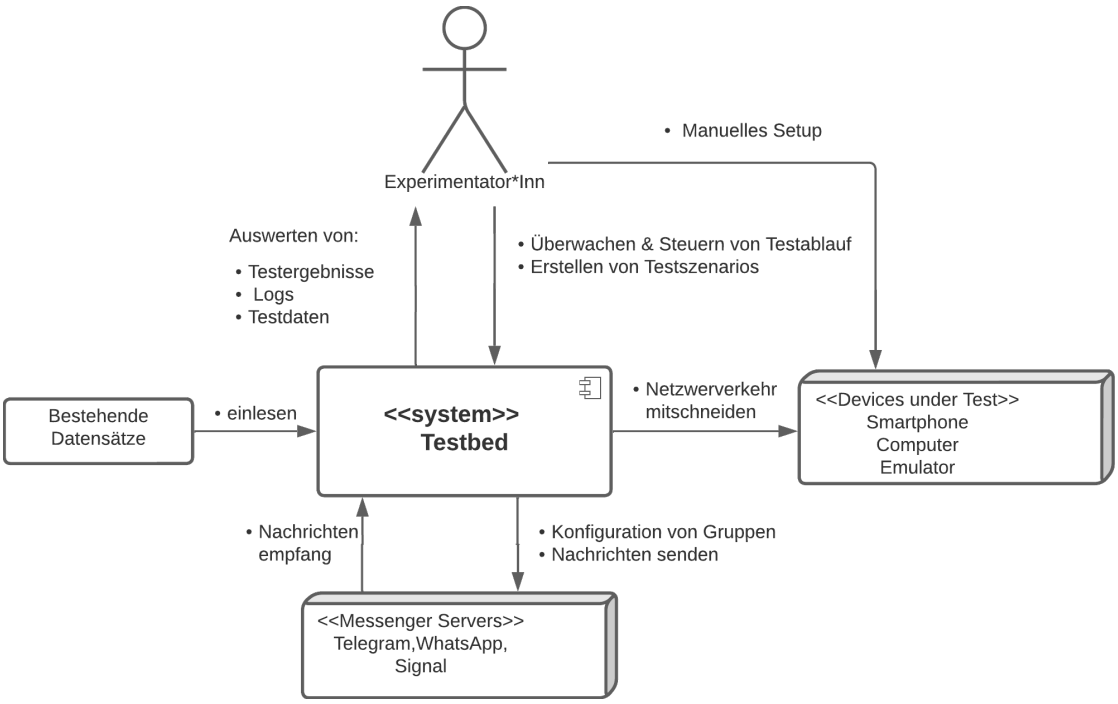


Abbildung 3.4: Fachlicher Kontext

Experimentator*inn:

Die experimentierende Person kann Testpläne und Szenarios mithilfe des Testbed UIs erstellen. Testpläne können Setup-Prozesse beschreiben, verschiedene Algorithmen und Parameter beinhalten und Werkzeuge zur Auswertung bereitstellen. Während der Ausführung kann der Ablauf überwacht und z.B. durch Stoppen und Wiederholen von Teilschritten, gesteuert werden. Auch die Auswertung und Analyse von Testergebnissen, der erzeugten Artefakte und Logs erfolgt durch die experimentierende Person und wird durch das Testbed UI unterstützt.

Device under Test:

Auf dem DUT wird die zu untersuchende IM-Anwendung ausgeführt. Das Gerät kann unterschiedliche Hard- und Softwarekonfigurationen aufweisen und wird mit dem Testnetzwerk verbunden.

Messenger Server:

Für das Versenden und Empfangen von Teststimuli wird eine Kommunikation mit dem Server der IM-Anwendung benötigt. Das Hinzufügen und Verlassen von Gruppen oder Kanälen kann hierüber teilautomatisiert durchgeführt werden.

Bestehende Datensätze:

Bereits bestehende Datensätze können vom System als Eingabedaten verwendet werden. Eine interne Vorverarbeitung stellt einen reibungslosen Ablauf sicher.

3.7.2 Technischer Kontext

Zur Identifikation der technischen Schnittstellen und Systemgrenzen bietet Abbildung 3.5 einen Einblick in den technischen Kontext des Systems.

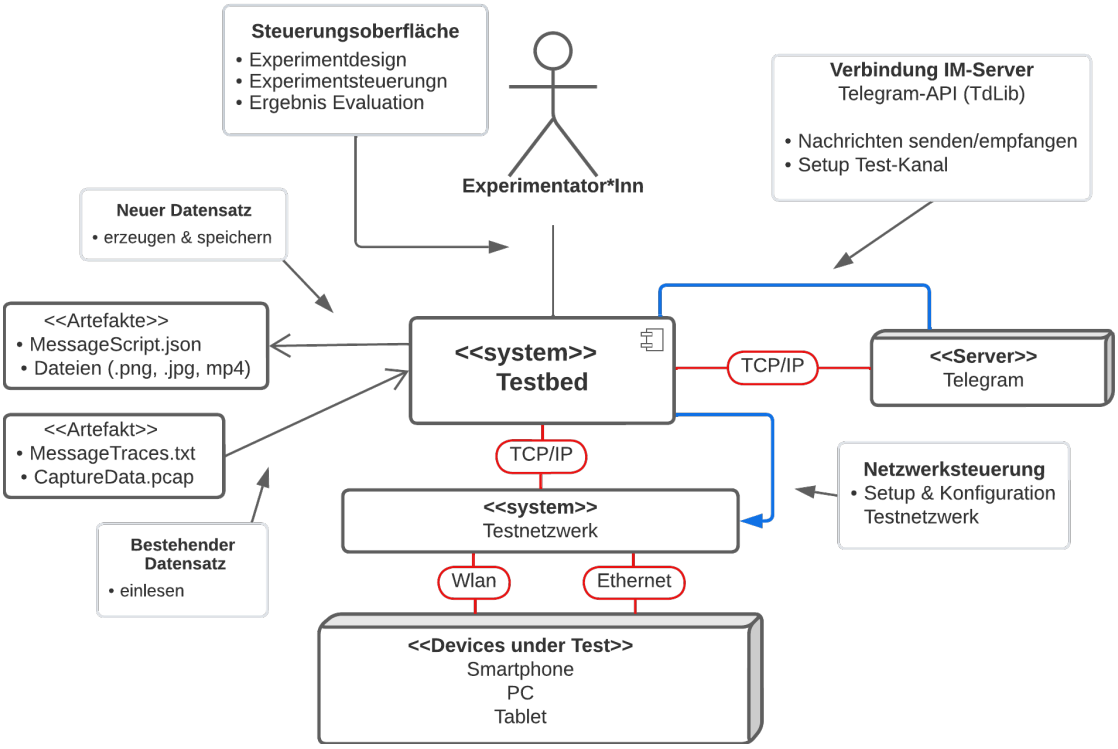


Abbildung 3.5: Technischer Kontext

Besondere Aufmerksamkeit gilt den Schnittstellen zwischen der IM-Anwendung und dem Testbed. In diesem Zusammenhang muss das spezifische Protokoll der jeweiligen Nachrichten-App berücksichtigt werden. Bei Telegram handelt es sich beispielsweise um das Protokoll *MTProto*. Mit TDlib[2] existiert eine Bibliothek die das Protokoll in großen teilen abstrahiert und eine plattformübergreifende Verwendung ermöglicht.

Abschließend:
Im Rahmen dieses Abschnitts wurden die möglichen Systemgrenzen und Schnittstellen zur Außenwelt identifiziert, um den genauen Kontext des Testbeds zu definieren. Besondere Aufmerksamkeit galt den Schnittstellen zwischen der IM-Anwendung und dem

[2] https://github.com/tdlib/td

Testbed, insbesondere in Bezug auf das spezifische Protokoll der jeweiligen Nachrichten-App.

3.8 Technologie Entscheidungen

Im folgenden Abschnitt werden wichtige Technologieentscheidungen erläutert, die für das Testbed von zentraler Bedeutung sind.

3.8.1 Testmanagement-Software

Für die automatisierte Testdurchführung wird ein Testmanagement-System benötigt. Die Hauptaufgaben des Systems sind folgende [5]:

- Verwalten der DUT und Instrumente im Testsystem
- Bereitstellung eines Editors zur flexiblen Erstellung von Testabläufen
- Bereitstellung eines UI, in dem die Testausführung überwacht und gesteuert werden kann.
- Bereitstellung einer Schnittstelle zur Einbindung von Test- Instrumenten und Treibern.
- Verwalten interner Protokolle und Testergebnisse.
- Ausführung und Planung der Tests.

Die Auswahl eines passenden Systems ist ein wichtiger Schritt, da die eingesetzte Umgebung maßgeblichen Einfluss hat, auf die Integrierbarkeit von Testinstrumenten und Geräten, sowie die Interoperabilität und damit Wiederverwendbarkeit von Testplänen und Treibern.
Darüber hinaus spielt die Langlebigkeit eine wichtige Rolle für die Reproduzierbarkeit und wird durch kontinuierliche Entwicklung, Wartung und Dokumentation gewährleistet [4][5].
Diese Merkmale stehen in direktem Zusammenhang mit den angestrebten Qualitätszielen und stellen daher wesentliche Kriterien dar, die neben den Randbedingungen berücksichtigt werden müssen.
Im Folgenden werden daher einige ausgewählte Testmanagement-Systeme untersucht, um darauf aufbauend, eine Entscheidung für ein geeignetes System zu treffen. Eine Übersicht

über diese Systeme wird in Tabelle 3.1 präsentiert.

Framework	**Maintainer**	**Lizenz**	**Kosten**	**Aufwand**	**Sprache**
Eigenentwicklung	-	Frei	Frei	Hoch	C#
OpenTAP	Keysight	MPL v.2	Frei	Gering	C#, Python
Teststand	NI	Proprietär	2.620$/Jahr	Gering	LabVIEW, Python, C/C++, .NET
ActiveATE	Astronics	Proprietär	auf Anfrage	Mittel	C#, VB.NET, VBScript

Tabelle 3.1: Vergleich von Test-Management Systemen

Eigenentwicklung

Häufig werden automatisierte Testsysteme von Grund auf neu entwickelt [47], weshalb eine Eigenentwicklung in Erwägung gezogen wurde. So lässt sich die Entwicklung gezielt auf die Qualitätsziele abstimmen und die Funktionalitäten, sowie der Umfang, auf die Anforderungen zuschneiden. Das resultierende System kann kostenfrei genutzt werden und erlaubt die Veröffentlichung unter einer freien Lizenz.
Allerdings weisen von Grund auf neu entwickelte Systeme häufig eine verringerte Flexibilität und Wartbarkeit auf [47]. Dies steht im Konflikt zur Änderbarkeit, Langlebigkeit und Modularität. Zudem ist ein deutlicher Entwicklungsaufwand zu beobachten [39], der im Konflikt zu den zeitlichen Randbedingungen stehen kann.

Der Einsatz von bestehenden Frameworks dagegen, kann die Entwicklungseffizienz verbessern und fördert die Wartbarkeit und Wiederverwendbarkeit des Testsystems [47]. Aus diesen Gründen wurden die Systeme von marktführenden Unternehmen wie National Instruments (NI) und Astronics miteinbezogen.

NI (Teststand)

NI liefert mit Teststand, LabVIEW und vielen weiteren Produkten eine umfassende Soft- und Hardware-Sammlung für das automatisierte Testen. Als langjähriger Vertreter in dem Bereich der Test- und Messinstrumente existieren viele bereits bestehende Instrumente und Gerätetreiber, sowie eine ausführliche Dokumentation und Support. Zudem erlaubt eine modulare Architektur eine auf den Anwendungsbereich zugeschnittene Entwicklung [17]. So lässt sich eine sehr gute Modularität, Integrierbarkeit und Langlebigkeit abschätzen, wie auch der Einsatz von Teststand und LabVIEW in [43] und [49] zeigt.

Verwendbar ist Teststand allerdings nur unter einer proprietären Lizenz, die für 2.620 $/Jahr (*Stand 02.07.23*) erhältlich ist [17]. Dies steht im Konflikt mit der Kosteneffizienz und Interoperabilität.

ActiveATE

Astronics bietet mit ActiveATE ein ähnliches System an, das auf .NET basiert und eine vollständige integrierte Entwicklungsumgebung (IDE), einen Testsequenzer und eine Geräteverwaltung bereitstellt [6]. Jedoch sind die proprietäre Lizenz und die Hardwarebindung nicht im Einklang mit den angestrebten Qualitätszielen und Randbedingungen.

Control and Management Framework (OMF)

Zu erwähnen ist noch das OMF, welches ursprünglich für das Orbit Wireless Testbed entwickelt wurde und nun unter einer freien MIT-Lizenz zur Verfügung steht [39]. Die Entwicklung wurde allerdings vor einigen Jahren eingestellt.

OpenTAP

OpenTAP ist ein Testsequenzer und eine Automatisierungsumgebung, die ursprünglich als internes Open-Source-Projekt namens „TAP“ von Keysight Instruments entwickelt wurde. Seit 2019 steht das Kernprojekt unter einer MPLv2-Lizenz zur Verfügung und wird aktiv von Keysight und Nokia weiterentwickelt [35].

OpenTAP basiert auf einer herstellerneutralen, standardisierten Architektur und verwendet offene Dateiformate [35]. Dies sind wichtige Kriterien für Interoperabilität und Wiederholbarkeit. Die Architektur von OpenTAP umfasst eine Kernkomponente, an die Plugins mit spezifischer Funktionalität für die DUT, die Testplanung, Instrumente, sowie Logging, Steuerung und Export angebunden werden [36]. Dies ermöglicht die Entwicklung wiederverwendbarer Plugins und unterstützt die Modularität des Gesamtsystems.

Vorhandene Plugins stellen neben einigen Kontrollelementen und Logging Funktionalitäten auch einen grafischen Editor für die Testplanung und -überwachung bereit, was die Benutzerfreundlichkeit verbessern kann und den Entwicklungsaufwand reduziert.

Das Framework basiert auf .NET und ermöglicht die Plugin-Entwicklung sowohl in C#, als auch in Python. Auch das TRIANGLE-Projekt setzte für seine Automatisierungsumgebung auf TAP, damals noch in einer kommerziellen Version. Die Entwicklung des Projekts wird in [20] und [37] beschrieben.

Schlussfolgerung

Aufgrund mehrerer Faktoren wurde die Entscheidung getroffen, OpenTAP als Testmanagement Framework zu verwenden. Erstens bietet OpenTAP eine herstellerneutrale, standardisierte Architektur und verwendet offene Dateiformate, was die Interoperabilität und Wiederholbarkeit der Tests unterstützt. Zweitens ermöglicht die Integration von spezifischen Funktionalitäten über Plugins eine hohe Modularität des Systems. Dies erleichtert die Anpassung an die Anforderungen und verbessert die Entwicklungs- und Wartungseffizienz.

Zusätzlich stellt OpenTAP einen grafischen Editor für die Testplanung und -überwachung bereit, was die Benutzerfreundlichkeit erhöht und den Entwicklungsaufwand reduziert. Darüber hinaus basiert das Framework auf .NET und ermöglicht sowohl die Entwicklung von Plugins in C# als auch in Python, was Flexibilität bei der Änderbarkeit des Systems bietet.

Im Vergleich zu anderen Testmanagement-Systemen wie Teststand von National Instruments und ActiveATE von Astronics, ermöglicht OpenTAP eine kostenfreie Nutzung und die Veröffentlichung unter einer freien Lizenz. Dies ist im Einklang mit den Qualitätszielen der Kosteneffizienz und Interoperabilität. Die aktive Entwicklung und Unterstützung von OpenTAP durch Keysight und Nokia, sowie die Tatsache, dass es bereits in anderen Projekten, wie dem TRIANGLE-Projekt, erfolgreich eingesetzt wurde, sprechen für die Langlebigkeit und Stabilität des Systems.

Basierend auf diesen, Gründen wurde OpenTAP als das geeignete Testmanagement-System ausgewählt.

3.8.2 Hardware

Im folgenden Abschnitt werden geeignete Hardware Komponenten für das Testbed diskutiert.

Device-Under-Test

Das System soll den Einsatz physischer Endgeräte ermöglichen.
Alle netzwerkfähigen Geräte, die eine IM-App verwenden, sind prinzipiell zur Untersuchung von Flow-Korrelationsangriffen interessant. Das können z.B. Smartphones, Laptops, Desktop-PCs, Tablets und IoT-Geräte sein.
Diese Geräte unterscheiden sich zum Teil erheblich in der verwendeten Systemarchitektur, dem Betriebssystem und der einsetzbaren IM-Versionen, sowie in der zur Verfügung stehenden Schnittstelle.
Dies kann eine Herausforderung für eine automatisierte Integration der Geräte in das Testbed bedeuten, da die Heterogenität der Systeme einerseits ein sehr spezifisches Setup voraussetzt, während im gleichen Moment eine große Variation von Geräten abgedeckt werden muss.
Um den Entwicklungsaufwand realistisch zu gestalten und gleichermaßen eine ausreichende Flexibilität und Offenheit gegenüber unterschiedlichen Gerätetypen zu ermöglichen, wird daher von einem automatisierten Setup der DUT in dieser Entwicklungsiteration abgesehen. Durch die modulare Architektur des Systems wird das nachträgliche Einbinden von Automatisierungskomponenten ermöglicht.

Test Netzwerk

Um spezifische Systemaspekte gezielt zu untersuchen, ist es erforderlich, ein kontrollierbares und unabhängiges Testnetzwerk zu verwenden. Dies ermöglicht eine isolierte Untersuchung von Zusammenhängen und Effekten [22].

In dem von [10] beschriebenen Angriffsszenario zeichnet der Angreifer den Netzwerkverkehr der Endgeräte an den Internet exchange point (IXP)s oder Internet service provider (ISP)s auf. Prinzipiell kann dies allerdings auch an anderen Stellen des Netzwerks stattfinden.

Um eine kontrollierbare und reproduzierbare Testumgebung zu schaffen, die im Einklang mit der Kosteneffizienz steht, wird daher ein lokales Testnetzwerk verwendet.
Um Flexibilität für verschiedene Gerätetypen zu gewährleisten, sollen sowohl Wi-Fi- als auch Wired-Ethernet-fähige Geräte unterstützt werden.

Darüber hinaus wird eine Komponente zum Mitschneiden des Netzwerkverkehrs benötigt. [44] beschreiben verschiedene Methoden mit welchen dies möglich ist:

1. Mitschneiden des Verkehrs direkt auf der Netzwerk-Gateway.
2. Umleiten des Datenverkehrs durch ARP/NDP-Spoofing, auf die Beobachtungskomponente.
3. Durch die Verwendung eines Switch mit Port-Spiegelung, an dem die Beobachtungskomponente angeschlossen ist.
4. Durch die Verwendung eines Hubs, welcher jeglichen Netzwerkdaten an alle angeschlossenen Geräte spiegelt.

Punkt 3. erfordert den Einsatz spezialisierter und kostenintensiver Hardware, weshalb von dieser Möglichkeit abgesehen wird.
Bei 2. besteht die Herausforderung darin, eine stabile Implementierung von ARP/NDP-Spoofing zu gewährleisten, um den Datenverkehr umzuleiten. Dies stellt ein Risiko für die Wiederholbarkeit dar und eignet sich damit ebenfalls nur bedingt.
[44] entscheiden sich daher für die Verwendung eines Hubs, da dies eine kostengünstige Alternative darstellt. Ein Hub bildet bei mehreren Geräten allerdings eine Kollisionsdomäne und wird in der Realität nur noch selten verwendet. Dies schränkt die Auswahl verfügbarer Geräte ein.

Eine korrekt konfigurierte und ausgestattete Gateway hingegen, vereint die benötigten Anforderungen in einem Gerät. Dieses lässt sich nicht nur kostengünstig umsetzen, sondern reduziert auch die Komplexität des Testaufbaus.
So wird die Portierbarkeit unterstützt, welche wiederum einen flexibleren Einsatz in verschiedenen Umgebungen ermöglicht und so positive Auswirkungen auf die Testbed-Fidelität haben kann.

Aus diesen Gründen wurde sich für die Gateway-Methode entschieden.

So lassen sich die Netzwerkgeräte per Wired-Ethernet oder Wi-Fi zu dem Testnetzwerk hinzufügen. Mithilfe von *iptables* (Layer-3) oder *ebtalbes* (layer-2) ist zudem die Realisierung einer NAT möglich. So kann der Netzwerkverkehr des Testnetzes weitergeleitet und gefiltert werden.

Selektion des Netzwerkverkehrs

Um den Traffic von IM-Anwendungen gezielt aufzuzeichnen, liegt es nahe, die Verbindungsinformationen der Pakete zu verwenden. Da IM-Anwendungen wie Telegram, WhatsApp und Signal in der Regel zentralisiert aufgebaut sind [10], nutzen sie zentrale

Server als Vermittlungsstellen für die Kommunikation und ermöglichen so das Filtern des Traffics auf die IP-Adressen der Server.

Es ist jedoch zu beachten, dass einige IM-Apps Content Delivery Networks (CDNs) verwenden, was Filterung erschweren kann, da nicht mehr nur eine Zieladresse verwendet wird [10]. Im Falle von Telegram, sind die Netzwerk-IP-Adressen online verfügbar[3] und können so als Filterkriterien festgelegt werden. Auf diese Weise kann der Traffic der IM-Anwendungen gezielt untersucht werden.

Es ist wichtig anzumerken, dass diese Vorgehensweise nur funktioniert, wenn kein VPN (Virtual Private Network) oder Proxy verwendet wird. Falls in einem Experiment ein VPN eingesetzt werden soll, muss eine alternative Methode zum Filtern in Betracht gezogen werden. Ein Ansatz wäre beispielsweise die Verwendung von Deep Packet Inspection (DPI).
Die Aufzeichnungskomponente sollte daher auch die Möglichkeit bieten, andere Filterkriterien zu setzen, um den gewünschten Traffic zu erfassen und zu analysieren.
Auch denkbar ist es den ungefilterten Netzwerkverkehr zu analysieren. Dies bleibt weiterhin möglich.

Zusammenfassend

In diesem Abschnitt wurden verschiedene Methoden zur Erfassung des Netzwerkverkehrs im Testnetzwerk besprochen. Die ausgewählte Lösung zeichnet sich durch ihre kostengünstige Umsetzung in einem Gerät aus. Anschließend wurden einige Rahmenbedingungen für das Mitschneiden und Filtern des Netzwerkverkehrs näher beleuchtet.

[3] https://core.telegram.org/resources/cidr.txt

3.9 Architektur

In diesem Abschnitt sollen die Architekturprinzipien, -muster und -komponenten vorgestellt werden, die das Testbed definieren und seine verschiedenen Aspekte beeinflussen. Dabei werden sowohl die Anwendungsszenarien, als auch die angestrebten Qualitätsziele berücksichtigt. Zur Übersichtlichkeit und Begrenzung des Umfangs wird zunächst eine Übersicht über die zentralen Komponenten gegeben. Anschließend wird in einer detaillierteren Betrachtung auf einzelne Komponenten und Prinzipien eingegangen. Es sei darauf hingewiesen, dass dieser Abschnitt keinen Anspruch auf eine vollständige Dokumentation erhebt.

3.9.1 Komponenten

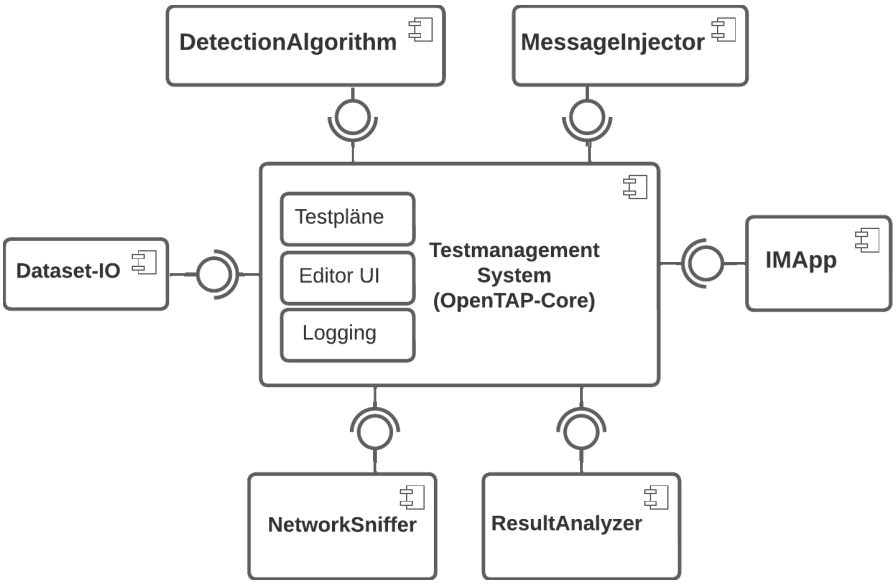


Abbildung 3.6: Komponentendiagramm Ebene 1

Zentraler Bestandteil ist das Testmanagement-System, welches durch OpenTAP bereitgestellt wird. An dieses sind die funktionalen Komponenten angebunden, die durch die Testpläne parametrisiert und gezielt ausgeführt werden können.
Eine standardisierte und flexible Schnittstelle zum Testmanagement-System, ermöglicht die modulare Anbindung dieser Komponenten als Plugins. Dies unterstützt die Wartbarkeit und ermöglicht eine unabhängige Entwicklung und Kompilierung dieser Komponen-

ten als eigenständige DLLs [13]. Des Weiteren können die Plugins auf zentrale Dienste zurückgreifen, welche z.B. die Verwaltung der Testinstrumente übernehmen, oder ein zentrales System zum Logging bereitstellen.
Die Hauptaufgaben des Testbeds werden durch die folgenden Module gekapselt:

Detection-Modul

Das *Detection-Modul* verwaltet die Erkennungsalgorithmen, welche zur Durchführung der Angriffe benötigt werden.
Hierfür können die, durch das *Dataset-IO*-Modul bereitgestellten Daten, vorverarbeitet und den Erkennungsroutinen bereitgestellt werden. Die Routinen werden automatisiert mit den benötigten konfigurierbaren Parametern und vorverarbeiteten Daten versorgt. Die Ergebnisse der Erkennung werden im Anschluss aufbereitet und zur Weiterverarbeitung bereitgestellt.

IMApp

Das *IMApp*-Modul fungiert als Schnittstelle und Kapselungsschicht für die Verbindung zu den IM-Servern, wie zum Beispiel Telegram.
Das *IMApp*-Modul beinhaltet Funktionen, wie das Senden und Empfangen von Nachrichten und das Beitreten und Verlassen von Gruppen. Durch diese Funktionen wird die Interaktion mit den konkreten IM-Servern abstrahiert und dem Testmanagement System eine einheitliche und vereinfachte Schnittstelle geboten.
Das Modul spielt somit eine entscheidende Rolle bei der Bereitstellung der Messaging-Funktionalität im Testbed und ermöglicht es, die Verbindung zu den IM-Servern effizient zu verwalten.

Dataset-IO

Das Modul *Data-IO* ist verantwortlich für die Verarbeitung von unterschiedlichen Datensätzen im Testbed. Es führt robuste Serialisierungs- und Deserialisierungsoperationen für verschiedene Datenformate durch, um die Daten für die interne oder externe Weiterverarbeitung bereitzustellen.

Das Modul ermöglicht es, Daten aus verschiedenen Quellen einzulesen und in das interne Testbed-Format zu konvertieren. Es bietet auch die Möglichkeit, Ergebnisse während des Testvorgangs zu speichern oder zwischen verschiedenen Komponenten auszutauschen.
Dadurch ermöglicht das Modul eine effiziente Handhabung von Daten im Testbed, sowohl für den Eingabe- als auch für den Ausgabeprozess. Es stellt sicher, dass die Daten korrekt verarbeitet und bereitgestellt werden, um eine reibungslose Durchführung der Testszenarien zu gewährleisten.

Network-Sniffer

Das *Network-Sniffer*-Modul ist für das Erfassen und Verarbeiten von Netzwerkpaketen zuständig. Es kann sowohl lokale als auch entfernte, über das Netzwerk erreichbare Mitschneide-Geräte, steuern und verwalten.
Das Modul stellt Funktionen zum Starten und Stoppen von Netzwerkaufzeichnungen bereit, ermöglicht die Konfiguration eines Aufzeichnungs-Filters, und kann entfernte Mitschneide-Geräte vollständig automatisiert konfigurieren.
Darüber hinaus können die aufgezeichneten Daten automatisiert zum Kontrollsystem kopiert werden, um eine zentralisierte Verarbeitung und Analyse der erfassten Pakete zu ermöglichen.

Message-Injector

Um die Testbed Stimuli in das System zu geben, wird das *Message-Injector* Modul benötigt. Dieses simuliert die Endbenutzer*innenaktivitäten, indem gezielte Nachrichten aus realen Aufzeichnungen in die spezifizierten IM-Testkanäle gesendet werden.

Result-Analyzer

Der *Result-Analyer* kapselt die Funktionen zum Analysieren und Darstellen von (Zwischen-) Ergebnissen. Es können automatisiert Grafiken erzeugt, sowie Statistiken zu den vorliegenden Datensätzen generiert werden.

3.9.2 Framework Kopplung

Die zentrale Verwendung eines Frameworks, wie in diesem Fall OpenTAP, birgt einige Risiken für die Modularität und Wiederverwendbarkeit der entwickelten Module.
So bedeutet die Verwendung eines Frameworks häufig eine enge Kopplung der Module an die durch das Framework bereitgestellte Architektur und Schnittstellen. Dies lässt sich besonders dann beobachten, wenn Komponenten von Basisklassen des Frameworks erben müssen um die Funktionalitäten nutzen zu können [13].

Dies kann zu verschiedenen Implikationen führen. Zum einen ist die Architektur damit nicht immer sauber von Frameworks getrennt, was die Modularität und Flexibilität beeinträchtigen kann. Zum anderen kann die Integration in das Framework, die Skalierbarkeit und Wiederverwendbarkeit behindern, da Updates zu veränderten Schnittstellen oder implementationsspezifischen Änderungen führen können. Insbesondere wenn sich das Framework in eine andere Richtung entwickelt, keinen Support mehr erhält, oder besser geeignete Alternativen verfügbar werden, ist eine enge Abhängigkeit zu einem Framework problematisch [13].

Als Lösung für diese Problematik empfiehlt die Fachliteratur, das Framework zwar zu nutzen, aber keine enge Bindung einzugehen. Das Framework sollte als Detail betrachtet werden. Anstatt von Framework-Basisklassen abzuleiten, sollten Proxys verwendet werden [13].

Dieser Ansatz wurde in der Architektur berücksichtigt und lässt sich in der Umsetzung anhand der folgenden Abschnitte beobachten.

3.9.3 Testschritte und Instrumente

Im OpenTAP-Framework werden Instrumente genutzt, um (physikalische) Geräte zu verwalten. Sie stellen eine Verbindung zu den Geräten her und kapseln die Funktionalität für die Testschritte. Dies Ermöglicht den Benutzer*innnen einen spezifischen Testablauf anhand vorhandener Testschritte und Instrumente zu entwerfen.

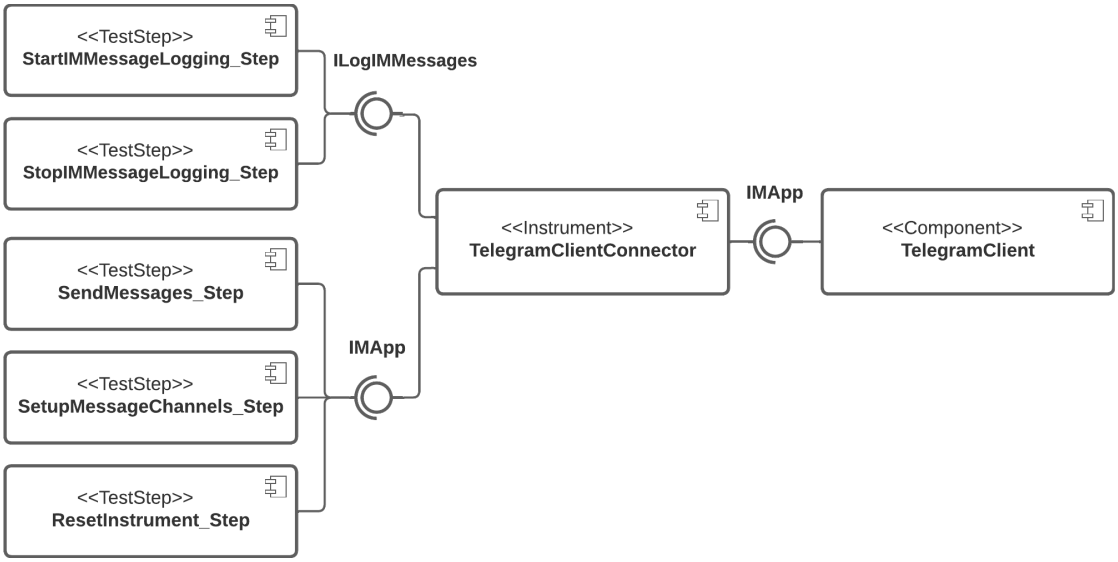


Abbildung 3.7: *IMApp*-Modul: Verwendung von Instrumenten und Testschritten

Ein Beispiel hierfür ist das Telegram-Verbindungsmodul (siehe Abbildung 3.7). Der *TelegramClientConnector* stellt über zwei Schnittstellen die Funktionen einer IM-Anwendung bereit. Um die Unabhängigkeit der Implementierung vom Framework zu gewährleisten, liegt die eigentliche Logik in einer separaten Komponente, dem *TelegramClient*. Dieser implementiert die gleiche Schnittstelle (*IMApp*), die durch den *TelegramClientConnector* als Instrument für die Testschritte bereitgestellt wird. Dies verdeutlicht den Einsatz des Proxy-Patterns, um eine lose Kopplung zum Framework sicherzustellen.

Die Testschritte stellen die spezifischen Funktionen bereit, die durch den *TelegramClient-Connector* verwaltet werden. Dadurch wird eine modulare Verwendung von kontrollierten Funktionen durch die Testschritte ermöglicht. Dies erlaubt den Anwender*innen, einen spezifischen Testaufbau aus modularen Komponenten selbst zu erstellen.

Die definierten Schnittstellen stellen neben der Entkopplung des Framework sicher, dass der Messenger-Client jederzeit durch eine andere Implementierung ersetzt werden kann. Experimente mit anderen Messenger-Clients (z.B. Signal oder WhatsApp) können somit ohne Rekonfiguration des Testablaufs durch zusätzliche Messenger-Instrumente realisiert werden.

IMApp Verbindungskomponente

Ein detaillierter Einblick in die Architektur der IMApp Verbindungskomponente wird in Abbildung 3.8 gegeben. Neben den erwähnten Schnittstellen für die lose Kopplung werden zwei weitere Entwicklungsmuster deutlich. Das Factory-Pattern wird verwendet, um die Instanziierung der TelegramClient-Komponente zu ermöglichen, ohne tiefe Abhängigkeiten zu den darunterliegenden Klassen zu erfordern. Eine Fassade abstrahiert zusätzlich die Logik zur Verwaltung des Nachrichtenempfangs und der Kommunikation mit dem Telegram-Server.

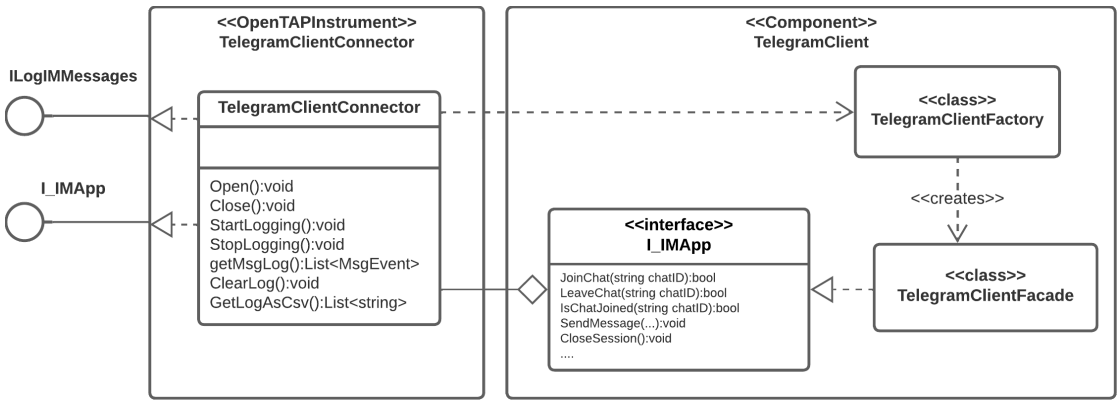


Abbildung 3.8: Klassendiagramm des Telegram-Clients

Der so erreichte Aufbau zielt darauf ab, eine wartbare und wiederverwendbare Architektur sicherzustellen. Die genannten Prinzipien wurden in den weiteren Komponenten ebenfalls umgesetzt.

3.9.4 Logging

Das Logging spielt eine wichtige Rolle, um den Nutzerinnen die Möglichkeit zu geben, den Testablauf während der Ausführung zu überwachen und nachträglich zu kontrollieren. Hierfür wird ein mehrstufiges, strukturiertes Logging-System eingesetzt, das eine Überwachung interner Prozesse ermöglicht. OpenTAP stellt entsprechende Funktionen über modulare Loghandler (Komponenten zum Verwalten von Protokollnachrichten) zur Verfügung, sodass Fehlermeldungen und Anomalien während und nach der Ausführung schnell

über das UI oder Protokolldateien erkannt werden können. Alle OpenTAP-Komponenten nutzen diese Logging-Funktionalität.

Für OpenTAP-unabhängige Komponenten wie den *TelegramClient* ist es dennoch wichtig, strukturierte Protokoll-Funktionen anzubieten. Zu diesem Zweck wird als internes Logging-System, *NLog*[4] verwendet. Um die erzeugten Protokollnachrichten zusätzlich zentral, innerhalb des Testmanagement-Systems zu erfassen, ohne eine direkte Kopplung zu OpenTAP spezifischen Funktionen einzugehen, wurde ein *NLog-zu-OpenTAP-Adapter* entwickelt. Dieser Adapter spiegelt Nlog-Protokollnachrichten in das OpenTAP System und lässt sich zur Laufzeit über die NLog-Konfiguration flexibel anpassen und konfigurieren.
Auf diese Weise wird eine entkoppelte Logging-Funktionalität erreicht, die unabhängig von OpenTAP funktioniert und dennoch ein zentral einseh- und verwaltbares Protokollierung ermöglicht.

3.9.5 Übersicht verwendbarer Testschritte

Mehrere Testschritte wurden entwickelt, um den Anwender*innen eine Vielzahl wiederverwendbarer Funktionen bereitzustellen. Dadurch erhalten sie die Möglichkeit, ihre eigenen maßgeschneiderten Testpläne zu erzeugen, die ihren spezifischen Anforderungen entsprechen. Diese Testschritte bieten nicht nur die Umsetzung der Anwendungsszenarien, sondern dienen auch als Grundlage für das Design und die Konzeption weiterer Experimente. Die folgende Tabelle bietet einen Überblick über eine Auswahl der entwickelten Instrumente und Testschritte, die den Benutzer*innen zur Verfügung stehen.

[4]https://github.com/NLog

Testschritt	**Beschreibung**
RunExternalPythonScript	Parametrisiert einen externen Python-Erkennungsalgorithmus, führt diesen aus und lädt die Ergebnisse zur weiteren Verwendung zurück.
MessageTraceExtraction	Führt den Extraktionsschritt des Event-Based Algorithmus für Ground-Truth-Nachrichten durch.
NetworkEventExtraction	Führt den Extraktionsschritt des Event-Based Algorithmus für Netzwerkmitschnitte durch.
BatchPreProcess	Führt eine Stapelverarbeitung auf Ground-Truth und Netzwerkmitschnitten durch.
SinglePreProcess	Vorverarbeitung einzelner Nachrichten.
JsonMessageLoad	Lädt die Nachrichten aus einer .json Nachrichtendatei.
PcapFileLoad	Liest einen Netzwerkmitschnitt als Pcap-Datei ein.
WriteOutEvents	Serialisiert Nachrichten und Ereignisse aus dem internen Datenmodell in eine .json Datei.
SendMessages	Sendet Nachrichten aus einer Nachrichtendatei über eine IM-App.
SetupMessageChannels	Konfiguriert die Gruppen und Kanäle eines IM-App-Clients.
Start/StopIMMessageLogging	Startet und stoppt die Aufzeichnung von Nachrichten eines IM-App-Clients.
ResetInstrument	Führt eine Neuverbindung eines IM-App-Clients durch.
SetupRemoteSniffer	Führt ein automatisches Setup und Konfiguration eines Netzwerk-Sniffing-Geräts durch.
StartPacketCapture	Startet die Aufzeichnung eines *Sniffing-Instruments*.
StorePacketCapture	Speichert einen Netzwerkmitschnitt zur weiteren Verarbeitung auf dem Host-Rechner.

3.10 Laufzeitsicht

Der folgende Abschnitt gibt einen Einblick in den Entwurf des Systemverhaltens. Eine generelle Betrachtung der Verhaltensweise des Systems ist an dieser Stelle nicht zielführend, da sich der konkrete Ablauf nach dem Experimentdesign richtet. Dieses wird durch den modularen Aufbau des Testbeds den Nutzer*innen überlassen, und ermöglicht es, das

Verhalten auf ihre zu untersuchenden Systemaspekte und Experimente zuzuschneiden. Aus diesem Grund wird lediglich exemplarisch die Kommunikation und das Verhalten eines Experiments betrachtet.

3.10.1 Nachrichtenaufzeichnung

Die Abbildung 3.9 bietet einen detaillierten Überblick über den Ablauf der Nachrichtenaufzeichnung gemäß Anwendungsszenario 3. Sie visualisiert die verschiedenen Schritte und Interaktionen zwischen den beteiligten Komponenten.

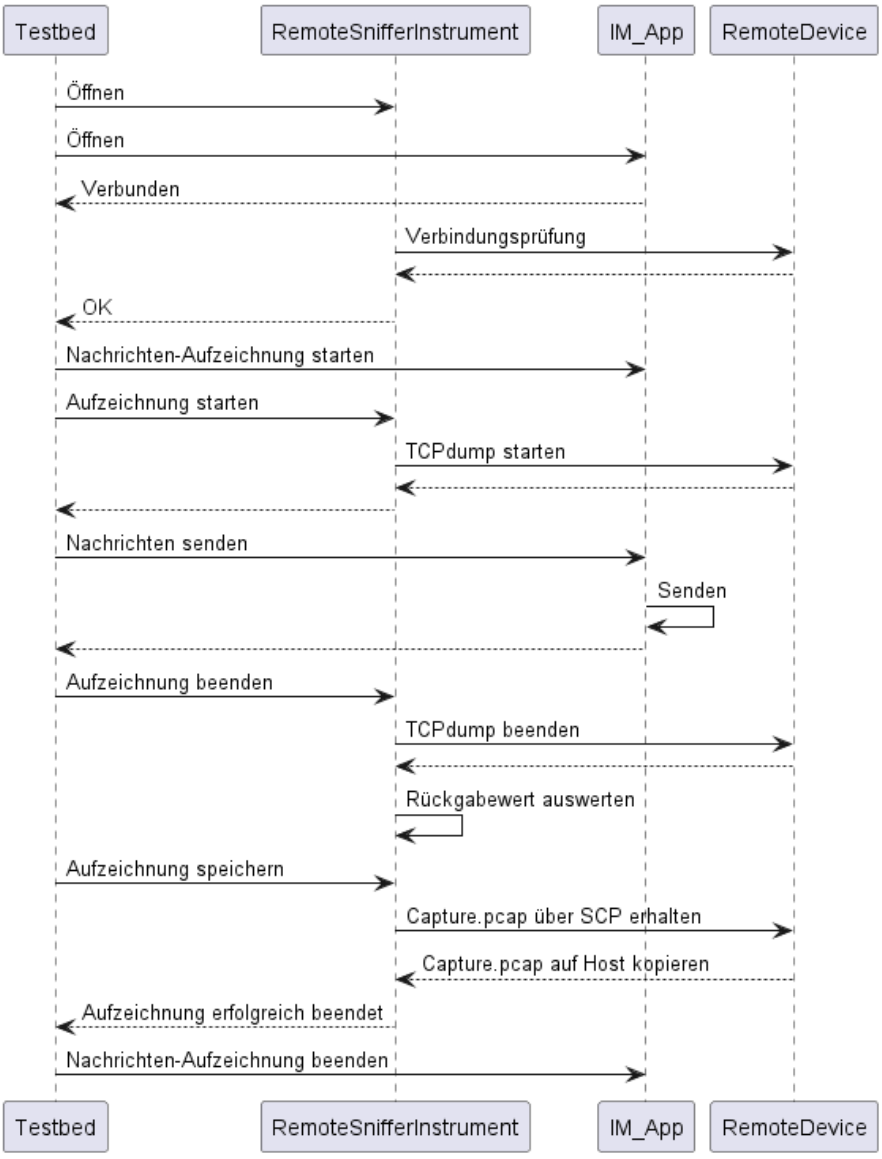


Abbildung 3.9: Sequenzdiagramm der Nachrichtenaufzeichnung

Zu Beginn werden die benötigten OpenTAP Instrumente, das *RemoteSnifferInstrument* und das Anwedungspezifische *TelegramConnectionInstrument* der IM-Anwendung, geöffnet. Dabei wird eine Verbindungsaufnahme oder -prüfung durchgeführt, um sicherzustellen, dass eine stabile Verbindung besteht. Ist die Verbindung erfolgreich, erfolgt der Übergang zur nächsten Phase der Testsequenz.

Die Funktion zur Protokollierung von Nachrichten in der *IMApp*-Komponente wird aktiviert, wodurch sämtliche eintreffenden Nachrichten der konfigurierten Accounts aufgezeichnet werden. Zum Aufzeichnen des Netzwerkverkehrs initiiert das *RemoteSnifferInstrument* die asynchrone Aufzeichnung des Netzwerkverkehrs durch *tcpdump* auf dem *RemoteDevice*. Lies sich der erfolgreiche Start der Aufzeichnung durch eine Prüfung bestätigen, signalisiert das *RemoteSnifferInstrument* ein `pass` an den Testsequenzer.

Daraufhin erfolgt das Versenden der vordefinierten Nachrichten aus dem Nachrichtenprotokoll. Sobald alle Nachrichten erfolgreich übermittelt wurden, wird das *RemoteSnifferInstrument* angewiesen, den *tcpdump*-Prozess zu beenden. Der Rückgabewert des Beendigungsvorgangs wird ausgewertet, um den Erfolg zu validieren. Anschließend erfolgt das Kopieren des aufgezeichneten Netzwerkmitschnitts auf das Kontrollsystem.

Abschließend wird die Nachrichtenprotokollierung gestoppt und die erfassten Nachrichten auf dem Kontrollrechner abgelegt.
Dieser Ablauf ermöglicht die Aufzeichnung und Speicherung von IM-Nachrichten bei paralleler Aufzeichnung Netzwerkverkehrs. Die erfassten Daten stehen somit zur weiteren Analyse und Verwendung in anderen Anwendungsszenarien bereit.
Die Prüfung nach jedem Schritt dient der Absicherung des Testablaufs, um mögliche Fehler frühzeitigt zu erkennen und den Test gegebenenfalls zu stoppen.

3.11 Zusammenfassend

Im vorliegenden Kapitel wurde zunächst die Aufgabestellung und das Ziel des Testbeds detailliert betrachtet. Aus diesen Aspekten konnten wichtige Qualitätsziele abgeleitet werden, die für die Entwicklung des Testbeds von entscheidender Bedeutung sind. Zur weiteren Kontextualisierung erfolgte die Analyse der Simulationstimuli, auf deren Grundlage anschließend die Anwendungsszenarien entwickelt wurden.

Ein wichtiger Schritt bestand darin, den relevanten Kontext und Rahmen für das Testbed zu untersuchen und einige zentrale Technologie Entscheidungen zu treffen. Zusätzlich

wurden einige wichtige Prinzipien und Entwurfsentscheidungen in der Architektur hervorgehoben, die die Grundlage für die Umsetzung des Testbeds bilden. Anhand eines Anwendungsszenarios wurde abschließend, exemplarisch, das Verhalten des Systems veranschaulicht.

4 Implementation

4.1 Einführung

In diesem Kapitel wird ausgehend vom Design die entworfenen Komponenten, wie das Telegram-Verbindungsmodul, die Sniffing-Komponente und das Erkennungsmodul in ihrer Implementierung beschrieben. Es wird ein Blick auf die Konfiguration des Testbeds geworfen, auf die verwendeten Technologien und Frameworks sowie auf den Zusammenbau und das Testen der Komponenten. Abschließend werden einige mögliche Verbesserungsmöglichkeiten beschrieben.

4.2 Verwendete Technologien und Frameworks

Im Rahmen der Implementierung der Testbed-Komponenten wurden verschiedene Technologien und Frameworks verwendet. Die Wahl der Programmiersprache fiel auf C#, da es sich um eine moderne objektorientierte Sprache handelt, die eine strukturierte Entwicklung unterstützt. Zudem ermöglicht es, die implementierten Module nahtlos in das ebenfalls in C# geschriebene Testmanagement-Framework zu integrieren.
Als Ziel Framework wurde *netstandard2.0* gewählt, um in Kombination mit OpenTAP, eine plattformunabhängige Nutzung zu ermöglichen.

Als Entwicklungsumgebung wurde *JetBrains-Rider* verwendet.
Obwohl es sich dabei um Proprietäre Software handelt, ist die Kompilierung auch mit freier Software wie Roslyn[1] möglich, und steht somit den Randbedingungen nicht entgegen. Zusätzlich wurde *VSCode* verwendet, um Bash-Skripte zu erstellen und die in Python geschriebenen Erkennungsalgorithmen zu modifizieren.

[1] https://github.com/dotnet/roslyn

4.3 Umsetzung der Kernkomponenten

Der folgende Abschnitt gibt einen Einblick in die Implementierung der drei Hauptkomponenten. Dabei werden die verwendeten Frameworks und Technologien beschrieben sowie Abweichungen vom ursprünglich geplanten Design erläutert.

4.3.1 Telegram Verbindungskompontente

Das *TelegramConnector-Modul* ermöglicht die Kommunikation mit dem Telegram-Server und umfasst Funktionen zur Konfiguration des verwendeten Accounts, sowie zum Empfangen und versenden von Nachrichten. Für die Umsetzung wurde die Tdlib Bibliothek verwendet, die das Telegram spezifische Protokoll *MTProto*, effektiv abstrahiert und neben der Authentifizierung, die Konfiguration von Kanälen und Gruppen sowie das Versenden und Empfangen von Nachrichten ermöglicht. Aus dem Analyse- und Designprozess wurden folgende Funktionen identifiziert:

- Authentifizierung anhand eines API-Schlüssels und einer Telefonnummer (einschließlich Eingabe eines 2-faktor-PINs)
- Versenden und Empfangen von Text-, Video-, Audio- und Fotonachrichten.
- Das Hinzufügen und Verlassen von Gruppen.

Für die Konfiguration von Gruppen und den Versand von Nachrichten erfordert der Telegram-Server spezifische Chat-IDs anstelle von Gruppennamen. Dies bedeutet, dass eine Namensauflösung vorgenommen werden muss.
Dieses Detail wurde der aufrufenden Komponente transparent gemacht, indem die Schnittstelle um eine Funktion zur Auflösung von Gruppennamen erweitert wurde. Auf diese Weise können die Gruppen-IDs innerhalb einer Sitzung gespeichert und verarbeitet werden, ohne für jede Nachricht eine Namensauflösung durchführen zu müssen. So konnten die API-Aufrufe bei mehrfachem Versand von Nachrichten in dieselbe Gruppe, reduziert werden. Dies ist ein wichtiges Vorgehen, da zusätzliche, synchrone Aufrufe über das Netzwerk, beispielsweise vor jedem Nachrichtenversand, zu weiteren Latenzen führen können, die sich abhängig der Server- und Netzwerklast negativ auf die Reproduzierbarkeit auswirken können.
Eine weitere Idee wäre ein Caching Mechanismus innerhalb der Komponente. Dies erzeugt allerdings ein intransparentes Verhalten, da nicht vorhersehbar ist, wann die Aufrufe zum Aktualisieren des Cache Stattfinden.

Darüber hinaus setzt die Telegram API eine Begrenzung für wiederholte API-Aufrufe fest. Die genaue Anzahl aufeinanderfolgender Aufrufe konnte nicht aus der Dokumentation ermittelt werden, weshalb eine empirisch ermittelte Verzögerung für wiederholte Aufrufe implementiert wurde.
Um potenzielle negative Auswirkungen auf die Testbed-Fidelität zu minimieren, wird der Einsatz der Verzögerung, Transparent und Nachvollziehbarkeit protokolliert. In umfangreichen Tests wurde festgestellt, dass während der kritischen Phase, (beim Versand und Empfang von Nachrichten), das Limit für die wiederholten Aufrufe und somit der Einsatz der Verzögerung nicht erreicht wurde. Dies Demonstriert die Bewahrung der Testbed-Fidelität.

Zusätzlich wurden eine Vielzahl weiterer Details behandelt, darunter die Implementierung eines Mechanismus zur Protokollierung und Analyse von fehlenden oder nicht erfolgreichen Downloads. Auf diese spezifischen Details soll an dieser Stelle nicht weiter eingegangen werden. Im Allgemeinen wurden die im Designprozess entwickelten Architekturvorgaben erfolgreich befolgt und umgesetzt.

4.3.2 Mitschneide Komponente

Die Gateway- und Mitschnittkomponente ermöglicht das Mitschneiden des Datenverkehrs im Testnetzwerk. Abbildung 4.1 veranschaulicht den allgemeinen Aufbau und die Interaktionen dieser Komponente. Im Rahmen der Analyse und des Designs wurde der Raspberry Pi 4B als geeignete Hardware für diese Komponente ausgewählt.

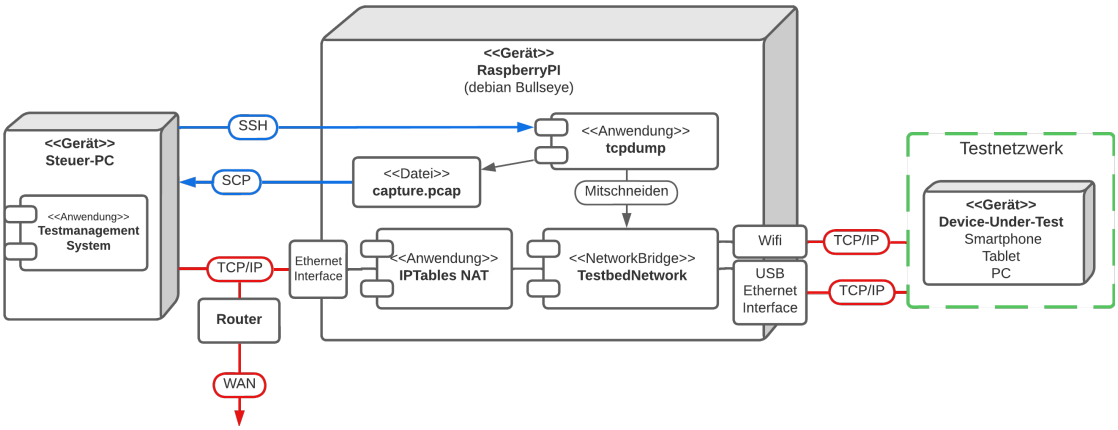


Abbildung 4.1: Deployment-Diagram des „Sniffing-Geräts“

Die Hauptfunktion des „Sniffing-Gerät“ besteht darin, dass Testnetzwerk mit dem Kontrollnetzwerk zu verbinden und den Netzwerkverkehr des Testnetzes aufzuzeichnen.
Für diese Aufgabe wird die Anwendung *iptables* verwendet, um den Netzwerkverkehr aus dem Testnetzwerk über eine NAT in das Kontrollnetzwerk weiterzuleiten. Dieser Vorgang ist notwendig, um den Testgeräten, einen Zugriff auf den Telegram Server zu ermöglichen, damit diese den zu beobachteten Netzwerkverkehr generieren können. *iptables* eignet sich als Firewall und Netzwerkfilter und schafft durch den Einsatz der NAT eine Isolationsschicht um störenden externen Netzwerkverkehr fernzuhalten.

Für die drahtlose Einbindung der DUT in das Testnetzwerk stellt der Netzwerk-Sniffer einen Wifi-Accesspoint mithilfe von *hostapd* bereit.
Aus dem Design geht hervor, dass außerdem Geräte über Wired-Ethernet verbunden werden sollen. Die verwendete Hardware (Raspberry-PI 4B) bietet allerdings nur einen Ethernet Port, welcher bereits für die Kommunikation mit dem Kontrollnetzwerk benötigt wird.
Aus diesem Grund wird ein USB zu Ethernet Adapter verwendet, welcher ein zweites (Gigabit fähiges) Ethernet-Interface bereitstellt.
Diese beiden Netzwerkschnittstellen werden mithilfe der Anwendung *systemd-networkd*, in einer Netzwerkbrücke zusammengefasst. Verbundene Testgeräte erhalten anschließend von einem DHCP Server, IP-Adressen aus dem lokalen Testnetzwerk (192.168.4.0/24).
Das dritte, bereits erwähnte Netzwerkinterfaces des Raspberry-Pi (eth0) wird als Kontrollnetzwerk-Port verwendet. Die IP-Adresse dieses Interfaces wird von einem externen DHCP Server bezogen, um einen flexiblen Einsatz in verschiedenen Netzwerkumgebungen zu ermöglichen.

Durch diesen Aufbau fließt der gesamte Netzwerkverkehr des Testnetzes über den Raspberry-Pi. Für das Mitschneiden des Verkehrs wird die Anwendung *tcpdump* installiert.

Um die vorgestellten Konfigurationsschritte zu automatisieren, wird ein Installationsskript implementiert, welches wahlweise Interaktiv, die Erzeugung aller Konfigurationsdateien und Setupschritte auf dem Gerät vornimmt. Diese Automatisierung wurde vorgenommen, um den Setup-Prozess zu vereinfachen und die Fehleranfälligkeit zu reduzieren, mit dem Ziel die Reproduzierbarkeit zu unterstützen.

Um die Ausführung des Setups ebenfalls zu automatisieren werden die folgenden Schritte durch einen Testschritt bereitgestellt. Zunächst wird das benötigte Konfigurationsskript mittels SCP auf das Gerät geladen. Daraufhin wird das Skript ausgeführt, und nach einer erfolgreichen Konfiguration, die erzeugten Wi-Fi und Account-Zugangsdaten an den

Hostrechner übertragen.
Die Kommunikation findet dabei verschlüsselt über SSH statt.
Die SSH.NET Bibliothek wird zur Abstraktion dieser Protokolle verwendet, um plattformübergreifend die erforderlichen Funktionen bereitzustellen.

Nach der automatischen Konfiguration stellt das OpenTAP-Instrument *RemoteSniffer-Instrument* den Testschritten drei zentrale Funktionen zur Verfügung.

1. Starten einer Aufzeichnung (wobei ein Aufzeichnungsfilter und das Netzwerkinterface spezifiziert wird)
2. Stoppen einer Aufzeichnung
3. Speichern einer Aufzeichnung (Wobei die Aufzeichnungsdatei auf den Kontroll-Rechner kopiert wird.)

Für den Start der Aufzeichnung wird *tcpdump* als Hintergrundprozess ausgeführt. Es folgt eine kurze Überprüfung, um einen erfolgreichen Start sicherzustellen. Dies dient dazu, fehlerhafte Konfigurationen wie z.B. angegebene, aber unbekannte Netzwerkinterfaces zu erkennen. Die Ausgabe von `Stdout` und `Stderr` wird in eine Datei umgeleitet, um sie anschließend auswerten zu können.

Während der Aufzeichnung wird keine aktive Verbindung zum Sniffing-Gerät aufrechterhalten, sondern lediglich die Erreichbarkeit über Pings überprüft.

Zum Beenden der Aufzeichnung werden alle laufenden *tcpdump*-Prozesse beendet und die generierte Protokolldatei ausgewertet. Falls die Aufzeichnung keine Pakete erfassen konnte, oder ein anderer Fehler aufgetreten ist, wird dies den Nutzer*innen signalisiert. Auf diese Weise wird eine zuverlässige Aufzeichnung des Datenverkehrs gewährleistet und potenzielle Probleme bei der Aufzeichnung können frühzeitig identifiziert werden.

4.3.3 Erkennungskomponente

Zur Integration bestehender Erkennungsalgorithmen, wurde ein Testschritt entwickelt, der als Adapter für die vorliegenden Python-Skripte fungiert. Dieser Testschritt ermöglicht die Ausführung des spezifizierten Skripts unter Berücksichtigung der erforderlichen Parameter, wie den Pfaden zu den Eingangsdateien und den Beobachtungsintervallen.

Die vorliegenden Python-Skripte wurde entsprechend modifiziert, um die erzeugten Ergebnisse in einer JSON-Datei zu speichern. Nach Abschluss des Skripts übernimmt der genannte Testschritt die Aufgabe, die Ergebnisdatei zurück in das Testmanagement-System zu laden. Dadurch wird eine nahtlose Weiterverarbeitung und Analyse der Ergebnisse ermöglicht.

Abschließend

In diesem Abschnitt wurden anhand der Telegram-Verbindungs-, Mitschneide- und Erkennungskomposte einige implementationsspezifische Entscheidungen und Details erläutert.

4.4 Konfiguration

Im nachfolgenden Abschnitt wird erläutert, wie das Testbed konfigurierbare Parameter verwaltet und zur Verfügung stellt.

Viele Testbed-Komponenten wie der *TelegramClientConnector* enthalten variations- und Konfigurationsparameter, die maßgeblich das Verhalten beeinflussen können. Einige Module müssen zudem mit Zugangsdaten konfiguriert werden.
Zur zentralen Verwaltung dieser Parameter wurde eine interne Datenstruktur angelegt, die implementationsspezifische Konstanten und Werte enthält. Für Laufzeit-Einstellungen hingegen wurden die konfigurierbaren Parameter in die Testschritte und Instrumente ausgelagert.
Durch eine elegante Schnittstelle über C#-Annotations, lassen sich die Parameter in OpenTAP registrieren. Dies ermöglicht eine intuitive Modifikation der Einstellungen über das UI, in welches Beschreibungen und Hinweise eingeblendet werden können, und sich Einstellungen auch über Testschritten hinweg verknüpfen lassen.
In Abbildung 4.2 lässt sich der Einsatz anhand eines Testschritts zum Versenden von Nachrichten erkennen.

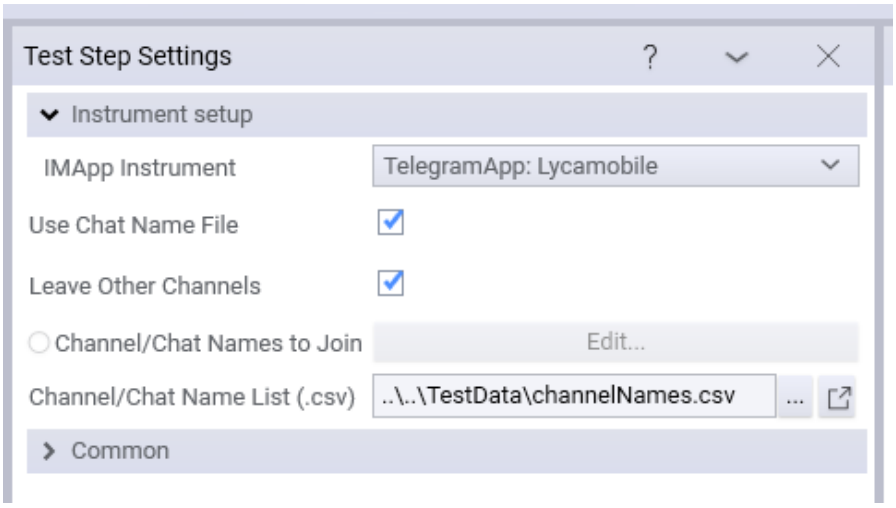


Abbildung 4.2: Einstellungen eines Testschritts

Der konfigurierte Testablauf, einschließlich der Testschritte, Instrumente und Einstellungen, lässt sich als sogenannter „.TapPlan“ speichern. Das Testsetup wird durch OpenTAP als .XML serialisiert und gespeichert.
Dies ermöglicht die wiederholbare Durchführung gespeicherter Testabläufe auf unterschiedlichen Systemen.

Telegram Accounts

Für die Verwendung des *TelegramConnector*-Moduls, muss dieses sich zunächst beim Server als gültiger Telegram-Client Authentifizieren. Hierfür wird ein aktivierter Telegram-Account benötigt, um die angeforderten API-Authentifizierungsdaten generieren zu können.
Um diese Anmelde-Informationen getrennt von den Testschritt-Einstellungen zu speichern, wurden diese in eine separate JSON Konfigurationsdatei ausgelagert. Dadurch ist es möglich, die Testschritte gemeinsam mit dem „.TapPlan“ auszuliefern, ohne dass dabei sensible Authentifizierungsdaten veröffentlicht werden. Auch die Account-Konfigurationsdatei lässt sich so, unabhängig des Setups auf andere Systeme portieren.

Die Konfigurationsdatei unterstützt dabei mehrere Account-Profile, welche sich durch eine einfache Struktur, intuitiv konfigurieren lassen. Das *TelegramClientConnector*-Instrument verwaltet und validiert diese Konfigurationsdatei und stellt die geladenen Profile über ein Dropdown-Menü in der Benutzungsoberfläche bereit (Siehe Abbildung 4.3).

Dies erleichtert den Aufbau komplexerer Tests, insbesondere wenn mehrere Accounts benötigt werden. So lässt sich z.B. über einen Account der Nachrichtenversand simulieren, während ein weiterer den Nachrichtenempfang des DUT-Accounts protokolliert.

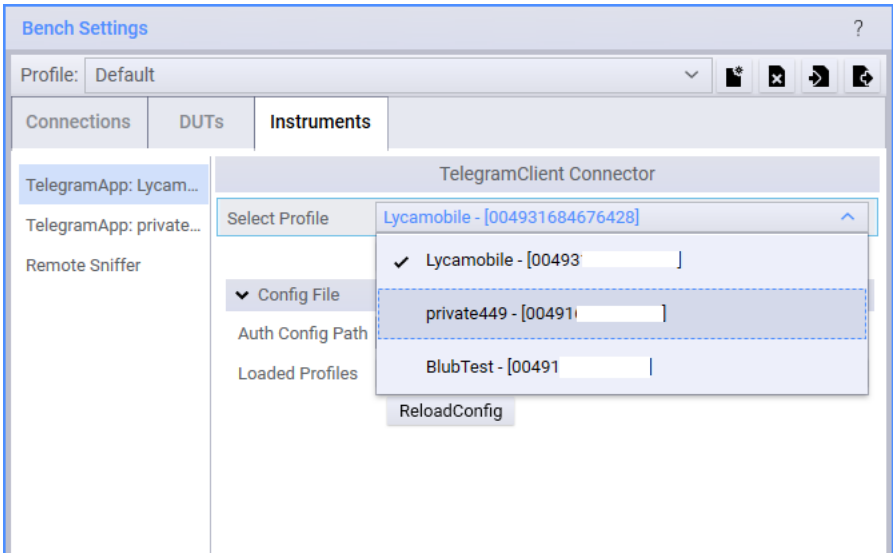


Abbildung 4.3: Auswählbare Account-Profile

Abschließend

In diesem Abschnitt wurde beschrieben, wie das Testbed konfigurierbare Parameter verwaltet, und über das UI bereitstellt. Zudem wurde erläutert wie die benötigten Authentifizierungsdaten für den Telegram-Client in einer separaten Konfigurationsdatei ausgelagert werden.

4.5 Zusammenbau der Komponenten

Im Folgenden wird ein Blick auf die Komposition und Abhängigkeiten der Komponenten geworfen.

Bei der Zusammenstellung der Komponenten wurde darauf geachtet, dass sie in separaten Projekten verwaltet werden, um eine unabhängige Entwicklung und modulare Verwendung zu gewährleisten. Einige projektübergreifende Datenstrukturen und Tools wurden in einem separaten „Commons“-Projekt ausgelagert und zentral verwaltet.

Die resultierende Projektstruktur zeigt einige Abhängigkeiten zwischen den Komponenten, wobei darauf geachtet wurde, möglichst wenige transitive und keine zyklischen Abhängigkeiten zu erzeugen. Das „Commons“-Projekt enthält dabei wichtige Datenstrukturen und Hilfsklassen, die von nahezu allen Komponenten genutzt werden, um ein einheitliches Datenmodell zu gewährleisten.

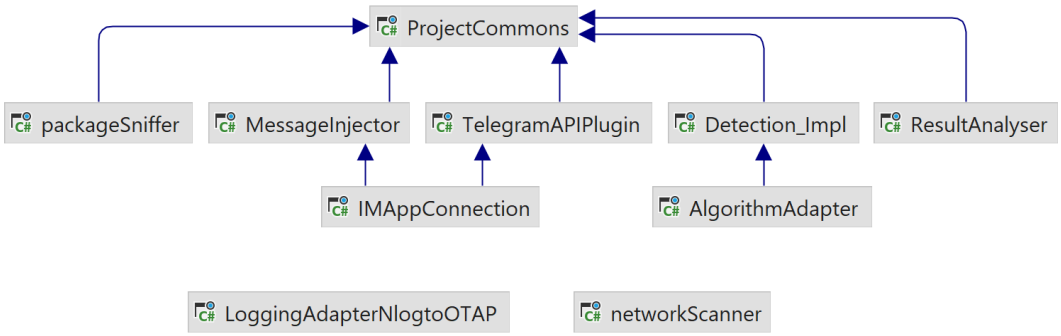


Abbildung 4.4: Abhängigkeitsgraf der erstellten Komponenten

Aus dieser Struktur heraus werden 11 separate DLLs kompiliert, die sich weitgehend unabhängig voneinander in das Testmanagement-System einbinden lassen.

Die klare Trennung der Komponenten bereits in der frühen Entwicklungsphase ermöglichte die rechtzeitige Erkennung und Vermeidung einiger Abhängigkeiten. Dadurch ist es möglich, die einzelnen Module in ihren eigenen Testprojekten unabhängig voneinander zu testen und weiterzuentwickeln.

OpenTAP ermöglicht zudem das Verpacken dieser Komponenten in sogenannte „Tap-Packages“. Dadurch können die einzelnen Komponenten über den quelloffenen OpenTAP Package Manager verteilt und installiert werden.

4.6 Test und Validierung

Im folgenden Abschnitt wird die Validierung des Gesamtsystems und der entwickelten Komponenten, durch Unit-Tests sowie speziellen Testplänen beschrieben.

Die Validierung der entwickelten Komponenten und des Gesamtsystems spielen eine entscheidende Rolle, um deren Funktionalität, Zuverlässigkeit und Reproduzierbarkeit zu gewährleisten.

Die Hauptkomponenten wurden durch Unit-Tests abgesichert, wobei der Schwerpunkt auf der korrekten Implementierung lag. Insgesamt wurden 17 ausführliche Unit-Tests für den Telegram-Client und die Aufzeichnung von Nachrichten erstellt.

Um die Reproduzierbarkeit und Fidelität sicherzustellen, wurde ein besonderes Augenmerk auf die Tests zum Versenden und Empfangen von Nachrichten gelegt.
Dabei wurde die zeitliche Variation zwischen dem Versand einer Nachricht durch einen Client, und dem Empfangen dieser Nachricht auf einem weiteren Account untersucht. Eine Grenze von 3 Sekunden wurde dabei als Erfolgskriterium für diesen Test festgelegt. Obwohl dies relativ hoch erscheinen mag, haben die Versuche gezeigt, dass insbesondere Video-Nachrichten teilweise eine erhebliche, zeitliche Latenz zwischen Versand und Empfang aufweisen.

Die Implementierung der Event-Based Erkennungsroutine wurde durch 10 Unit-Tests abgesichert, wobei der Schwerpunkt auf der korrekten Funktionsweise der Teilfunktionen lag, die beispielsweise die Aufteilung in simulierte Beobachtungsintervalle übernehmen.

Die weiteren Komponenten wurden in OpenTAP, zunächst separat und anschließend in ihrer Integration mit dem Gesamtsystem getestet. Für die Durchführung wurden spezielle Testpläne mit Testdaten erstellt, die teilweise eine manuelle Auswertung erforderten.

Abschließend

In diesem Abschnitt wurde die Validierung der entwickelten Komponenten und des Gesamtsystems durchgeführt, indem umfangreiche Unit-Tests und spezielle Testpläne mit Testdaten verwendet wurden.

4.7 Auswertung

Der vorliegende Abschnitt dient einer kurzen abschließenden Beurteilung des durchgeführten Implementationsprozess.

4.7.1 Bewertung der Implementierung

Grundsätzlich verlief die Implementierung größtenteils reibungslos, obwohl einige Details, die im Designprozess nicht vorhersehbar waren, zu Änderungen geführt haben. Ein Beispiel hierfür ist das Hinzufügen der Namensauflösungsfunktion für den Versand von Nachrichten und der Gruppenkonfiguration.
Es gab jedoch auch andere Details, die zusätzlichen Entwicklungsaufwand erforderten und möglicherweise durch eine zielführende Planung, hätten vermieden werden können.

Ein solches Beispiel ist die Implementierung einer plattformunabhängigen Methode zur Erfassung des 2-Faktor-PINs über ein zusätzliches Eingabefenster. Möglicherweise hätte dies durch die Verwendung der OpenTAP Benutzungsoberfläche leichter umgesetzt werden können.

Auch die Implementierung verschiedener Hilfskomponenten, die eine umfangreiche „Fluent-API“ vorsahen, erwies sich an dieser Stelle als wenig zielführend. Obwohl dies die Wartung und Verwendung der Komponenten vereinfachte, war es für den produktiven Einsatz des Testbeds nicht erforderlich.

Zudem wurden zahlreiche Auswertungs-, Analyse- und Erkennungsroutinen entwickelt, die lediglich der Verwaltung und Analyse von Quell- und Ergebnisdateien dienten. Obwohl sie sich während der Entwicklung als praktisch erwiesen haben, sind sie für den produktiven Einsatz des Testbeds wahrscheinlich vernachlässigbar.

Die Entscheidung, die Konfigurationsdatei des Telegram-Clients auszulagern, erwies sich dagegen als positiv, obwohl dies ebenfalls einen zusätzlichen Implementierungsaufwand bedeutete, der nicht ursprünglich geplant war.
Verbesserungspotenzial wird in der Verschlüsselung dieser Konfigurationsdatei gesehen, um den Forschenden die Möglichkeit zu geben, ihre spezifischen Setups auf verschiedenen Systemen sicher abzulegen.

Zusätzlich lässt sich Potenzial für Verbesserungen im Bereich des Testens identifizieren. Um die Erweiterbarkeit und Wartung des Testbeds zu optimieren, wäre es von Vorteil, die Automatisierung der Integrations- und Komponententests weiter voranzutreiben. Obwohl dies aufgrund des Einsatzes von Hardware, die teilweise physische Verbindungen erfordert, nur begrenzt möglich ist, würde eine automatisierte Durchführung von Tests dennoch einen Mehrwert bieten. Hier wird Verbesserungspotenzial erkannt.

4.7.2 Abschließend

In diesem Kapitel wurden zunächst die verwendeten Technologien und Frameworks vorgestellt. Es folgte ein Blick auf die Implementierung der Telegram-Verbindungskomponente, der Mitschneide- und Erkennungskomponente, und die damit einhergegangenen Entscheidungen. Daraufhin wurde betrachtet wie das Testbed mit einstellbaren Parametern und Authentifizierungsdaten umgeht.
Im nächsten Abschnitt wurde betrachtet wie die Komponenten in einzelne Projekte unterteilt werden und welche Abhängigkeiten dort bestehen. Daraufhin wurde auf die durchgeführten Tests und Validierungsmaßnahmen eingegangen. Zu guter Letzt folgte eine kurze Bewertung des Implementationsprozess und ein Ausblick über mögliche Verbesserungen.

5 Evaluation

Im weiteren Verlauf werden zunächst die Anwendungsszenarien anhand von Proof-of-Concept-Experimenten demonstriert, wodurch ein erster Einblick in die Fähigkeiten des Testbeds ermöglicht wird. Anschließend werden die Metriken der Qualitätsziele vorgestellt, um eine detaillierte Evaluierung in der anschließenden Diskussion zu ermöglichen.

5.1 Anwendungsszenarien

5.1.1 Szenario 1

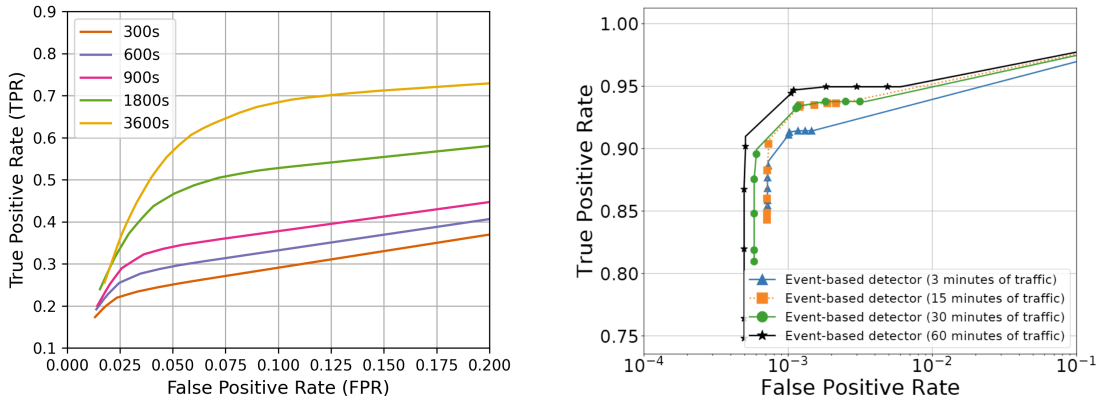
Ein zentraler Bestandteil des Angriffs besteht darin, den Erkennungsalgorithmus auf die beobachteten Daten anzuwenden, um einen Korrelationswert zu erhalten, der die Entscheidung für eine der Hypothesen ermöglicht.
Die Effektivität des Erkennungsalgorithmus ist entscheidend für den Erfolg des Angriffs, weshalb ein Anwendungsszenario zur Durchführung des Algorithmus entwickelt wurde.

Das Testbed bietet dabei den Rahmen, um einen solchen Erkennungsalgorithmus auszuführen und die erzeugten Daten zu analysieren und zu verarbeiten.
Die Entwicklung oder Verbesserung des Algorithmus selbst, ist jedoch nicht Teil dieser Arbeit, obwohl das Testbed die Möglichkeit dazu bieten kann.

Im Folgenden wird die Durchführung des ersten Anwendungsszenarios anhand eines Proof-of-Concept Experiments demonstriert.

Als Eingangsdaten wird der bereits vorgestellte Referenzdatensatz verwendet, der 852 Stunden Telegram-Aufzeichnungen enthält. Der verwendete Erkennungsalgorithmus ist eine angepasste Version der Github-Implementierung, auf welche die Publikation [10] verweist.

Zur Durchführung des Experiments wird ein Testplan über das Testmanagement UI geladen und mit den Ein- und Ausgabepfaden konfiguriert. Nach dem Start des Testsequenzers lädt dieser zunächst alle benötigten Dateien und überführt diese in ein vereinfachtes, standardisiertes Format. Im nächsten Schritt wird ein extern eingebundenes Python-Script gestartet, welches die eigentliche Erkennung durchführt.
Nach Abschluss werden die Ergebnisse automatisiert wieder in das Testmanagement System hineingeladen und dort textuell dargestellt. Zur genaueren Analyse und Vergleichbarkeit werden die Ergebnisse in der folgenden Grafik den veröffentlichten Werten aus [10] gegenübergestellt.



(a) Veröffentlichte Implementierung

(b) Veröffentlichte Werte aus [10]

Abbildung 5.1: Vergleich der Erkennungsleistung

Die Ergebnisse des durchgeführten Experiments, wie in Abbildung 5.1a dargestellt, zeigen deutliche Abweichungen von den veröffentlichten Werten. Insbesondere in Bezug auf die Richtig-Positiven (TP) und Falsch-Positiven (FP) Werte, wurden signifikante Unterschiede festgestellt. Im Vergleich zu den erwarteten Werten von $TP = 0,95$, bei $FP = 10^{-3}$, gemäß der Veröffentlichung von [10], erreicht die vorliegende Implementierung beispielsweise nur $TP = 0,7$, bei $FP = 0,1$. Diese Diskrepanzen erstrecken sich über alle beobachteten Intervalle, wobei kürzere Intervalle noch niedrigere TP Werte aufweisen und daher als kaum brauchbar eingeschätzt werden können.

5.1.2 Szenario 2

Um möglichst realitätsnahe Testdaten zu generieren, ermöglicht dieses Anwendungsszenario das Erzeugen echter Endbenutzer*innenaktivitäten, durch das Mitschneiden von Nachrichten aus IM-Kanälen. Dies soll im folgenden Versuch als Proof-of-Concept, anhand der 20 reichweitenstärksten Telegram-Kanäle im Iran, demonstriert werden.

Zunächst wird eine Liste der Telegram Kanal IDs benötigt. Die IDs können z.B. als URL (`https://t.me/groupname`) oder als `@<name>` verarbeitet werden und stammen für diesen Versuch von der Plattform *TGStat*[1].

Außerdem benötigt, wird ein Account der IM-Anwendung, in diesem Fall Telegram. Es muss ein API-Key erzeugt werden, der in der Testbed spezifischen Konfigurationsdatei (5.1), für das `TelegramConnectorModul` hinterlegt wird. Für die Erstellung eines Accounts ist eine SIM-Karte erforderlich.

```
{  "auths": [{
    "ApiId": 20345xxxx,
    "ApiHash": "1c60d7d7xxxxx",
    "PhoneNumber": "00493xxxxxxx",
    "Name": "Lycamobile",
    "DBlocation": ".\\tg_profiles\\db_Lycamobile"}]
}
```

Listing 5.1: *TelegramConnector* Konfiguration

Zur Ausführung wird ein bereits konfigurierter Testplan geladen. Dieser enthält eine Sequenz an Testschritten, die das Setup des Testaccounts vornehmen (Beitreten selektierter Kanäle), die Aufzeichnung der Nachrichten starten und nach einer vorgegebenen Zeit die Aufzeichnung stoppen und den Test beenden. Empfangene Nachrichteninhalte wie Fotos und Videos werden zudem automatisch heruntergeladen und zur weiteren Verwendung separat gespeichert.

Für diesen Durchlauf wurde ein Zeitraum von 4 Std. gewählt. Nach Ablauf dieser Zeit stand neben einem Nachrichtenprotokoll auch ein Sitzungsprotokoll des Testablaufs zur Verfügung.
In 4 Ebenen (Error, Warn, Info, Debug), stellt dieses Protokoll, detailliert alle Ereignisse

[1] https://tgstat.com/

der Hintergrundprozesse sowie Warnungen oder Anomalien des Testablaufs dar.
Das Nachrichtenprotokoll hingegen, enthält neben Metainformationen wie die aufgezeichneten Gruppen-IDs und den Zeitpunkt von Start und Stopp der Aufzeichnung, alle Nachrichten die in dem beobachteten Zeitraum und Kanälen aufgezeichnet wurden. Eine Nachricht wird dabei mit ihrer Größe (Größe der Datei oder des Texts), dem Zeitstempel als `Unix Time`, dem Typ der Nachricht (Text, Foto, Audio, Video) und einem Flag für ein/oder ausgehende Nachrichten, gespeichert.
Textnachrichten enthalten außerdem den vollständigen Text in Unicode. Handelt es sich um ein Foto, Video oder Audio, wird der entsprechende Pfad zur lokalen Datei gespeichert. Das Nachrichtenprotokoll ist dabei als `JSON`, vollständig maschinenverarbeitbar.

Von 01:40 bis 5:40 der Iran Standard Time (IRST) wurden in diesem Experiment insgesamt 165 Nachrichten aufgezeichnet. Tabelle 5.1 bietet einen Einblick in das Ergebnis dieser Aufzeichnung.

Typ	**Anzahl**	**Volumen**	**Kleinste Nachricht**	**Größte Nachricht**
Foto	54 (32.73%)	4,8 MB (30.13%)	37,1 KB	208,7 KB
Audio	0 (0%)	0 B (0%)	0 B	0 B
Video	4 (2.42%)	11,1 MB (69.80%)	695,7 KB	7,8 MB
Text	107 (64.85%)	11,4 KB (0.07%)	4 B	978 B

Tabelle 5.1: 4-stündige Aufzeichnung der 20 größten Iranischen Telegram Kanäle

Die Analyse der Zusammensetzung und Größe der Nachrichten zeigt, dass Textnachrichten, die mehr als 60 % der Gesamtzahl ausmachen, in Bezug auf die Größe kaum einen signifikanten Datendurchsatz bedeuten. Die durchschnittlichen, maximalen und minimalen Größen, stimmen weitgehend mit den von [10] veröffentlichten Werten überein.

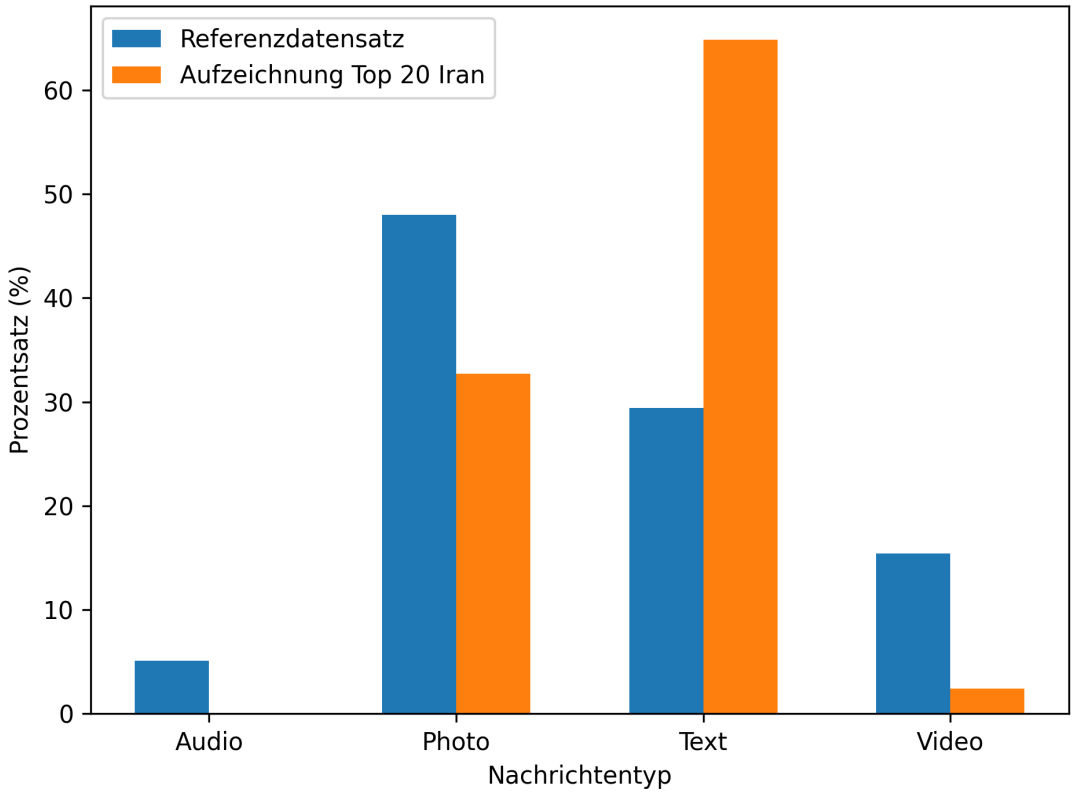


Abbildung 5.2: Nachrichtenzusammensetzung des erzeugten Datensatzes im Vergleich

Ein anderes Bild ergibt sich jedoch bei der Betrachtung der Nachrichtenzusammensetzung. Abbildung 5.2 zeigt deutliche Unterschiede in der Anzahl von Textnachrichten, Fotos und Videos. Im veröffentlichten Datensatz machen Textnachrichten lediglich 30 % aus, während Video-, Foto- und Audionachrichten deutlich häufiger vertreten sind.

5.1.3 Szenario 3

Das Anwendungsszenario 3 soll unter Einbindung physischer Endgeräte, aufgezeichnete Nachrichten abspielen, während die Netzwerkdaten des Testnetzwerks aufgezeichnet werden. Als Proof-of-Concept Demonstration, wurde für dieses Experiment ein *Samsung S5 mini* mit Android 6.0.1, als Endgerät verwendet.
Das Smartphone wurde zunächst mit der Anwendung (Telegram 9.6.7) und einem Tes-

taccount aufgesetzt. Zusätzlich wurde das Gerät so konfiguriert, dass der Bildschirm dauerhaft angeschaltet blieb.

Für das Testnetzwerk benötigt wird außerdem ein „Sniffing-Gerät“. Für diesen Test wurde ein Raspberry Pi 4 B, über einen bestehenden Testplan automatisch konfiguriert. Da das Gerät für das Testnetzwerk eine NAT durchführt, wurde es WAN-seitig per Ethernet mit dem Kontrollnetzwerk verbunden, welches eine Internetverbindung besitzt. Auf der lokalen Testnetzwerk-Seite wurde das Endgerät per WLAN verbunden.

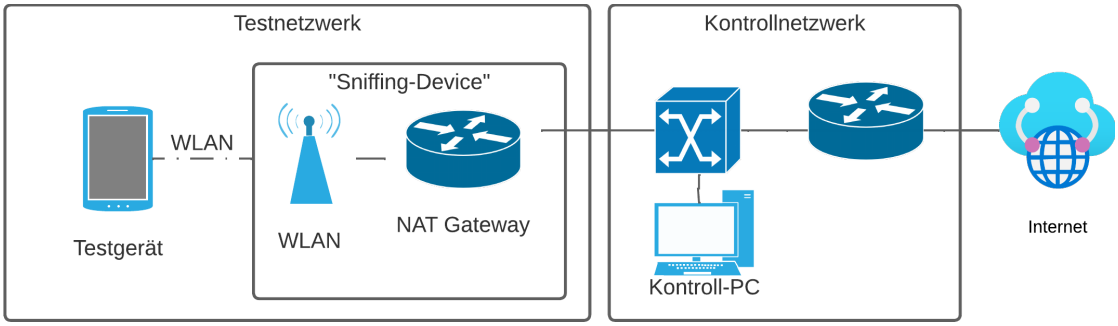


Abbildung 5.3: Testaufbau Anwendungsszenario 3

Die konfigurierte Testinfrastruktur ermöglicht nun die Durchführung des Anwendungsszenarios 3.
Nach dem Start des Testplans auf dem Kontroll-PC, wird zunächst automatisiert der Telegram-Testaccount des Smartphones mit dem nötigen Testkanal konfiguriert. Das „Sniffing-Device“ wird mit einem Mitschnittfilter ausgestattet, der bei der Aufzeichnung, Pakete mit den IP-Adressen der Telegram-Server selektiert.

Im nächsten Schritt wird die Aufzeichnung des Netzwerkverkehrs gestartet. Anschließend erfolgt das Abspielen der aus Szenario 2 aufgezeichneten Nachrichten in Echtzeit. Nach 4 Stunden steht ein neues Nachrichtenprotokoll zur Verfügung, welches alle 165 empfangenen Nachrichten, sowie die aktuellen Zeitstempel enthält.
Neben diesen Telegram Nachrichten steht die Mitschnitt-Datei bereit, die den aufgezeichneten Testnetzwerkverkehr beinhaltet.
In diesem Versuch wurden ca. 92.000 Pakete mit einer Größe von durchschnittlich 1020 Byte erfasst. Insgesamt entspricht dies 95 MB. Diese Daten stehen nun zur Analyse und Weiterverarbeitung, in z.B. Anwendungsszenario 1 zur Verfügung.

Um die Auswirkungen verschiedener Endgeräte und Netzwerkkonfigurationen zu untersuchen, ermöglicht dieses Anwendungsszenario zudem die Einbindung weiterer Geräte.

In diesem Fall wurde das gleiche Nachrichtenprotokoll verwendet, um Daten an einen anderen Telegram-Client auf einem, mit Ethernet angebundenen PC, zu senden. Dabei handelt es sich um die Desktop-App (Version 4.8.3) unter Windows 10. Es wurde festgestellt, dass sich das Verhalten im Vergleich zu anderen Geräten unterscheidet. Die Verteilung der Paketgrößen in den beiden Aufzeichnungen wird in Abbildung 5.4 dargestellt.

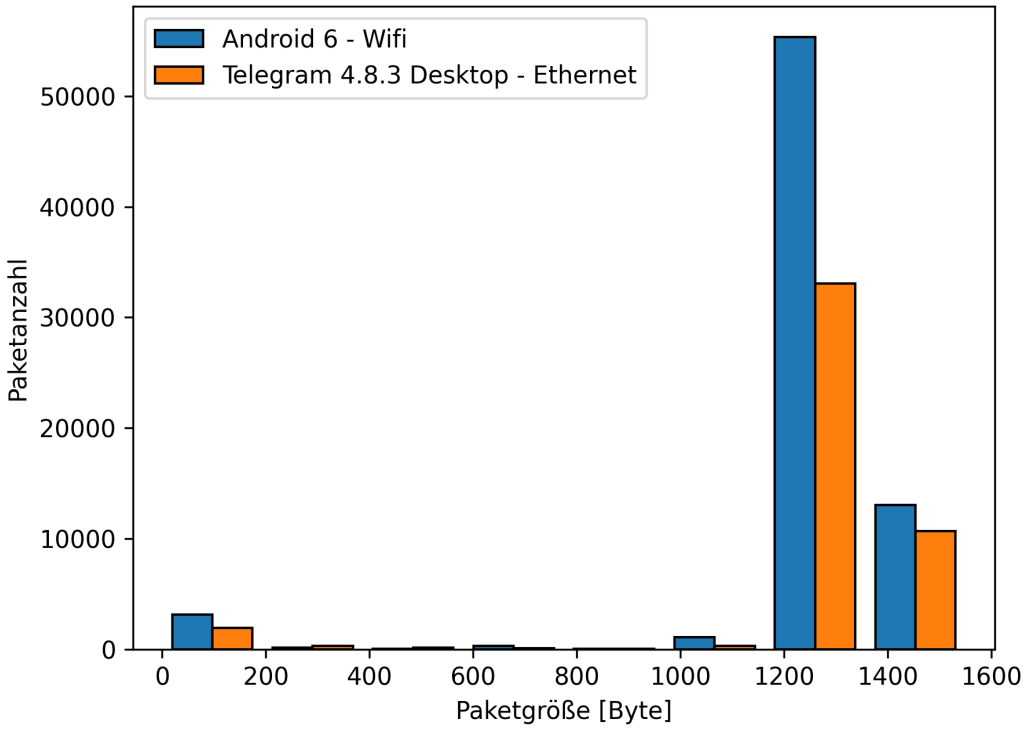


Abbildung 5.4: Aufzeichnung verschiedener Endgeräte: Verteilung der Paketgrößen

Im vorliegenden Datensatz ist ein geringer Anteil sehr kleiner Nachrichten (<200 Byte) erkennbar, die somit in die Größenordnung der Protokollnachrichten fallen, wie sie von [10] beschrieben wurden. Die Desktop-Aufzeichnung zeigt hier eine geringfügig geringere Anzahl empfangener Pakete im Vergleich.

Die größte Häufung von Paketen ist bei einer Größe von etwa 1200 Byte zu beobachten, wobei die Desktop-Aufzeichnung nur etwa 60 % der Pakete der Android-Aufzeichnung

enthält. Etwa 1000 Pakete wurden mit einer Größe von ungefähr 1500 Byte empfangen, wobei die Desktop-Aufzeichnung etwa 20 % weniger Pakete enthält.

Dieser Trend zieht sich über die gesamte Grafik hinweg, wodurch deutlich wird, dass die Desktop-Aufzeichnung insgesamt weniger Daten empfangen hat. Der Datenempfang des Desktop-Clients beträgt dabei rund 60 MB im Gegensatz zu den 95 MB des Android Smartphones.

5.1.4 Abschließend

Durch die Durchführung dieser Proof-of-Concept Experimente konnte erfolgreich gezeigt werden, dass die Anwendungsszenarien umsetzbar sind. Die gewonnenen Ergebnisse stellen jedoch nur einen ersten Einblick dar und bedürfen einer eingehenderen Betrachtung im Diskussionskapitel.
Dort werden die Experimente und ihre Ergebnisse im Kontext, der im Anschluss vorgestellten Metriken, für die Qualitätsziele analysiert und evaluiert, um ein umfassendes Verständnis der Leistung und Funktionalität zu gewinnen.

5.2 Vorstellung der Evaluationsmetriken

5.2.1 Fidelität

Fidelität ist ein wichtiges Qualitätsziel bei der Entwicklung und Evaluation von Testbeds. In der Literatur wird die Fidelität als Maß dafür beschrieben, wie genau das Testbed die Umgebungsdaten und -werte präsentiert. Typischerweise hat der Einsatz physischer Geräte positiven Einfluss auf die Fidelität [3].

Um die Fidelität zu untersuchen, werden verschiedene Aspekte des Testbeds betrachtet. Ein wichtiger Faktor ist die Datensatzkonsistenz, die die Korrektheit und Genauigkeit der vom Testbed erzeugten Daten betrifft. Es wird analysiert, wie viele Fehler das Testbed bei der Datenerzeugung generiert, und wie präzise es dabei arbeitet.

Ein weiterer Aspekt ist die Verwendung von Hardware. Hier wird untersucht, welchen Einfluss und welche Relevanz die Verwendung von physischen Geräten auf die erzeugten Daten hat.

Die Bewertung der Fidelität eines Testbeds ist entscheidend, um sicherzustellen, dass die durchgeführten Experimente und Simulationen die realen Umgebungsbedingungen möglichst genau widerspiegeln.
Eine hohe Fidelität gewährleistet, dass die erzielten Ergebnisse und Erkenntnisse auf die tatsächliche Umgebung übertragbar sind und somit eine aussagekräftige Grundlage für weiterführende Untersuchungen und Optimierungen bieten [3].

5.2.2 Reproduzierbarkeit

Das Qualitätsziel der Reproduzierbarkeit bezieht sich auf die Fähigkeit des Testbeds, bei wiederholter Durchführung mit gleichen Eingabedaten auch statistisch ähnliche Ergebnisse zu erzielen. Es wird angestrebt, dass exakt gleiche Aufbauten zu identischen oder zumindest statistisch konsistenten Ergebnissen führen.
Die Erreichung dieses Ziels kann gemäß [3] durch eine sorgfältige Dokumentation und Aufzeichnung der Konfiguration erreicht werden. Dies ermöglicht es anderen Forschenden den Experimentaufbau nachzuvollziehen, um so die Experimente zu wiederholen und zu validieren.
Es ist daher von Bedeutung, die Variationen zu untersuchen, die bei mehrfacher Durchführung der Anwendungsszenarien auftreten können.

So soll für das Anwendungsszenario 3 untersucht werden, welche Variationen der Korrelationsrate durch das Testsetup erzeugt werden. Aufgrund der Anfälligkeit für externe Einflüsse, die durch einen realitätsnahen Testaufbau entstehen, ist zu erwarten, dass signifikante Variationen über mehrfache Durchführungen auftreten können.
Es wird angenommen, dass dies durch Faktoren wie Wi-Fi-Interferenzen, Netzwerk- und Server-Auslastung sowie Netzwerklatenz verursacht wird.

Bei mehrfacher Anwendung der Erkennungsalgorithmen hingegen, wird eine geringe, bis nicht messbare Variation erwartet. Dies wird angenommen, da die verwendeten Algorithmen statistische Metriken verwenden, die nach einer ersten Einschätzung bei gleichen Eingabewerten auch gleiche Ausgaben erzeugen.

Abschließend ist zu untersuchen, welche Variationen bei der Aufzeichnung von Nachrichten auftreten können und wie sich diese auf die Reproduzierbarkeit der Ergebnisse auswirken.

5.2.3 Abschließend

Dieses Kapitel bot einen ersten Einblick in die Ergebnisse und Leistungsfähigkeit des Testbeds. Anhand verschiedener Proof-of-Concept Experimente konnte die erfolgreiche Durchführung der Anwendungsszenarien gezeigt werden.
Zusammen mit den im Anschluss vorgestellten Evaluationsmetriken soll eine tiefergehende Betrachtung und Auswertung im Diskussionskapitel erfolgen.

6 Diskussion

Der folgende Teil soll zunächst die vorgestellten Ergebnisse des Evaluationskapitels in einen Kontext setzen und anhand dessen die Qualitätsziele auswerten und diskutieren. Anschließend werden weitere, während der Entwicklung aufgedeckte Beobachtungen, diskutiert.

6.1 Ergebnisse

Die Ergebnisse des Experiments aus Sektion 5.1.1 zeigen deutliche Unterschiede zu den publizierten Werten. Dieses Verhalten wirft Fragen auf und soll im Folgenden näher betrachtet werden.

Zur Durchführung des Experiments wurde der veröffentlichte und in Abschnitt 3.3.2 analysierte Datensatz der Studie [10] verwendet. Es wird angenommen, dass die veröffentlichten Erkennungswerte ebenfalls auf Basis dieses Datensatzes erzeugt wurden.

Auf Basis dieser Annahme lässt sich die Diskrepanz der Ergebnisse auf die verwendete Implementierung des Erkennungsalgorithmus zurückführen.
So liegt die Vermutung nahe, dass die verwendete Implementierung, mit den vorgestellten Parametern, in dieser Version nicht für die Erzeugung der veröffentlichten Ergebnisse verwendet wurde.

Eine ausführliche Analyse des Codes bestätigt die Zweifel an der Leistungsfähigkeit der veröffentlichten Version. So wurden mehrere Programmierfehler, Ungenauigkeiten und Abweichungen zu dem in [10] beschriebenen Ablauf des Algorithmus identifiziert.
Des Weiteren wurde im Kontakt mit einem Autor der Studie deutlich, dass der veröffentliche Code in dieser Version nicht lauffähig ist. Die in Abbildung 5.1 und 6.1 verwendete Version wurde daher geringfügig angepasst, um die Lauffähigkeit für die Durchführung des Experiments wiederherzustellen.

Da die erzeugten Erkennungswerte solch signifikante Differenzen aufweisen und um der Frage nachzugehen, ob sich die Ergebnisse mithilfe des Testbeds verbessern lassen, wurde zudem eine Eigenimplementierung vorgenommen. Die Ergebnisse dieser Versionen sind in Abbildung 6.1 gegenübergestellt.

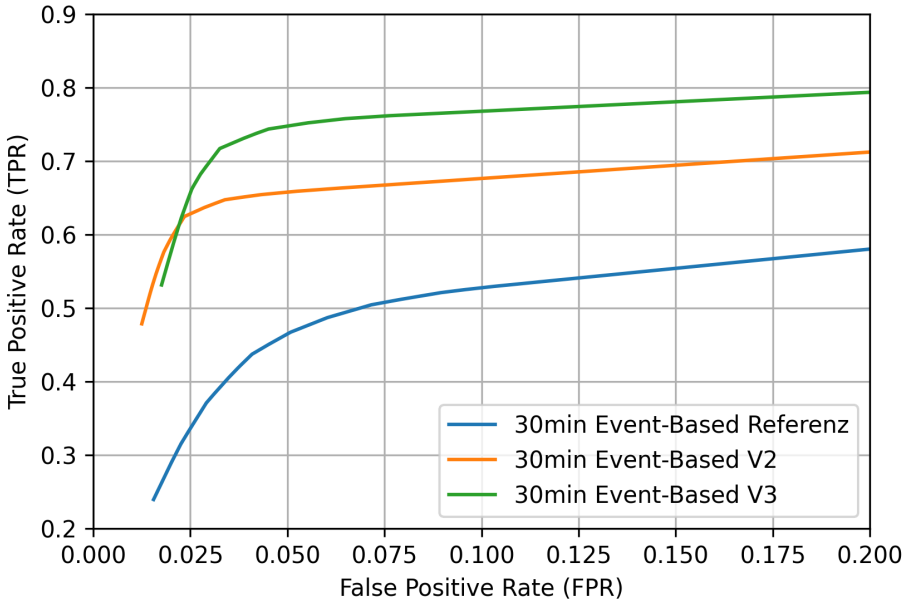


Abbildung 6.1: Vergleich verschiedener Implementationen des Erkennungsalgorithmus

Die Graphen der Versionen 2 und 3 zeigen ein Verhalten, das der veröffentlichten Version ähnlich ist. So steigt die Richtig-Positiv-Rate zunächst steil an, um im weiteren Verlauf, linear mit der Falsch-Positiv-Rate zu steigen. Version 2 und 3 weisen zu Beginn eine stärkere Steigung auf, während die Referenzimplementierung im Verlauf eine geringere Krümmung beschreibt.

Die Werte der beiden Versionen V2 und V3, stellen allerdings insgesamt, eine deutliche Verbesserung gegenüber der Referenzimplementierung dar. So erreicht Version 3 eine um knapp 30 % gesteigerte TPR von 0.46 auf 0.74, bei einer FPR von 0.05.
Implementation V3 verwendet hierfür einen überarbeiteten Extraktionsprozess, während zur Erkennung, die bestehende Erkennungsroutine der Referenzimplementierung genutzt wird.

Trotz der deutlichen Verbesserung der Erkennungsleistung durch Implementation V3 bleibt zu bemerken, dass die erzielten Werte nach wie vor hinter den veröffentlichten Werten zurückbleiben. Dies kann möglicherweise auf den Umstand zurückgeführt werden, dass die entwickelten Versionen weiterhin auf dem veröffentlichten Referenzcode basieren. Für die genauen Gründe der geringen Leistung ist eine weiterführende Analyse des in [10] beschriebenen Algorithmus notwendig.

Die Erkenntnisse, die in diesem Abschnitt gewonnen wurden, legen einerseits deutliches Verbesserungspotenzial in Bezug auf die Implementierung des veröffentlichten Algorithmus nahe. Andererseits belegen sie, dass das entwickelte Testbed nicht nur eine Analyse des Algorithmus ermöglicht, sondern auch als Instrument zur Verbesserung der Implementierung dienen kann.
Die Ergebnisse verdeutlichen, dass das Testbed einen Mehrwert bieten kann, indem es sowohl die Evaluierung der Implementation als auch dessen Weiterentwicklung unterstützt.

6.2 Qualitätsziele

Im nächsten Schritt wird die Leistungsfähigkeit des Testbeds mithilfe der vorgestellten Evaluationsmetriken ermittelt. Hierbei werden die Ergebnisse des Evaluationskapitels genauer betrachtet.

6.2.1 Fidelität

Analyse Datensätze

In diesem Abschnitt wird die Testbed-Fidelität anhand einer Untersuchung des Referenzdatensatzes bewertet. Der Datensatz bildet die Grundlage für die Experimente und ermöglicht den Vergleich der ermittelten Erkennungswerte mit den veröffentlichten Referenzwerten.

```
1141095048 2019-08-02 06:14:28+00:00 UTC+00:00 audio 1943763
1141095048 2019-08-02 06:29:21+00:00 UTC+00:00 photo none
1141095048 2019-08-02 06:29:46+00:00 UTC+00:00 text 16
1141095048 2019-08-02 06:44:23+00:00 UTC+00:00 photo 235975
```

Listing 6.1: Auszug aus dem Veröffentlichten Telegram Datensatz (channel-0.txt)

Eine Analyse mit Fokus auf die Konsistenz des Datensatzes zeigt, dass 17,83 % dieser veröffentlichten Ground-Truth Nachrichten, keine Größenangaben enthalten, und anstelle dessen den Platzhalter 'None' ausweisen (Siehe Listing 6.1).
Dies wirft die Frage auf, ob diese Nachrichten weiterhin vom Client empfangen wurden und ihre tatsächliche Größe lediglich unbekannt ist. Dies würde bedeuten, dass die übertragenen Daten weiterhin auch auf den Netzwerkmitschnitten vorhanden sind.

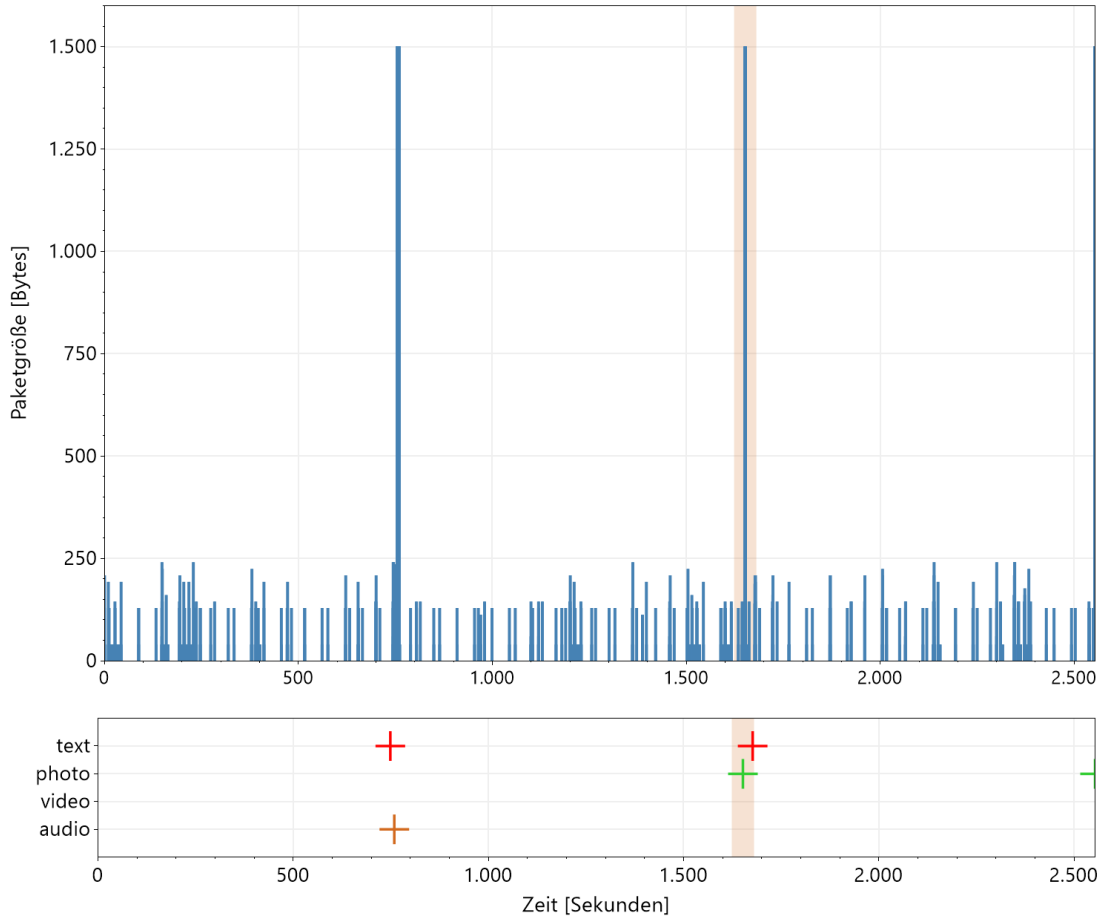


Abbildung 6.2: Detaillierte Analyse des bestehenden Datensatzes

Bei genauer Betrachtung der Netzwerkmitschnitte (Abbildung 6.2) zu den Zeitpunkten der unvollständigen Nachrichten (orange markiert), lässt sich eine deutliche Spitze in der Übertragung beobachten. Die Ansammlung von Paketen in MTU-Größe, die nur sehr kurz nach dem Zeitstempel der versendenden 'None' Nachricht beobachtbar sind, bestärken die Vermutung, dass diese Nachrichten dennoch vom Client empfangen wurden.

Interessanterweise handelt es sich bei 100 % dieser unvollständigen Nachrichten um Fotos. Diese weisen in ihrer durchschnittlichen Größe von 352,4 KB, eine hohe Differenz zu den Protokollnachrichten (typischerweise unter 500 Byte [10]) auf und erzeugen somit ein starkes charakteristisches Muster in den Netzwerkmitschnitten, wie sich auch in Abbildung 6.2 erkennen lässt.
Daher ist es nicht praktikabel, diese Nachrichten einfach zu ignorieren.

Die Diskrepanz dieser unvollständigen Nachrichten kann je nach verwendetem Algorithmus sehr problematisch sein. Falsch gelabelte, oder inkonsistente Daten, erschweren das Training Maschine-Learning basierter Algorithmen, da sie zu inkorrekten Modellannahmen führen können und die Verallgemeinerungsfähigkeit beeinträchtigen [34]. Darüber hinaus können sie auch den Erfolg statistischer-Erkennungsroutinen beeinflussen.

Daher wurde bei der Entwicklung des Testbeds besonderes Augenmerk auf die Datenfidelität während der Aufzeichnung gelegt.
Durch die detaillierte Analyse der Telegram-API und das ausführliche Testen der Empfangskomponente konnten Nachrichten mit unbekannten oder fehlenden Dateigrößen vollständig ausgeschlossen werden, wie sich in Abbildung 6.3 erkennen lässt.

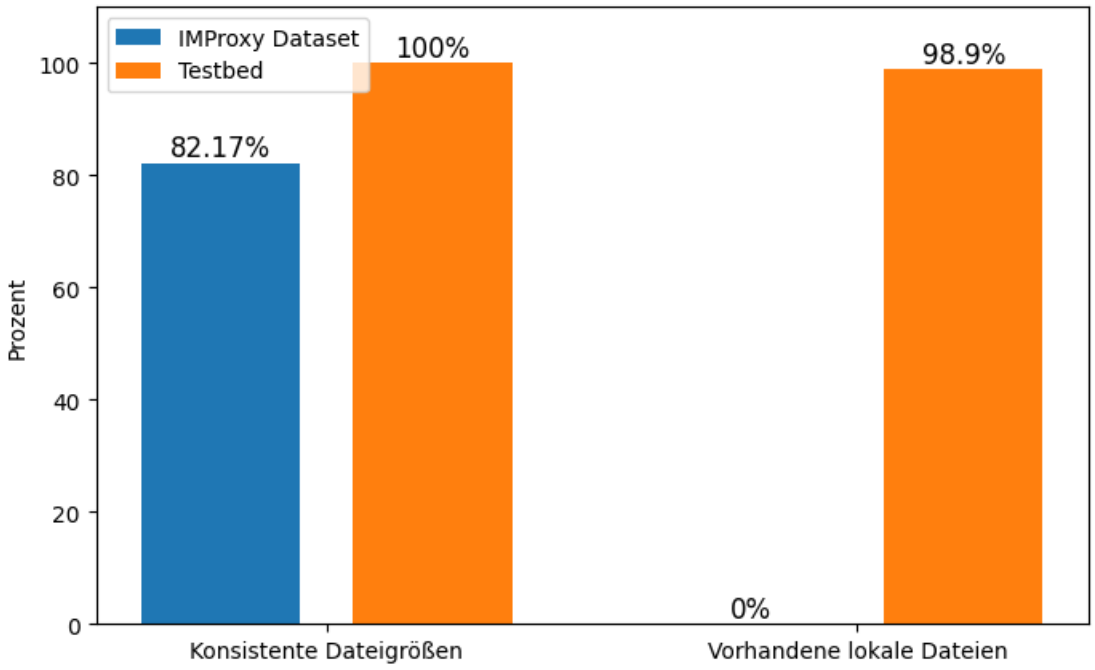


Abbildung 6.3: Vergleich der Datensatz-Fidelität

Jedoch wurde festgestellt, dass trotz korrekter Erfassung der Dateigrößen, im Durchschnitt 1,02 % der empfangenen Dateien nicht erfolgreich heruntergeladen werden konnten.
Dies stellt eine Herausforderung dar, wenn die Nachrichten im Rahmen von Szenario

2 wiedergegeben werden sollen und als lokale Dateien zum erneuten Versand vorliegen müssen.

Eine mögliche Lösung besteht darin, synthetische Dateien zu generieren, welche korrekte Dateigrößen aufweisen. Dieser Ansatz kann zwar zeitaufwendig sein, aber wird als durchaus realisierbar eingeschätzt.
Eine alternative Methode besteht darin, die Nachrichten mit fehlenden lokalen Kopien aus der ursprünglichen Aufzeichnung zu entfernen.
Dies ist in diesem Fall, durch die Trennung der Anwendungsszenarien in Aufzeichnung und Wiedergabe, problemlos möglich. So lässt sich das bereinigte Nachrichtenprotokoll erneut abspielen und weist lediglich eine rechnerische Differenz von 1,02 % zu den original aufgezeichneten Aktivitäten auf. Dagegen bleibt die Konsistenz zwischen den Ground-Truth-Nachrichten und dem Netzwerkverkehr vollständig gewährleistet und ermöglicht eine glaubwürdige und aussagekräftige Analyse.

Um die Ergebnisse angemessen in den Kontext zu stellen, ist es wichtig zu erwähnen, dass der Referenzdatensatz keinerlei heruntergeladene Daten enthält. Dies hat zur Folge, dass die Reproduzierbarkeit der Ergebnisse nur schwer möglich ist.

Zusammenfassend kann festgestellt werden, dass das Testbed einen messbaren Mehrwert bietet, indem es die Datenfidelität durch konsistente Größenangaben verbessert.
Durch die Aufteilung des Angriffs in die Anwendungsszenarien lassen sich zudem Fehler oder Anomalien, die während der Aufzeichnung erfolgten, bereinigen und korrigieren. Dies ermöglicht die Verwendung der erzeugten Daten mit hoher Genauigkeit, trotz der Fehlerrate von 1,02 %

Nachrichtenzusammensetzung

Im Rahmen einer weiteren Analyse wurde die Fidelität des Datensatzes hinsichtlich der Repräsentation der Umgebung untersucht, wobei insbesondere die Zusammensetzung der Nachrichten betrachtet wurde.

So lieferte das Experiment 5.1.2 deutliche Abweichungen zu den veröffentlichten Werten aus [10]. Aus diesen Gründen wurde das Experiment mit anderen Gruppenkonfigurationen wiederholt, um die verschiedenen Ergebnisse in einen Kontext zu setzen.

In Abbildung 6.4 wird daher die Zusammensetzung der Nachrichten aus Aufnahmen unterschiedlicher Gruppen dargestellt.

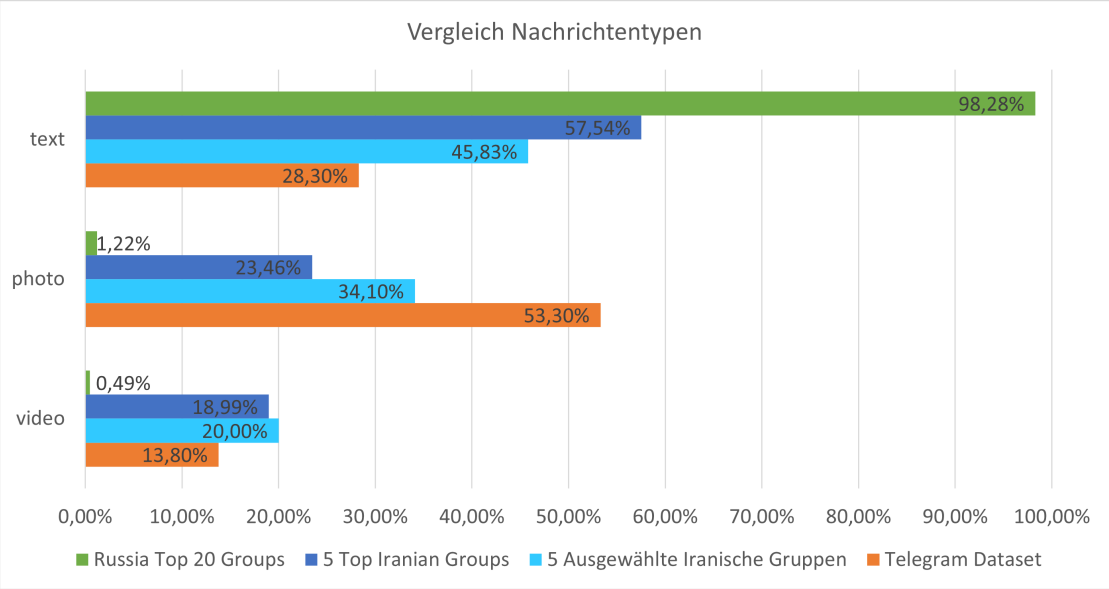


Abbildung 6.4: Vergleich der Nachrichtentypen verschiedener Datensätze

Dabei zeigt sich erstens, dass die Zusammensetzung der Nachrichten stark variieren kann, insbesondere bei Textnachrichten, die zwischen 28 % und 98 % des Gesamtdatensatzes ausmachen können. Zweitens fällt auf, dass der Referenzdatensatz („Telegram-Dataset“) im direkten Vergleich, wenige Textnachrichten, dafür aber verhältnismäßig viele Fotos enthält.

So wird deutlich, dass die Zusammensetzung der Nachrichten stark von den aufgezeichneten Gruppen abhängt. In weiteren Experimenten ließ sich zudem eine Abhängigkeit des Nachrichtenaufkommens zu den Tageszeitpunkten identifizieren.

Die Nachrichtenzusammensetzung kann, wie bereits erwähnt, Auswirkungen auf den Erfolg der Erkennungsroutinen haben, da unterschiedliche Arten von Nachrichten auch unterschiedliche Größen aufweisen (Siehe 5.1). Große Nachrichten, wie Videos oder Fotos, erzeugen unweigerlich sehr charakteristische Netzwerkmuster, wohingegen Textnachrichten in ihrer Größe schwer von Protokollnachrichten zu unterscheiden sind.
Im Referenzcode werden Textnachrichten herausgefiltert, möglicherweise durch die Annahme, dass diese mit durchschnittlich 309 Byte zu klein sind, um erkennbare Verkehrsmuster zu erzeugen. Erwähnt wird diese Filterung in der entsprechenden Veröffentlichung [10] nicht.

So lässt sich anhand dieser Analyse beobachten, dass der Referenzdatensatz, auf welchem die veröffentlichten Ergebnisse basieren, für die Erkennung eine verhältnismäßig gute Nachrichtenzusammensetzung enthält.

Dies verdeutlicht, dass Fidelität auch bedeutet, den Rahmen zu berücksichtigen, in welchem die Simulationsstimuli aufgezeichnet wurden.
Sollen die Ergebnisse vergleichbar und validierbar sein ist es wichtig, dass die Umgebungsbedingungen ebenfalls erfasst und gespeichert werden. In diesem Fall bedeutet es, dass die Gruppen, aus welchen aufgezeichnet wurde, zusammen mit den (Tages)-Zeitpunkte erfasst und protokolliert werden.

```
"RecordedGroups": [
    "Xshab_sinama",
    "SEPAHCYBERY",
    "Bazneshast_Tehran",
    "dokhtarone_profail",
    "qadrat_a1",
    "ZZ_SO"],
  "Start_TimeStamp": 1687049221.5910001,
  "Stop_TimeStamp": 1687063621.5929999,
  "Path": "messageTraces-2023-06-18-16-30-44"
```

Listing 6.2: Protokollierte Umgebungdbedingungen - Auszug aus Nachrichtenprotokoll

Das Testbed liefert beides. In Listing 6.2 lässt sich erkennen, dass die Anzahl und die genauen Gruppen, sowie der Zeitpunkt vom Start und Stop der Aufzeichnung maschinenverarbeitbar gespeichert werden. Dies ermöglicht eine unabhängige Einschätzung und Verifikation der aufgezeichneten Daten und unterstützt so die Reproduzierbarkeit.

6.2.2 Reproduzierbarkeit

Zur Untersuchung der Reproduzierbarkeit von Experimentdaten werden im Folgenden, die Ergebnisse und ihre Variationen bei mehrfacher Durchführung der Anwendungsszenarien betrachtet.

Anwendungsszenario 1

Die mehrfache Durchführung eines Erkennungsalgorithmus auf demselben Datensatz, führt zu den exakt gleichen Ergebnisse.
Variationen ergeben sich nur durch die Verwendung anderer Datensätze, Parameter oder Erkennungsroutinen.

Dies Entspricht den Erwartungen und deckt sich mit der Annahme, dass die eingesetzten Erkennungsalgorithmen ein deterministisches Verhalten aufweisen.

Anwendungsszenario 2

Im folgenden Abschnitt wird untersucht, ob die erfassten Endbenutzeraktivitäten bei wiederholter Durchführung der Aufzeichnung, reproduzierbar sind. Hierfür wurde das Experiment aus Abschnitt 5.1.2 fünfmal wiederholt und die Variationen des Nachrichtenaufkommens analysiert.

Wie sich in Abbildung 6.5 erkennen lässt, erzeugt die wiederholte Durchführung der Aufzeichnung aus derselben Gruppe eine variierende Anzahl Nachrichten. Die größten Schwankungen weisen Textnachrichten auf. Foto und Video Nachrichten scheinen dagegen eine geringere Variation aufzuweisen.

Trotz der deutlichen Schwankungen in der absoluten Anzahl der Nachrichten, bleibt das statistische Verhältnis der Nachrichtentypen zueinander, über die mehrfache Durchführung hinweg, bestehen.

Die beobachteten Schwankungen, bei mehrfachen Aufzeichnungen, entsprechen den Erwartungen, da die Aufzeichnung realer Gruppenaktivitäten unkontrollierbare Faktoren beinhalten kann, die zu nicht vorhersehbaren Ergebnissen führen. Ein extremes Beispiel hierfür sind reale Ereignisse wie politische Spannungen oder Naturkatastrophen, die zu plötzlichen Veränderungen im Nachrichtenaufkommen führen können und sich nicht reproduzierbar abbilden lassen. Solche unvorhersehbaren Faktoren können die Ergebnisse beeinflussen und müssen bei der Interpretation der Daten unbedingt berücksichtigt werden.

Die Tatsache, dass sich das Verhältnis der Nachrichtentypen trotz der Schwankungen in der absoluten Anzahl nicht signifikant verändert, deutet auf eine Korrelation zwischen den Gruppen und der Zusammensetzung der Nachrichten hin. Diese Beobachtung stimmt mit

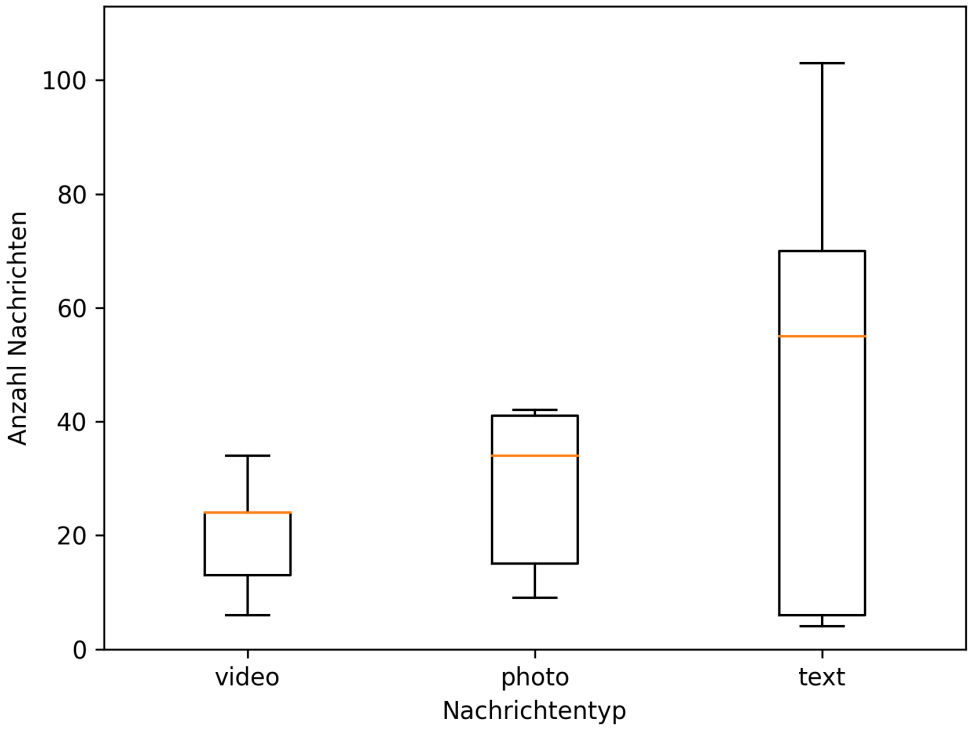


Abbildung 6.5: Anzahl Nachrichten bei wiederholter Aufzeichnung derselben Gruppe

den bereits in Abschnitt 6.2.1 diskutierten Ergebnissen überein. Dort wurde festgestellt, dass sich die Zusammensetzung der Nachrichten erheblich zwischen den verschiedenen Gruppen unterscheidet.

Um trotz der Schwankungen, reproduzierbare Experimente zu ermöglichen, wurde eine Trennung zwischen der Erfassung der Endbenutzer*innenaktivitäten als Simulationsstimuli und der Durchführung des simulierten Angriffs vorgenommen.
Diese Trennung ermöglicht es, den Angriff mit den gleichen Aktivitäten zu wiederholen und somit eine konsistente Grundlage für die Experimente zu schaffen.

Anwendungsszenario 3

Um die Reproduzierbarkeit des dritten Anwendungsszenarios zu untersuchen, wurde auch hier ein Experiment mehrfach durchgeführt.

Als Quelldatei dient ein 70-minütiger Datensatz mit insgesamt 57 Nachrichten. Der Experimentaufbau wurde gemäß Abschnitt 5.1.3 festgelegt und während der Durchführung nicht verändert.
Das Experiment wurde zehnmal durchgeführt, wobei für jede Durchführung die Korrelationswerte aus den erzeugten Daten ermittelt wurden. Um die aufgetretenen Variationen zu untersuchen, stellt Abbildung 6.6 die Streuung und den Median für die jeweiligen Beobachtungsintervalle als Boxplot dar.

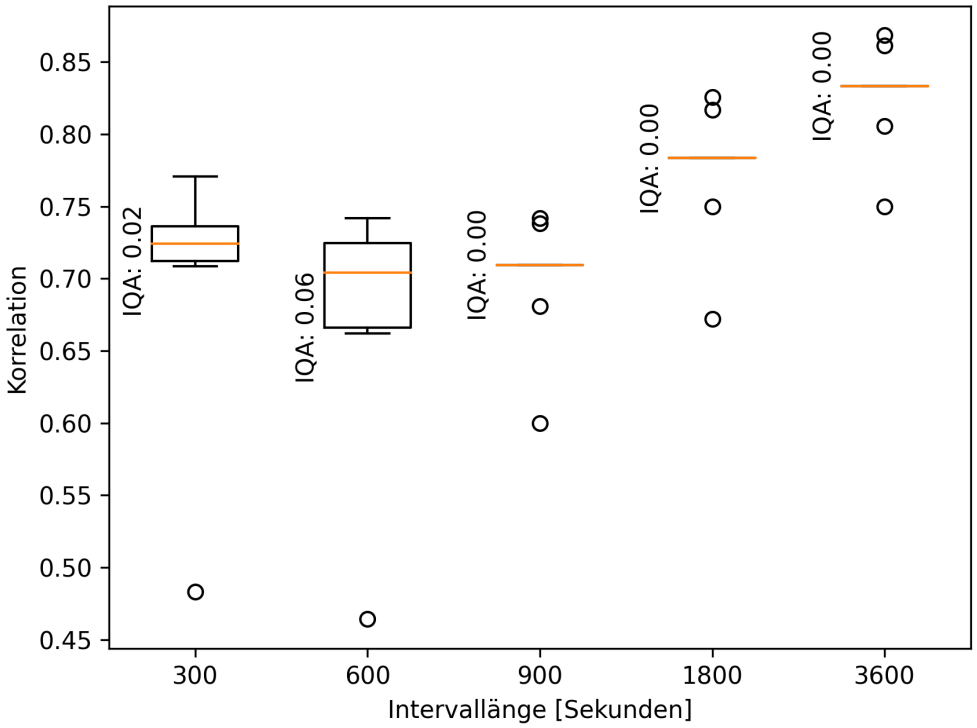


Abbildung 6.6: Verteilung der Korrelationswerte für verschiedene Intervallängen

In der Abbildung ist zu erkennen, dass die Korrelationswerte abhängig von der Beobachtungsdauer geringfügige Schwankungen aufweisen. Bei 300 und 600 Sekunden lässt sich mit Ausnahme eines Ausreißers eine sehr kompakte Verteilung feststellen. Bei längeren Beobachtungsintervallen liegen die mittleren 50 % der Werte so eng beieinander, dass die Interquartilabstände (IQA), die auch als robustes Streuungsmaß verwendet werden können, 0 betragen. Dadurch wird die Box des Boxplots nicht mehr sichtbar. Die Ausrei-

ßer sind bei diesen Beobachtungswerten etwas näher beieinander, weisen jedoch zwischen Maximum und Minimum einen Abstand von 0,15 Punkten auf.

Die Schwankungen in der Trefferrate entsprechen weitestgehend den Erwartungen, da sie durch verschiedene Faktoren ausgelöst werden können. Ein möglicher Einflussfaktor ist die Auslastung der Telegram-Server. Wenn viele Anfragen gleichzeitig auftreten, kann dies zu einer verzögerten Abarbeitung führen, was sich wiederum auf die Latenz der Nachrichtenübertragung auswirken kann. Darüber hinaus können störanfällige oder ausgelastete Netzwerkverbindungen den Datendurchsatz beeinträchtigen und somit zu abweichenden Übertragungsmustern führen [10].

Nicht auszuschließen sind selbstverständlich auch Schwankungen, die durch den Testaufbau selbst verursacht werden.

Um dieses Verhalten genauer zu untersuchen, lohnt sich eine weitere Betrachtung. Dabei konnte unter Beobachtung der empfangenen Datenmenge ein interessantes Verhalten festgestellt werden.

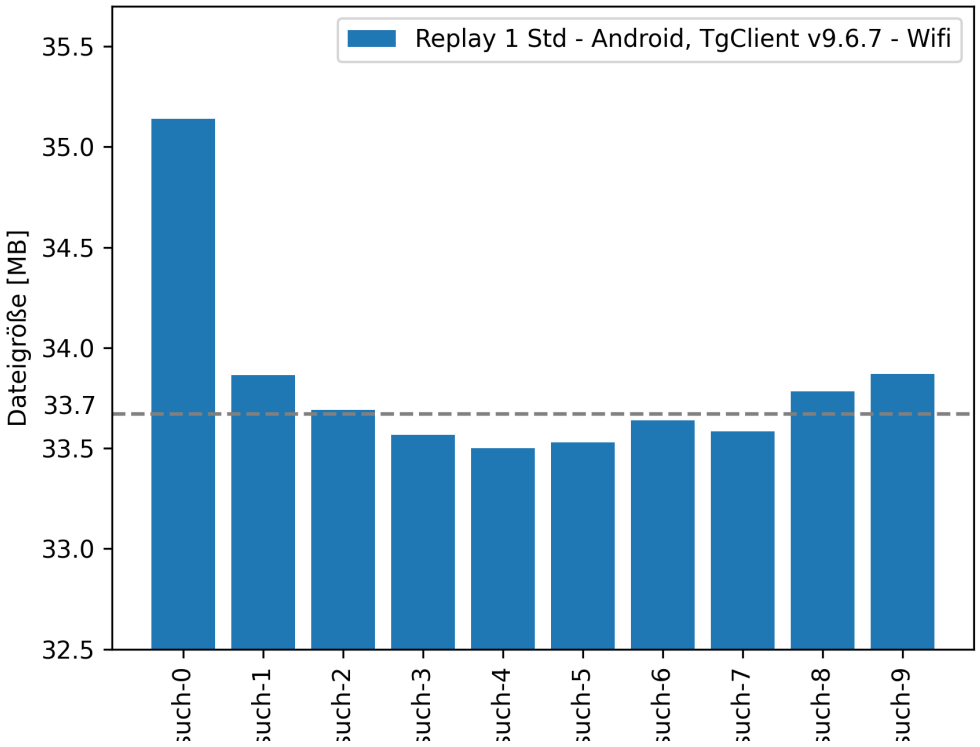


Abbildung 6.7: Analyse der übertragenen Daten bei wiederholter Durchführung

In Abbildung 6.7 lässt sich erkennen, dass während der ersten Durchführung des Versuchs, im Mittel 1,4 MB mehr Daten durch den Client empfangen wurden. In den darauf folgenden Versuchen blieb die Datenmenge mit einer mittleren Abweichung vom Mittelwert von 0,34 % verhältnismäßig konstant. Dieses Verhalten ließ sich auch unter Verwendung des Desktop-Clients reproduzieren.

Da es sich bei den Übertragungen jedes Mal um die gleichen Nachrichten handelt, ist es denkbar, dass zur Verringerung des Netzwerkverkehrs und Verbesserung der Performance, ein Clientseitiger Caching-Mechanismus eingesetzt wird.

Dieser könnte durch die Identifizierung bereits heruntergeladener Nachrichten funktionieren, indem erneut empfangene Nachrichten beim Download übersprungen werden.

So ließ sich während der Entwicklung des Testbed Telegram-Client Moduls, ein ähnliches Verhalten beobachten: Die Nachrichten-IDs von Fotos bleiben über eine Sitzung hinweg

bestehen. Wird ein weiteres Bild, mit derselben ID empfangen, lässt sich schnell, ohne erneuten Download, auf die lokale Kopie verweisen.

Dieses Verhalten verdeutlicht, dass zur Beurteilung der Reproduzierbarkeit, verschiedene Einflussfaktoren untersucht werden müssen.
Weitere Versuche sind erforderlich, um die Auswirkungen der Serverlast, Netzwerkeinflüsse und implementierungsspezifischen Variationen der Clients genauer zu ermitteln und präzisere Aussagen über die Verlässlichkeit und Reproduzierbarkeit des Testbeds treffen zu können. Die vorliegenden Daten ermöglichen bisher eine Abschätzung der unteren Grenze der Testbed-Leistung.

Dieser Versuch unterstreicht erneut die Fähigkeit des Testbeds, derartige Verhaltensweisen aufzuzeigen und mögliche Zusammenhänge zwischen den variablen Faktoren identifizierbar zu machen.

6.2.3 Automation

Um den Setup-Prozess zu vereinfachen, wurde dieser automatisiert. Das Testbed stellt hierfür einen Testplan bereit, der ein „Sniffing Instrument“ und den nötigen Testschritt beinhaltet. Das Instrument übernimmt die Verwaltung der Verbindung, sodass der Setup-Schritt die Installation der erforderlichen Software, die Konfiguration des Gateways und des Testnetzwerks (einschließlich eines Access Points und zusätzlicher Ethernet-Adapter) automatisch durchführen kann.

Die manuelle Konfiguration des Testbeds ist zeitaufwendig und erfordert spezifische Anwendungskompetenz. Es müssen Konfigurationsdateien erstellt und angepasst, sowie mehrere Services gestoppt und gestartet werden. Dies birgt zusätzliches Fehlerpotenzial. Durch die Automatisierung des Setups wurden die Nutzer*inneninteraktionen auf folgende Schritte reduziert:

1. Eintragen der IP-Adresse und Authentifizierungsdaten des Raspberrys in die Testbed Management Software.
2. Ausführen des Setup-Testplans.

Im Vergleich zu den 26 Schritten, die bei einer manuellen Konfiguration erforderlich sind, stellt dies eine Reduzierung um den Faktor 13 dar.
Dies ermöglicht einer der Kernkomponenten im Testbed, ein schnelles und wiederholbares

Setup. Durch die Automatisierung wird die Effizienz gesteigert und mögliche Fehlerquellen minimiert.

6.3 Einfluss der Endgerätekonfiguration

Während der Entwicklung ist ein bemerkenswertes Verhalten aufgefallen, das den Einfluss des Zustands der App auf der Empfangsseite betrifft. In sämtlichen bisherigen Experimenten wurde die Nachrichten-App geöffnet, das Endgerät entsperrt und der Bildschirm aktiviert gehalten.
Es wurde festgestellt, dass sich das Verhalten signifikant verändert, wenn das Display des Endgeräts gesperrt bleibt und die App im Hintergrund ausgeführt wird.

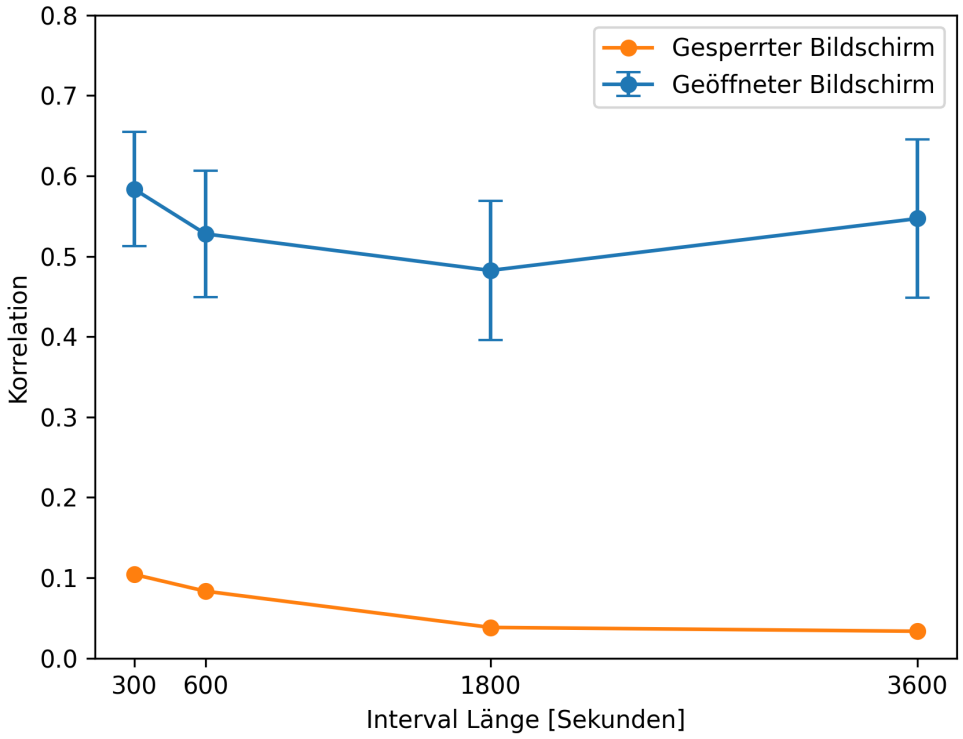


Abbildung 6.8: Korrelation eines gesperrten und entsperrten Smartphones

Eine genaue Untersuchung dieses Szenarios zeigt interessante Ergebnisse. So stellt Abbildung 6.8 die Korrelationswerte eines gesperrten Geräts (Orange), mit den Werten der bisherigen Versuche gegenüber (Blau). Das gesperrte Gerät erzeugt dabei Netzwerkverkehr dessen Korrelationswerte, in Bereich von unkorreliertem Traffic liegt.

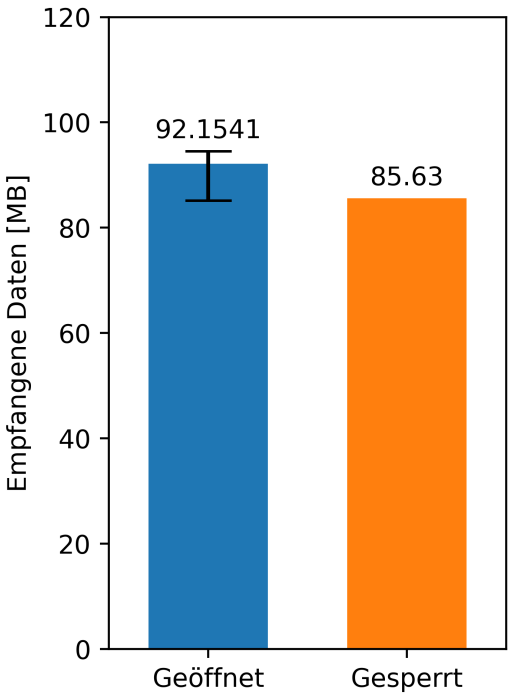


Abbildung 6.9: Empfangene Daten eines gesperrten und entsperrten Smartphones

Betrachtet man hingegen in Abbildung 6.9, die Menge der empfangenen Daten, so lassen sich nur geringfügige Unterschiede zur durchschnittlichen Dateigröße feststellen. Diese liegen noch im Bereich des beobachteten Fehlers.

Dieses Verhalten legt nahe, dass die Dateien zwar übertragen wurden, jedoch nicht in einer Weise, die durch den verwendeten Algorithmus als korrelierbare Verkehrsmuster erkannt werden konnten.

Um dieses Phänomen genauer zu untersuchen, bietet Abbildung 6.10 eine detaillierte Ansicht der Datenübertragung. Zur Demonstration wurde während der Hälfte der Aufzeichnung das Endgerät gesperrt. Auf der linken Seite der Abbildung sind daher die Übertragungsmuster bei entsperrtem Endgerät zu sehen, während ab der zweiten Hälfte,

die Muster bei gesperrtem Bildschirm zu erkennen sind. Unten in der Abbildung sind die gesendeten Nachrichten aufgeführt.

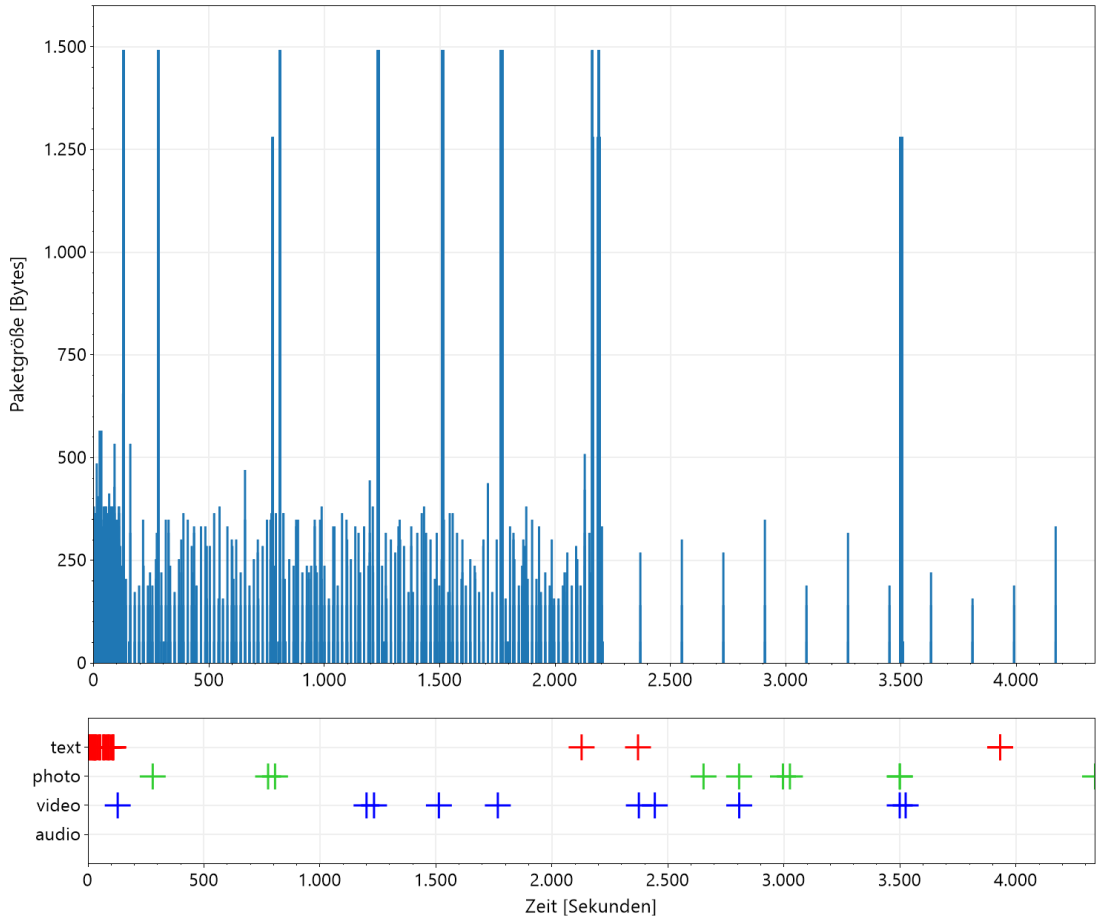


Abbildung 6.10: Empfangene Pakete eines offenen (links) und gesperrten Geräts (rechts)

In der ersten Hälfte der Aufzeichnung lässt sich das erwartete Verhalten erkennen. Videos oder Fotos, erzeugen eine reihe an Paketen in Größe der MTU, die in zeitlicher Korrelation zu den versendeten Nachrichten stehen.

Ab der zweiten Hälfte zeigt sich hingegen ein völlig anderes Bild. Der Client scheint ca. alle drei Minuten eine kurze Verbindung zum Server aufzubauen und Pakete in Protokollgröße auszutauschen. Bis auf einen Ausschlag bei 3.500s lassen sich keine Zusammenhänge zwischen den versendeten Nachrichten und den Verkehrsmustern erkennen.

Dieses Verhalten unterbindet jegliche zeitliche Korrelation. Somit ist es für den eingesetzten Algorithmus nicht möglich zu detektieren, ob zwischen dem Netzwerkverkehr und den versendeten Nachrichten ein Zusammenhang besteht.

Dieser Befund ist insbesondere für die passive Art des Angriffs relevant.
Bei diesem ist das Opfer ein stilles Mitglied einer Gruppe und empfängt lediglich die Nachrichten, die durch die Angreifenden beobachtet oder versendet werden. In diesen Fällen scheint das Sperren des Bildschirms eine der effektivsten Methoden zur Gegenmaßnahme zu sein.

Doch auch für die aktive Art des Angriffs bleibt fraglich wie gut sich die veröffentlichten Erkennungswerte in einem Realitätsnahen Anwendungsszenario reproduzieren lassen.

6.3.1 Zusammenfassend

Die vorliegenden Ergebnisse unterstreichen die Notwendigkeit weiterer Forschung, um das tatsächliche Risiko und die Anwendbarkeit des Angriffs in einem realitätsnahen Szenario zu untersuchen. Insbesondere die Auswirkungen von Umgebungsfaktoren wie der Konfiguration der Endgeräte erfordern eine eingehende Analyse.
Dabei hat sich das Testbed als geeignetes Instrument erwiesen, um das Verhalten und die Auswirkungen solch realitätsnaher Umgebungsfaktoren zu erkennen und zu analysieren.

6.4 Diskussionsfazit

Zusammenfassend lässt sich schließen, dass das Testbed eine breite Palette an Möglichkeiten bietet, um verschiedene Eigenschaften und Verhaltensweisen aufzudecken.
So ließ sich ein Verhalten zum Caching von Nachrichten beobachten, Unterschiede in der verwendeten Anwendungsversion (Desktop vs. Mobile Version) aufdecken, Inkonsistenzen im Referenzdatensatz erkennen und die Abhängigkeit eines entsperrten Displays für eine erfolgreiche Durchführung des Angriffs identifizieren. Diese Ergebnisse zeigen, dass mithilfe des Testbeds eine realitätsnahe und präzisen Nachbildung der Umgebungseigenschaften möglich ist.

Durch das Design des Testbeds, insbesondere durch die Aufteilung der Anwendungsszenarien, konnte eine reproduzierbare Durchführung gewährleistet werden. Eine weiterführende Analyse ist erforderlich, um die Ursachen für die beobachteten Schwankungen weiter zu untersuchen und eine Möglichkeit zur Umgehung des Caching-Mechanismus zu entwickeln.

Zusammenfassend lässt sich sagen, dass das Testbed einen messbaren Mehrwert bietet. Durch weitere Analysen und Anpassungen besteht das Potenzial, die Reproduzierbarkeit zu erhöhen und das Testbed noch effektiver zu gestalten. Insgesamt stellt das Testbed eine wertvolle Ressource für die Erforschung von Flow-Korrelationsangriffen dar.

7 Fazit

Dieses abschließende Kapitel fasst zunächst die zentralen Aspekte der Arbeit zusammen und bietet anschließend eine zusammenfassende Bewertung. Im letzten Abschnitt wird zudem ein Ausblick auf weitere offene Forschungs- und Entwicklungsthemen gegeben.

7.1 Zusammenfassung

Ziel dieser Arbeit war es, eine Test- und Evaluations-Umgebung zu entwickeln, welche die Anwendbarkeit und Rahmenbedingungen von Flow-Korrelationsangriffen auf verschlüsselte Messenger ermöglicht. Das Testbed sollte eine möglichst realitätsnahe Umgebung bieten, in der verschiedene physische Geräte eingebunden und unterschiedliche Netzwerk- und Endgerätekonfigurationen getestet werden können. Es wurde angestrebt, bestehende Datensätze und Erkennungsalgorithmen zu integrieren, um auf vorhandene Ressourcen aufzubauen.

Das Testbed wurde mit einem klaren Fokus auf Fidelität und Reproduzierbarkeit entwickelt, um verlässliche und glaubwürdige Ergebnisse für weiterführende Forschung und Validierung zu gewährleisten.

Im Designprozess wurde das Angriffsszenario in drei Anwendungsszenarien aufgeteilt, um einzelne Aspekte isoliert und unabhängig voneinander zu testen. Basierend auf dem Stand der Forschung und der Referenzliteratur, wurde die Testmanagement-Software OpenTAP ausgewählt und ein geeignetes Hardware-Setup entworfen.

Während der Implementierung wurde der Setup-Prozess der Mitschneidekomponenten entwickelt, um eine parallele und flexible Einbindung verschiedener Geräte über Wi-Fi und Ethernet zu ermöglichen. Durch eine Reihe von Unit-Tests wurde die korrekte Implementierung einiger Kernmodule, wie des *TelegramConnectors*, sichergestellt.

Im nächsten Schritt erfolgte die Evaluation der Anwendungsszenarien anhand von Proof-of-Concept Experimenten. Dabei wurde gezeigt, dass die Erkennungswerte des verwendeten Referenz-Erkennungsalgorithmus deutlich geringer ausfallen, als die in der Referenzpublikation veröffentlichten Werte. Mithilfe des Testbeds konnte daraufhin eine verbesserte Version entwickelt werden, die eine 30 % höhere Erkennungsrate erreichte.

Durch die prototypische Durchführung der weiteren Anwendungsszenarien konnten neue Datensätze generiert werden, die eine Kontextualisierung der Referenzergebnisse und des Datensatzes ermöglichten. Dadurch konnten einige Abhängigkeiten der Ergebnisse zu den Umgebungsbedingungen beobachtet werden, wie zum Beispiel die Auswirkungen der Messenger Gruppen auf die Zusammensetzung der Nachrichtentypen. Diese Variationen konnten mithilfe des Testbeds weiter untersucht werden und verdeutlichten die Bedeutung einer detaillierten Protokollierung der Umgebungsbedingungen für die Reproduzierbarkeit und Fidelität.

Mit Blick auf die Fidelität wurde das Testbed anhand weiterer Experimente untersucht. Es zeigte sich, dass das Testbed eine Verbesserung der Datensatz-Fidelität erreicht, da bei der Aufzeichnung von Simulationsdaten Nachrichten mit fehlenden Dateigrößen vollständig ausgeschlossen werden konnten.

Die Reproduzierbarkeit konnte anhand verschiedener Teilexperimente bestätigt werden, wobei ein Verbesserungspotenzial identifiziert wurde, hinsichtlich der Variationen beim Abspielen von Nachrichten. Dieser Aspekt sollte in zukünftigen Forschungsarbeiten weiter untersucht werden, um die Reproduzierbarkeit des Testbeds weiter zu verbessern.

Weitere Erkenntnisse betreffen die Auswirkungen des Endgeräts auf den Erfolg des Angriffs. Es wurde festgestellt, dass ein gesperrter Bildschirm die messbare Korrelation nahezu vollständig unterbricht und den Angriff somit unwirksam macht.
Auch der Einsatz unterschiedlicher Hardwarekomponenten zeigte interessantes Verhalten. Es wurde erkannt, dass der Desktop-Client weniger Daten überträgt als die Android-Anwendung.

7.2 Bewertung

Im Rahmen der abschließenden Bewertung können unterschiedliche Aspekte in Betracht gezogen werden.

Die in der Analyse ermittelten Anforderungen und Qualitätsziele erwiesen sich als geeignete Leitlinien für den Entwicklungsprozess. So lieferte die Reproduzierbarkeit und Fidelität, sowie das Streben nach der Verwendung quelloffener Technologien, gute Kriterien, um geeignete Hard- und Software Lösungen auszuwählen.
Hilfreich waren dabei die verwandten Arbeiten, die bereits eine Verwendung der entsprechenden Technologien und Methoden beschreiben. Unter Beachtung der dort veröffentlichten Erkenntnisse und Methoden, konnte eine risikoreduzierte Entwicklung auf dem aktuellen Forschungsstand durchgeführt werden.

Auch wenn die Implementierung der entworfenen Komponenten größtenteils reibungslos verlief, ließ sich innerhalb des Prozesses Verbesserungspotenzial identifizieren. Insbesondere betraf dies die Entwicklung von Komponenten und Funktionalitäten, die keine direkte Auswirkung auf die angestrebten Qualitätsziele hatten. Obwohl sie sich als hilfreich während der Entwicklungsphase erwiesen, spielten sie eine eher untergeordnete Rolle bei der Erreichung der geforderten Ziele.

Insgesamt liefern die Ergebnisse aber ein klares Bild und unterstreichen die Bedeutung des Testbeds für die Durchführung realitätsnaher Experimente.
Durch den Einsatz physischer Geräte konnten spezifische Verhaltensweisen identifiziert werden, die in einem rein virtuellen Umfeld nicht aufgetreten wären. Dies verdeutlicht die Notwendigkeit realitätsnaher Testszenarien, um aussagekräftige Ergebnisse zu erzielen.

Des Weiteren ermöglichte das Testbed eine kritische Betrachtung der von [10] veröffentlichten Datensätze und Ergebnisse und offenbart damit weiteren Forschungsbedarf.
Diese Erkenntnis belegt, dass sowohl die Validierung der Datensätze, als auch die Verbesserung der Erkennungsalgorithmen mithilfe des Testbeds realisierbar sind. Somit wird deutlich, dass das Testbed einen deutlichen Mehrwert für die Evaluierung bestehender Forschungsergebnisse bieten kann.

Die variierenden Ergebnisse bei Verwendung verschiedener Endgeräte oder Messengergruppen, demonstriert die Fähigkeit des Testbeds, solche Umgebungsfaktoren realitätsgetreu abzubilden. Dies spricht für eine hohe Fidelität des Testbeds und zeigt, dass es sich dafür eignet die Auswirkungen von Umgebungsdetails auf die Anwendbarkeit der Angriffe zu untersuchen.

Das Testbed erweist sich als äußerst wertvolles Instrument für die Erforschung von Flow-Korrelationsangriffen auf verschlüsselte Messenger. Seine Bedeutung und Relevanz für

dieses Forschungsgebiet sind klar ersichtlich und die gewonnenen Erkenntnisse unterstreichen die Notwendigkeit weiterer Forschung in diesem Bereich.

Zukünftige Forschung sollte sich auf die Untersuchung weiterer Einflussfaktoren konzentrieren, wie z.B. die Auswirkungen von Netzwerkeinflüssen und implementierungsspezifischen Variationen der Clients. Zusätzlich sollten mögliche Zusammenhänge bei wiederholter Verwendung gleicher Experimentdaten, genauer untersucht werden, um präzisere Aussagen über die Verlässlichkeit und Reproduzierbarkeit des Testbeds zu ermöglichen.

Mit dieser Forschungsarbeit wird die Hoffnung verbunden, dass solche Angriffe weiterführend untersucht werden - nicht nur, um ein größeres Bewusstsein für derartige Bedrohungen zu schaffen, sondern auch, um die Forschung im Bereich der Gegenmaßnahmen voranzutreiben.

7.3 Ausblick

Während der Entwicklung des Testbeds wurden verschiedene Möglichkeiten zur Verbesserung und Erweiterung identifiziert:

Derzeit erfordert der Einsatz von Endgeräten eine manuelle Konfiguration, um sie beispielsweise mit dem Test-WLAN zu verbinden. Eine vielversprechende Idee besteht darin, die Geräte über eine Programmier-Schnittstelle in das Testbed zu integrieren. In einer Veröffentlichung des *TRIANGLE-Projekts* [37] wird die Integration der Geräte mithilfe von Automatisierungssoftware wie *Quamotion* beschrieben. Dies erlaubt das Konfigurieren und Ausführen beliebiger Anwendungen und ermöglicht es Touch-Eingaben zu simulieren.
Obwohl dieser Schritt bereits frühzeitig in Betracht gezogen wurde, wurde aus Gründen des Umfangs in der Entwicklung darauf verzichtet.
Die automatisierte Geräteintegration hätte mehrere Vorteile. Zum einen würde sie ermöglichen, einen aktiven Teilnehmer zu simulieren, der Nachrichten in Gruppen verschickt. Dies entspricht dem Szenario, in dem ein Admin einer Gruppe, Ziel des Angriffs ist. Durch die bisherige Konfiguration konnte dies nicht untersucht werden.
Zum anderen könnten die Endgeräte automatisiert konfiguriert werden, was automatisierte Tests und Experimente mit geöffneter, oder im Hintergrund aktiver App ermöglicht. Dadurch ließen sich noch realitätsnähere Aktivitäten simulieren, wie das Schließen und

Öffnen verschiedener Chats oder Apps, und das tatsächliche Eintippen von Nachrichten.

Eine weitere Idee besteht darin, Komponenten hinzuzufügen, die eine bessere Kontrollierbarkeit der Umgebungsbedingungen ermöglichen. Beispielsweise könnte die Simulation von Netzwerkbedingungen für die Untersuchung der Reproduzierbarkeit vorteilhaft sein. Ein möglicher Ansatz wäre der Einsatz eines Simulators wie *OPNET* [15], der verschiedene Netzwerkumgebungen simulieren kann. Dadurch könnten auch die Auswirkungen von Mobilnetzen wie 5G oder LTE untersucht werden.

Auch wurde in diesem Testbed bisher keine Untersuchungen zur Verwendung von Gegenmaßnahmen durchgeführt. Es wäre interessant zu untersuchen, wie sich die Maßnahmen aus [10] und [50] in realitätsnahen Szenarien verhalten und welchen Einfluss sie auf die Ergebnisse haben.

Literaturverzeichnis

[1]

[2] AbdelHafeez, Mahmoud ; Ahmed, Ali H. ; AbdelRaheem, Mohamed: Design and Operation of a Lightweight Educational Testbed for Internet-of-Things Applications. In: *IEEE Internet of Things Journal* 7 (2020), Nr. 12, S. 11446–11459

[3] Ani, Uchenna ; Craggs, Barnaby ; Watson, Jeremy ; Green, Benjamin ; Nurse, Jason: Design Considerations for Building Credible Security Testbeds: Perspectives from Industrial Control System Use Cases. In: *Journal of Cyber Security Technology* 4 (2020), nov, Nr. 3, S. 1–49. – ISSN 2374-2925

[4] Arpaia, Pasquale: *Flexible test automation: A software framework for easily developing measurement applications*. Momentum Press, 2015. – ISBN 978-1-60650-383-6

[5] Arpaia, Pasquale ; De Matteis, Ernesto ; Inglese, Vitaliano: Software for measurement automation: A review of the state of the art. In: *Measurement* 66 (2015), 02

[6] Astronics: *ActivATE Test Management Software*. – URL https://www.astronics.com/searchResults?q=ActivATE&offset=0. – Zugriffsdatum: 01.06.2023

[7] Atkinson, J S. ; Adetoye, O ; Rio, M ; Mitchell, J E. ; Matich, G: Your WiFi is leaking: Inferring user behaviour, encryption irrelevant. In: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, S. 1097–1102

[8] Atkinson, J.S. ; Rio, M. ; Mitchell, J.E. ; Matich, G.: Your WiFi Is Leaking: Ignoring Encryption, Using Histograms to Remotely Detect Skype Traffic. In: *2014 IEEE Military Communications Conference*, 2014, S. 40–45

[9] Back, Adam ; Möller, Ulf ; Stiglic, Anton: Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In: Moskowitz, Ira S. (Hrsg.): *Information*

Hiding Bd. 2137, Springer Berlin Heidelberg, 04 2001, S. 245–257. – ISBN 978-3-540-45496-0

[10] Bahramali, Alireza ; Houmansadr, Amir ; Soltani, Ramin ; Goeckel, Dennis ; Towsley, Don: Practical Traffic Analysis Attacks on Secure Messaging Applications, 01 2020

[11] Bozorgi, Ardavan ; Bahramali, Alireza ; Rezaei, Fateme ; Ghafari, Amirhossein ; Houmansadr, Amir ; Soltani, Ramin ; Goeckel, Dennis ; Towsley, Don: I Still Know What You Did Last Summer: Inferring Sensitive User Activities on Messaging Applications Through Traffic Analysis. In: *IEEE Transactions on Dependable and Secure Computing* (2022), S. 1–18

[12] BSI: *Technische Richtlinie TR-02102-2 Kryptographische Verfahren: Empfehlungen und Schlüssellängen.* jan 2023. – URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=6. – Zugriffsdatum: 2023-5-20

[13] C., Robert M.: *Clean architecture: A craftsman's guide to software structure and design.* Pearson, 2017. – ISBN 978-0-13-449416-6

[14] Chaum, David L.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. In: *Commun. ACM* 24 (1981), feb, Nr. 2, S. 84–90. – URL https://doi.org/10.1145/358549.358563. – ISSN 0001-0782

[15] Chen, Min ; Miao, Yiming ; Humar, Iztok: *Introduction to OPNET Network Simulation.* S. 77–153. In: *OPNET IoT Simulation.* Singapore : Springer Singapore, 2019. – URL https://doi.org/10.1007/978-981-32-9170-6_2. – ISBN 978-981-32-9170-6

[16] Chun, Brent ; Culler, David ; Roscoe, Timothy ; Bavier, Andy ; Peterson, Larry ; Wawrzoniak, Mike ; Bowman, Mic: PlanetLab: An Overlay Testbed for Broad-Coverage Services. In: *SIGCOMM Comput. Commun. Rev.* 33 (2003), jul, Nr. 3, S. 3–12. – URL https://doi.org/10.1145/956993.956995. – ISSN 0146-4833

[17] CORP., NATIONAL I.: *Produkte.* – URL https://www.ni.com/de-de/shop.html. – Zugriffsdatum: 01.06.2023

[18] Coull, Scott E. ; Dyer, Kevin P.: Traffic Analysis of Encrypted Messaging Services: Apple IMessage and Beyond. In: *SIGCOMM Comput. Commun. Rev.* 44 (2014), oct, Nr. 5, S. 5–11. – URL https://doi.org/10.1145/2677046.2677048. – ISSN 0146-4833

[19] Danezis, George: The Traffic Analysis of Continuous-Time Mixes. In: *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*. Berlin, Heidelberg : Springer-Verlag, 2004 (PET'04), S. 35–50. – URL https://doi.org/10.1007/11423409_3. – ISBN 3540262032

[20] Díaz Zayas, Almudena ; García, Bruno ; Merino, Pedro: An End-to-End Automation Framework for Mobile Network Testbeds. In: *Mobile Information Systems* 2019 (2019), 04, S. 1–8

[21] Ede, Thijs ; Bortolameotti, Riccardo ; Continella, Andrea ; Ren, Jingjing ; Dubois, Daniel ; Lindorfer, Martina ; Choffnes, David ; Steen, Maarten ; Peter, Andreas: FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In: *Proceedings 2020 Network and Distributed System Security Symposium*, Internet Society, 01 2020. – URL https://www.ndss-symposium.org/ndss2020/. – NDSS 2020

[22] Edgar, Thomas W. ; Manz, David O.: Chapter 13 - Instrumentation. In: Edgar, Thomas W. (Hrsg.) ; Manz, David O. (Hrsg.): *Research Methods for Cyber Security*. Syngress, 2017, S. 321–344. – URL https://www.sciencedirect.com/science/article/pii/B9780128053492000133. – ISBN 978-0-12-805349-2

[23] Edgar, Thomas W. ; Rice, Theora R.: Experiment as a service. In: *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, IEEE, 2017, S. 1–6

[24] Elliott, Chip ; Falk, Aaron: An Update on the GENI Project. In: *SIGCOMM Comput. Commun. Rev.* 39 (2009), jun, Nr. 3, S. 28–34. – URL https://doi.org/10.1145/1568613.1568620. – ISSN 0146-4833

[25] Fortier, Paul J. ; Michel, Howard E.: 1 - Introduction. In: Fortier, Paul J. (Hrsg.) ; Michel, Howard E. (Hrsg.): *Computer Systems Performance Evaluation and Prediction*. Burlington : Digital Press, 2003, S. 1–38. – URL https://www.sciencedirect.com/science/article/pii/B9781555582609500011. – ISBN 978-1-55558-260-9

[26] Fu, Xinwen ; Graham, B. ; Bettati, R. ; Zhao, Wei: Active traffic analysis attacks and countermeasures. In: *2003 International Conference on Computer Networks and Mobile Computing, 2003. ICCNMC 2003.*, 2003, S. 31–39

[27] Galindo, José: *Evolution, testing and configuration of variability systems intensive*, Dissertation, 03 2015

[28] Herrmann, Dominik ; Wendolsky, Rolf ; Federrath, Hannes: Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In: *Proceedings of the ACM Conference on Computer and Communications Security*, ACM, 11 2009

[29] Houmansadr, Amir ; Kiyavash, Negar ; Borisov, Nikita: Non-Blind Watermarking of Network Flows. In: *IEEE/ACM Trans. Netw.* 22 (2014), Nr. 4, S. 1232–1244. – URL https://doi.org/10.1109/TNET.2013.2272740

[30] Kaye, David UN. Human Rights Council. Special Rapporteur on the P. ; Opinion, Protection of the Right to Freedom of ; Expression: *Report of the Special Rapporteur on the Promotion and Protection of the Right to Freedom of Opinion and Expression, David Kaye.* may 2015. – URL https://digitallibrary.un.org/record/798709/files/A_HRC_29_32-EN.pdf?ln=en. – Zugriffsdatum: 25.05.2023

[31] Kelly, Sanja ; Truong, Mai ; Shahbaz, Adrian ; Earp, Madeline: *Silencing the messenger: Communication apps under pressure.* 2016. – URL https://freedomhouse.org/report/freedom-net/2016/silencing-messenger-communication-apps-under-pressure#anchor-one. – Zugriffsdatum: 12.07.2023

[32] Meister, Andre: *Chatkontrolle: EU-Staaten Wollen Verschlüsselung doch nicht schützen.* Jul 2023. – URL https://netzpolitik.org/2023/chatkontrolle-eu-staaten-wollen-verschluesselung-doch-nicht-schuetzen/. – Zugriffsdatum: 10.07.2023

[33] Nasr, Milad ; Bahramali, Alireza ; Houmansadr, Amir: DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.* New York, NY, USA : Association for Computing Machinery, 2018 (CCS '18), S. 1962–1976. – URL https://doi.org/10.1145/3243734.3243824. – ISBN 9781450356930

[34] Northcutt, Curtis G. ; Athalye, Anish ; Mueller, Jonas: *Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks*. 2021

[35] OpenTAP: *Open Source in Test Automation*. 2022. – URL https://opentap.io/assets/whitepapers/Open-Source-in-Test-Automation-v5.pdf. – Zugriffsdatum: 01.06.2023

[36] OpenTAP: *opentap FAQ*. Apr 2023. – URL https://doc.opentap.io/FAQ/. – Zugriffsdatum: 01.06.2023

[37] Panizo, Laura ; Salmerón, Alberto ; Gallardo, María-del-Mar ; Merino, Pedro: Guided Test Case Generation for Mobile Apps in the TRIANGLE Project: Work in Progress. In: *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*. New York, NY, USA : Association for Computing Machinery, 2017 (SPIN 2017), S. 192–195. – URL https://doi.org/10.1145/3092282.3092298. – ISBN 9781450350778

[38] Park, Kyungwon ; Kim, Hyoungshick: Encryption is Not Enough: Inferring User Activities on KakaoTalk with Traffic Analysis. In: Kim, Ho-won (Hrsg.) ; Choi, Dooho (Hrsg.): *Information Security Applications*. Cham : Springer International Publishing, 2016, S. 254–265. – ISBN 978-3-319-31875-2

[39] Rakotoarivelo, Thierry ; Ott, Maximilian ; Jourjon, Guillaume ; Seskar, Ivan: OMF: A Control and Management Framework for Networking Testbeds. In: *SIGOPS Oper. Syst. Rev.* 43 (2010), jan, Nr. 4, S. 54–59. – URL https://doi.org/10.1145/1713254.1713267. – ISSN 0163-5980

[40] Reuter, Markus: *Stop CSAM act: Neues Gesetz in den USA könnte verschlüsselung schwächen*. Apr 2023. – URL https://netzpolitik.org/2023/stop-csam-act-neues-gesetz-in-den-usa-koennte-verschluesselung-schwaechen/. – Zugriffsdatum: 10.07.2023

[41] Rudl, Tomas: *Plattformregulierung: Vereinigtes Königreich Plant Kindersicherung fürs internet*. Dec 2020. – URL https://netzpolitik.org/2020/vereinigtes-koenigreich-plant-kindersicherung-fuers-internet/. – Zugriffsdatum: 11.07.2023

[42] Schulz, Hoboken: *Human rights and encryption*. UNESCO, 2016. – ISBN 978-92-3-100185-7

[43] Siboni, Shachar ; Sachidananda, Vinay ; Meidan, Yair ; Bohadana, Michael ; Mathov, Yael ; Bhairav, Suhas ; Shabtai, Asaf ; Elovici, Yuval: Security Testbed for Internet-of-Things Devices. In: *IEEE Transactions on Reliability* 68 (2019), Nr. 1, S. 23–44

[44] Tekeoglu, Ali ; Tosun, Ali ⋮ A Testbed for Security and Privacy Analysis of IoT Devices. In: *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2016, S. 343–348

[45] Turner, Hamilton ; White, Jules ; Reed, Jeffrey ; Galindo, José ; Porter, Adam ; Marathe, Madhav ; Vullikanti, Anil ; Gokhale, Aniruddha: *Building a Cloud-Based Mobile Application Testbed.* S. 382–403, 11 2012. – ISBN 9781466625372

[46] Wijermars, Mariëlle ; Lokot, Tetyana: Is Telegram a "harbinger of freedom"? The performance, practices, and perception of platforms as political actors in authoritarian states. In: *Post-Soviet Affairs* 38 (2022), Nr. 1-2, S. 125–145. – URL https://doi.org/10.1080/1060586X.2022.2030645

[47] Xiao-liang, Xu ; Le-yu, Wang ; Hong, Zhou: An object-oriented framework for automatic test systems. In: *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.*, IEEE, 2003, S. 407–410

[48] Zach Rosson, Carolyn T.: *Weapons of control, shields of impunity: Internet shutdowns in 2022.* Feb 2023. – URL https://www.accessnow.org/wp-content/uploads/2023/05/2022-KIO-Report-final.pdf. – Zugriffsdatum: 12.07.2023

[49] Zhang, Youquan ; Li, XuWen ; Wu, Qiang ; Liu, Jie: Design and Implementation of Automatic Testing Software Based on Labview and TestStand. In: *Proceedings of the 8th International Conference on Computing and Artificial Intelligence.* New York, NY, USA : Association for Computing Machinery, 2022 (ICCAI '22), S. 265–273. – URL https://doi.org/10.1145/3532213.3532253. – ISBN 9781450396110

[50] Zhang, Ziwei ; Ye, Dengpan: Defending against Deep-Learning-Based Flow Correlation Attacks with Adversarial Examples. In: *Security and Communication Networks* 2022 (2022), Mar, S. 2962318. – URL https://doi.org/10.1155/2022/2962318. – ISSN 1939-0114

[51] Zhu, Ye ; Fu, Xinwen ; Graham, Bryan ; Bettati, Riccardo ; Zhao, Wei: Correlation-Based Traffic Analysis Attacks on Anonymity Networks. In: *IEEE Transactions on Parallel and Distributed Systems* 21 (2010), Nr. 7, S. 954–967

A Anhang

Glossar

Nachrichtenprotokoll Eine `.json` Datei welche den Ablauf von aufgezeichneten Nachrichten und deren Inhalten beschreibt..

TAP Ein Testautomatisierungs Framework entwickelt durch 'Keysight Technologies' in Kooperation mit Nokia.

Unix Time Die Unix-Time oder auch Epoch-Time ist ein System zur darstellung von Zeit, indem die Anzahl der Sekunden seit dem 1. Januar 1970 um 00:00 Uhr UTC (Koordinierte Weltzeit) verwendet wird.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original