Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

SEMINARARBEIT

Privatsphärenprobleme bei Telegram Bots?

Student:
Thomas Koscheck

CyberSecurity
Semester 4

11. Juni 2020

HOCHSCHULE ALBSTADT-SIGMARINGEN

Zusammenfassung

IT Security
Semester 4

Privatsphärenprobleme bei Telegram Bots?

von Thomas Koscheck

Bots auf Messenger Plattformen erfreuen sich in den letzten Jahre immer größerer Beliebtheit. Der Messengerdienst Telegram bietet eine kostenlose API an, mit der jeder eigene Bots schreiben kann. Da in Gruppen, in denen auch Bots sein können, auch oft sensible Daten ausgetauscht werden, wird in dieser Seminararbeit der Frage nachgegangen, ob es eine Sicherheitslücke bei Telegram Bots und der Telegram API gibt und ob damit persönliche und sensible Daten - auch aus der Vergangenheit - abgegriffen werden können.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Einführung ins Thema

Bots als kleine Softwareprogramme sind aus unserem heutigen Alltag kaum mehr wegzudenken. Sie erfüllen meist hilfreiche und nützliche Funktionen, wie die eines 24h Chat- und Supportbots auf unserer Lieblingswebseite oder als Assistent für die persönliche Terminplanung, sie bringen uns automatisiert Nachrichten wie das aktuelle Wetter für meinen Standort oder helfen uns beim Sortieren unseres Posteinganges.

Besonders praktisch sind Bots dann, wenn sie auch auf unsere Lieblingsplattformen wie Facebook, Twitter oder Telegram verfügbar sind. Gerade auf Messagingplattformen wie Telegram erfreuen sich Bots immer größerer Beliebtheit [1] und der Funktionsradius wird immer größer. Es lassen sich einfache Abstimmungen in großen Gruppen mit Bots treffen, Nachrichtenseiten wie z.B. die Tageschau publizieren ihre News über einen Bot [2], man kann Spiele gegeneinander spielen, Fotos bearbeiten, Waren bestellen oder das eigene Smarthome steuern.

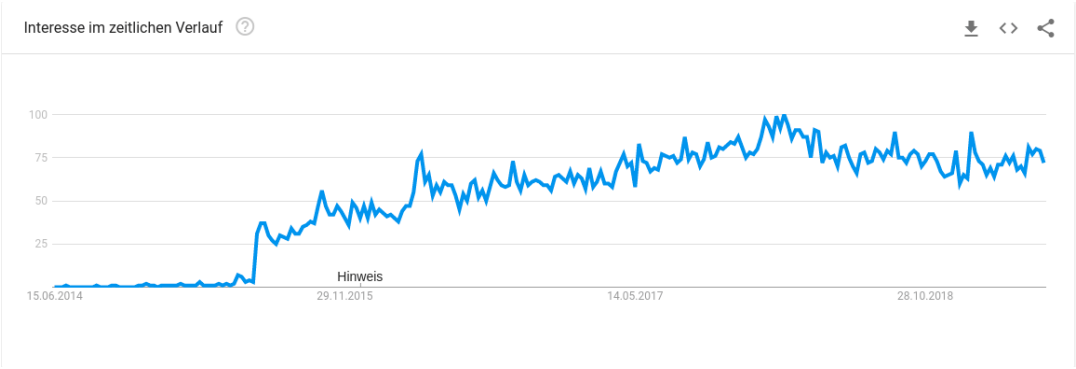

ABBILDUNG 1.1: Interesse am Suchbegriff "telegram bot‘ im zeitlichen Verlauf bei Google

Für Telegram kann prinzipiell jeder Nutzer eigene Bots erstellen und auf eigener Hardware laufen lassen, passende Frameworks gibt es dazu in fast jeder Programmiersprache [3].

1.2 Grundlage der Seminararbeit

Das Thema dieser Seminararbeit basiert auf einem am 18.01.2019 auf der heise online Webseite in der Kategorie Security erschienenen Artikel mit dem Titel Telegram-Bots

lassen sich anzapfen [4]. Sicherheitsforscher warnen vor Privatssphären-Problemen bei Telegrambots, da unter bestimmten Umständen sämtliche Kommunikation an Außenstehende gelangen könne, sogar bei geschlossenen Gruppen.

Kapitel 2

Vorbereitende Informationen

2.1 Der privacy mode

Telegram Bots haben einen privacy mode, der vom Entwickler angeschalten oder ausgeschalten werden kann. Als Standardeinstellunge ist der privacy mode aktiviert [5]. Die zwei verschiedenen Modi regeln, welche Nachrichten innerhalb einer Gruppe ein Bot sehen kann.

Alle Bots bekommen - egal welche Einstellungen getroffen wurden - folgende Nachrichten:

- Alle Nachrichten von privaten Chats mit Nutzern
- Alle Nachrichten von Kanälen, in denen sie ein Mitglied sind

Bots mit Adminrechten oder deaktiviertem privacy mode bekommen alle Nachrichten, außer die von anderen Bots.

Bots mit aktiviertem privacy mode erhalten diese Nachrichten:

- Kommandos, die speziell an den Bot gerichtet sind (z.B. /command@this_bot)
- Nachrichten, die über den Bot geschickt werden
- Antworten, auf jede Nachricht, die implizit oder explizit an den Bot gerichtet sind

In Gruppen ist unter dem Punkt *Mitglieder* einsehbar, welchen privacy mode ein Bot hat. Hier steht entweder die Meldung *Zugriff auf Nachrichten* oder *Kein Zugriff auf Nachrichten* bei einem aktiviertem privacy mode [5]

Die Einstellung, ob ein Bot Adminmitglied einer Gruppe ist oder nicht, lässt sich nur clientseitig in der entsprechenden Gruppe setzen und ist auch nicht mit einem API Token änderbar.

2.2 Verschlüsselung in Telegram

Nachrichten in Telegram zwischen zwei normalen Usern werden durch die eigenen Verschlüsselung MTProto geschützt [6]. In der Telegram Bot API werden die Nachrichten allerdings nur durch TLS geschützt und sonst nicht weiter verschlüsselt [7].

2.3 Zugriff auf alte Daten

Laut API Dokumentation von Telegram kann ein Bot kann nur auf Nachrichten zugreifen, ab dem Zeitpunkt, zu dem er zu einer Gruppe hinzugefügt wurde. Mit aktiviertem privacy mode sogar nur die Nachrichten, die auch explizit an ihn gerichtet waren. Weiterhin schreibt Telegram, alte Nachrichten nach einer nicht genauer spezifizierten Zeit für Bots zu löschen [8].

Kapitel 3

Betrachtung des Angriffes

3.1 Einleitung

Der eigentliche heise-Artikel [4] beschreibt nur sehr rudimentär die Schwachstelle und die Details dahinter, er verweist stattdessen auf einen Eintrag im Blog [9] der Firma Forcepoint, die auch die Schwachstelle gefunden hat. Daher wird diese Arbeit sich hauptsächlich auf den Blogeintrag der Firma Forcepoint beziehen.

3.1.1 Forcepoint

Forcepoint ist ein amerikanisches Unternehmen, das dem US - Verteidigungsunternehmen *Raytheon* und der Private-Equity-Firma *Vista Equity Partners* gehört. Sie entwickelt und vermarktet Cybersicherheitssoftware, sowie Firewall, Cloud Access und domänenübergreifende IT-Sicherheitsprodukte (vgl. [10]).

3.2 Entdeckung der Schwachstelle

Forscher von Forcepoint haben im Zuge einer Untersuchung von existierenden Bedrohungen eine Malware gefunden, die Telegram als Command and Control (C2) Infrastruktur genutzt hat.
Die Funktionsweise der Malware - getauft auf den Namen *GoodSender* - ist recht simpel. Sobald sich die Malware auf einem Rechner befindet, erstellt sie einen neuen Adminuser, aktiviert den Remote-Desktop und öffnet die Firewall. Der Username für den neuen User ist immer gleich, das zugehörige Passwort wird aber zufällig erstellt. Die nun erbeuteten Informationen, also Username, Passwort und IP-Adresse des Opfers werden nun über Telegram verschickt [9].

3.3 Beschreibung des Angriffes

Um die Kontrolle über einen Telegram Bot zu bekommen und damit auch potentiell alle Nachrichten mit lesen zu können, sind zwei wichtige Informationen nötig. Zum einen der Bot API Token, welcher in jeder Nachricht mitgeschickt wird und auch in jedem Programm, was die API verwenden will, enthalten sein muss.
Die andere wichtige Information ist die zufällig generierte Telegram `chat_id`. In individuellen Chats hat jeder User seine eigene und einmalige ID [11]. Auch diese Information wird bei jedem API Request mitgeschickt, da der Bot wissen muss, an welchen User/Gruppe er seine Antwort senden muss.

Die Forscher von Forcepoint schreiben nun:

> *"Thus, an attacker in MitM position with capability to decrypt TLS could gain access to the bot token as well as the chat_id leading to a full compromise of not just the current communication but all previous communication to which the bot was party as well."* [9]

Die Behauptung ist also, dass Angreifer in einer MitM-Position, die dazu in der Lage sind eine TLS - Verschlüsselung aufbrechen zu können, nun den API Token und die chat_id abgreifen können.

Die Telegram API bietet nun eine Funktion forwardMessage() an, mit der man jede Nachricht aus jedem Chat, auf die der Bot Zugriff hat, an jeden beliebigen anderen Telegram Nutzer weiterleiten kann, also auch an den Angreifer.
Mit dem Wissen über diese API Funktion und der Informationen, dass die message_id in einem Chat inkrementell von 0 beginnend wächst [9], lässt sich nun ein einfaches Skript schreiben, um alle Nachrichten die jemals in einen Chat geschrieben wurden (in dem der Bot aktuell eingeladen ist) weiterzuleiten.

3.4 Validierung des Angriffes

3.4.1 Untersuchung TLS-Verbindung zur Telegram API

Ein Test mit dem Dienst SSL Labs [12] ergibt folgende Informationen über die Telegram API im Bezug auf unterstützte TLS/SSL Versionen (Stand Mai 2019):

TABELLE 3.1: TLS Versionen

Protokoll	**Unterstützt**
TLS 1.3	Nein
TLS 1.2	Ja
TLS 1.1	Ja
TLS 1.0	Ja
SSL 3	Nein
SSL 2	Nein

Mit dem Tool testssl.sh [13] habe ich die Telegram API auf bekannte TLS Angriffe getestet. Bis auf einen Angriff ist die API aber sicher.

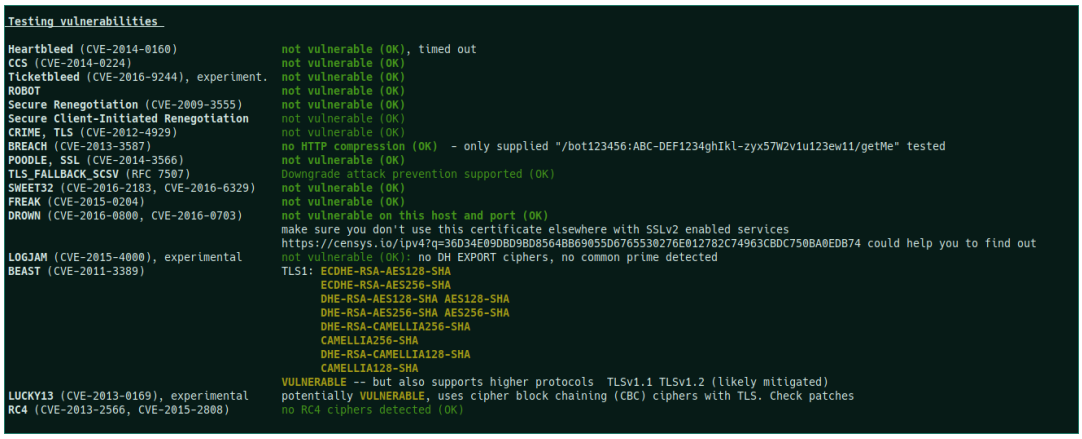

ABBILDUNG 3.1: Schwachstellentest mit testssl.sh

3.4.2 Detailbetrachtung: BEAST

BEAST (Browser Exploit Against SSL/TLS) - beschrieben in CVE-2011-3389 - ist ein Browser Exploit, der Ende September 2011 veröffentlicht wurde. TLS 1.0 und frühere Protokolle haben das Problem, dass der Initialisierungsvektor (IV) von einem aktiven Man-in-the-Middle (MITM)-Angreifer vorhergesagt werden kann.
IVs werden verwendet, um zu verhindern, dass die Verschlüsselung deterministisch ist; ohne sie erhalten Sie jedes Mal, wenn Sie den gleichen Datenblock mit dem gleichen Schlüssel verschlüsseln, die gleiche (verschlüsselte) Ausgabe. Technisch kann er keine Daten entschlüsseln, aber er kann herausfinden, ob seine Vermutungen richtig oder falsch sind. Mit genügend Vermutungen können beliebig viele Daten aufgedeckt werden.

Da das Schätzen nicht sehr effizient ist, kann der BEAST-Angriff in der Praxis verwendet werden, um nur kleine Datenfragmente wie HTTP-Session-Cookies, Authentifizierungsdaten usw. abzurufen.
BEAST ist eine reine client-seitige Schwachstelle. Seit der Veröffentlichung haben die meisten Hersteller (vor allem Browser-Hersteller) es mit einer Technik namens 1/n-1-Split behandelt. Diese Technik hindert den Angreifer daran, IVs vorherzusagen und behebt effektiv das zugrunde liegende Problem. (vgl. [14])

Für das Extrahieren des API Tokens und der chat_id wäre dieser Angriff also geeignet, wenn auch sehr aufwendig.

Schutzmaßnahmen um Angriffe auf TLS zu unterbinden wurden dabei folgende getroffen (Analyse mit testssl.sh):

TABELLE 3.2: Schutzmaßnahmen

Schutzmaßnahme	**Unterstützt**
Downgrade attack prevention	Yes, TLS_FALLBACK_SCSV supported
HSTS	Aktiviert
HSTS Preloading	Chrome, Edge, Firefox, IE

3.5 Eigene Umsetzung des Angriffes

3.5.1 Programmierung eines eigenen Bots

In wenigen Zeilen lässt sich der von den Forcepoint-Forschern beschriebene Angriff zumindest theoretisch umsetzen. Für den Bot wurde die Programmiersprache JavaScript und das Framework Telegraf [15] verwendet.

Der Bot liest die message_id aus, die die letzte Nachricht an den Bot hat. Da der Bot selbst erstellt wurde, musste deswegen nicht zurerst TLS geknackt werden um an den API Token zu kommen. Die ebenso wichtige chat_id bekommt der Bot auch, sobald er eine Nachricht empfängt. Mit einer einfachen while-Schleife hat der Bot dann über die erwähnt forwardMessage() - Funktion versucht, alle vergangenen Nachrichten an einen anderen Account weiterzuleiten. Details zum Source Code finden sich im Anhang A und A.1.

3.5.2 Realer Test

Um den Proof-of-Concept durchzuführen, ist eine Gruppe auf Telegram mit mehreren Personen erstellt worden, die verschiedene Textnachrichten in die Gruppe geschrieben haben. Danach wurde der Bot hinzugefügt und weitere Nachrichten ausgetauscht.

Wenn der oben erwähnt privacy mode deaktiviert ist, ist es für den Bot möglich, alle Nachrichten in der Gruppe zu empfangen (auch die, die nicht explizit mit einem @Bot an ihn gerichtet sind). Über die forwardMessage() - Funktion der Telegram API war es auch einfach möglich, alle Nachrichten aus der Gruppe an den zweiten Account weiterzuleiten.

Kapitel 4

Auswertung des Angriffes

4.1 Aktivierung des privacy modes

Um zu vermeiden, dass der Bot Zugriff auf alle Nachrichten einer Gruppe hat, sollte er nie als Administrator hinzugefügt werden. Denn in diesem Fall hat er automatisch Zugriff auf alle Nachrichten [5]. Weiterhin sollte man den privacy mode des Bots beachten und falls dieser deaktiviert ist, sich überlegen, inwiefern man dem Hersteller traut.

4.2 Extrahierung alter Nachrichten

Dazu dürfen die alten Nachrichten noch nicht von Telegram gelöscht worden sein und der Bot sollte einen deaktivierten privacy modehaben (Sonst sind nur die direkten Befehle an den Bot zu extrahieren, welche meist nicht besonders spannend sein dürften).

Anders als von Forcepoint behauptet, erlaubt Telegram sowohl das Weiterleiten als auch das Zugreifen auf alle Nachrichten in der Gruppe bevor der Bot hinzugefügt wurde nicht. Ist ein Bot über längere Zeit Mitglied einer Gruppe, können sich aber natürlich auch in dieser Zeit eine große Menge an sensiblen Daten anhäufen.

Weiterhin ist es nicht erlaubt, mehr als eine Nachricht pro Sekunde über die API zu senden, damit sollte sich das Extrahieren von vielen Nachrichten länger ziehen [16].

4.3 Aufbrechen von TLS

Telegram erlaubt den ausschließlichen Gebrauch von TLS für die API [5]. Bei meinen Test habe ich keine praktikable Schwachstelle durch gängige Angriffe gefunden. Ein MitM Angriff und die Extrahierung des Bot Tokens und der chat_id aus der TLS Verbindung sind aber trotzdem nicht ausgeschlossen.

Genauso betroffen sind davon aber auch alle Banking-Applikation oder generell alle anderen Anwendungen/Webseiten die sensible Daten transportieren.

4.4 Statement Telegram

Telegram hat kurz nach bekanntwerden des Artikel folgendes Statement dazu veröffentlicht (gekürzt).

> We just got alerted to a new instance of the old *if I had your keys, I could get into your home* story. [...]
>
> **If I could break HTTPS...**
> HTTPS (TLS) is the industry standard that is used not only by the bot platforms of Telegram, Slack, Discord, Kik and Messenger, but also by your bank and gadzillions of other services.
>
> **What you get for "breaking HTTPS"**
> By default, Telegram bots only get the messages specifically meant for them. That is, messages which have a /command for the bot, @mention the bot by its username or reply to one of its messages.
>
> So if you broke HTTPS, you could go hunting for a bot's token in hope to recover something useful from the messages it gets. In most cases, you would only get a list of /commands for the bot to do something.
>
> Instead, our advice for anyone who breaks HTTPS would be to head over to the nearest bank and transfer a few billion dollars into their private account. This would make breaking HTTPS a lot more fun and profitable [17].

4.5 Alternativen

Auch wenn es bei der Telegram Bot API an manchen Stellen noch Verbesserungspotenzial gibt, eine große Auswahl an Plattformen gibt es nicht. Und wer private Projekte damit umsetzen will, hat leider keine andere Möglichkeit, als auf Telegram zurück zu greifen. Denn die Bot API anderer Messenger sind meistens nur offen für Unternehmen. Nachfolgend eine Aufzählung bekannteren Messenger, die überhaupt eine API anbieten.

4.5.1 Facebook

Über Sicherheitsaspekte bei Bots und Verschlüsselung findet sich in der Dokumentation von Facebook nichts. Außerdem muss ein Bot eingereicht werden und dann nach manueller Prüfung freigegeben werden [18].

4.5.2 WhatsApp Business API

Hier werden Nachrichten auch bei Bots durch die Whatsapp Verschlüsselung geschützt. Dafür ist der initiale Aufwand auch schon deutlich höher: Es muss z.B. eine SQL-Datenbank für die Schlüsselverwaltung eingebunden werden. TLS ist hier aber keine Pflicht für API Aufrufe.

Auch bei Whatsapp muss ein Bot manuell freigegeben werden und eigentlich haben nur Firmen die Möglichkeit Bots zu erstellen [19].

4.6 Fazit

Das Statement von Telegram fasst die aktuelle Lage ziemlich gut zusammen. Mit dem richtigen Token lässt sich jede API änzapfenünd für Telegram war es eine bewusst getroffene Entscheidung, Nachrichten für Bots nicht mit MTProto zu verschlüsseln, um die Einstiegshürde für Entwickler möglichst gering zu halten.

Da alle Nachrichten weiterhin mit TLS geschützt sind und bis auf BEAST kein bekannter Angriff gegen die API funktioniert, ist der Aufwand, der gemacht werden müsste um an chat_id und API Token zu kommen, vergleichsweise hoch. Dennoch wäre ein solcher Angriff denkbar. Wer aber die Fähigkeit hat einen MiTM Angriff durchzuführen und TLS unbemerkt zu brechen, hat meist lohneswertere Ziele wie BankingApps, Logindaten auf Webseiten und verschickte Nachrichten von anderen Applikationen.

Wenn API Token und chat_id einmal bekannt sind, ist es sehr einfach Nachrichten, die an den Bot gesendet wurden an sich selbst weiterzuleiten. Da aber defaultmäßig immer der privacy mode aktiviert ist, sollten sich diese Nachrichten meistens auf Kommandos an den Bot beschränken, die für einen Angreifen nicht besonders interessant sein sollten. Anders als im Artikel behauptet, ist es auch nicht möglich Nachrichten aus Gruppen zu extrahieren, die verschickt wurden bevor der Bot hinzugefügt wurde.

Anhang A

Telegram Bot Source Code

A.1 Minimal version

```
1  const Telegraf = require('telegraf');
2
3  const chatIdMe = 123456789;
4  const bot = new Telegraf(process.env.BOT_TOKEN);
5
6  bot.on("text", (ctx) => {
7      forwardOldMessages(ctx.message.message_id, ctx);
8  });
9
10 function forwardOldMessages(actualMessageId, ctx) {
11     while (actualMessageId > 0){
12         bot.telegram.forwardMessage(chatIdMe, ctx.message.chat.id,
               actualMessageId);
13         actualMessageId--;
14     }
15 }
16
17 bot.launch();
```

A.2 Extended version

```
1  const Telegraf = require('telegraf');
2
3  const chatIdMe = 123456789;
4  const bot = new Telegraf(process.env.BOT_TOKEN);
5  bot.use(Telegraf.log());
6
7  bot.on("text", (ctx) => {
8      let text = ctx.message.text;
9      forwardOldMessages(ctx.message.message_id, ctx);
10     return ctx.replyWithMarkdown(
11         '*Message id: *' + ctx.message.message_id + '\n' +
12         '*From id*: ' + ctx.message.from.id + '\n' +
13         '*From is bot*: ' + ctx.message.from.is_bot + '\n' +
14         '*From first name*: ' + ctx.message.from.first_name + '\n' +
15         '*From username*: ' + ctx.message.from.username + '\n' +
16         '*From Language*: ' + ctx.message.from.language_code + '\n' +
17         '==============================' + '\n' +
18         '*Chat id*: ' + ctx.message.chat.id + '\n' +
19         '*Chat first_name*: ' + ctx.message.chat.first_name + '\n' +
20         '*Chat username*: ' + ctx.message.chat.username + '\n' +
21         '*Chat type*: ' + ctx.message.chat.type + '\n' +
22         '==============================' + '\n' +
23         '*Date*: ' + ctx.message.date + '\n' +
24         '*Text*: ' + text + '\n' +
25         '*Update id*: ' + ctx.update_id
26     );
27 });
28
29
30 function forwardOldMessages(actualMessageId, ctx) {
31     while (actualMessageId > 0){
32         bot.telegram.forwardMessage(chatIdMe, ctx.message.chat.id,
               actualMessageId);
33         actualMessageId--;
34     }
35 }
36
37 bot.start((ctx) => ctx.reply('Welcome'));
38 bot.launch();
```

Literatur

[1] GoogleTrends. *Google Trends, Suchbegriff: Telegram bot*. Mai 2019. URL: https://trends.google.de/trends/explore?date=today%205-y&q=telegram%20bot (besucht am 06. 04. 2019).

[2] Tagesschau. *Novi auch auf Telegram und Skype*. Juni 2018. URL: https://www.tagesschau.de/inland/novi-105.html (besucht am 06. 05. 2019).

[3] Telegram. *Telegram Bot Samples*. URL: https://core.telegram.org/bots/samples (besucht am 10. 06. 2019).

[4] Jürgen Schmidt. *Telegram-Bots lassen sich anzapfen*. Jan. 2019. URL: https://www.heise.de/security/meldung/Telegram-Bots-lassen-sich-anzapfen-4282130.html (besucht am 04. 06. 2019).

[5] Telegram. *Telegram Bot Übersicht*. Apr. 2019. URL: https://core.telegram.org/bots/ (besucht am 20. 05. 2019).

[6] Telegram. *Telegram Verschlüsselung*. URL: https://core.telegram.org/mtproto (besucht am 23. 06. 2019).

[7] Telegram. *Telegram Bot Verschlüsselung*. URL: https://core.telegram.org/bots#2-how-do-bots-work (besucht am 23. 06. 2019).

[8] Telegram. *Telegram Zugriff auf alte Daten*. URL: https://core.telegram.org/bots#4-how-are-bots-different-from-humans (besucht am 23. 06. 2019).

[9] Forcepoint. *Tapping Telegram Bots*. Jan. 2019. URL: https://www.forcepoint.com/blog/security-labs/tapping-telegram-bots (besucht am 07. 05. 2019).

[10] Wikipedia. *Forcepoint*. Aug. 2018. URL: https://en.wikipedia.org/wiki/Forcepoint (besucht am 07. 05. 2019).

[11] Telegram. *Telegram User chat_id*. URL: https://core.telegram.org/bots/api#user (besucht am 23. 06. 2019).

[12] SSL Labs. *SSL Labs*. URL: https://www.ssllabs.com/ssltest (besucht am 25. 05. 2019).

[13] testssl.sh. *testssl.sh Skript*. URL: https://github.com/drwetter/testssl.sh (besucht am 23. 06. 2019).

[14] SSL Labs. *Is BEAST Still a Threat?* Sep. 2013. URL: https://blog.qualys.com/ssllabs/2013/09/10/is-beast-still-a-threat (besucht am 09. 05. 2019).

[15] Telegraf. *Telegraf Framework*. URL: https://telegraf.js.org/ (besucht am 16. 06. 2019).

[16] Telegram. *Telegram API Limit*. URL: https://core.telegram.org/bots/faq#my-bot-is-hitting-limits-how-do-i-avoid-this (besucht am 16. 06. 2019).

[17] Markus Ra. *Telegram Bots, Keys and Pockets*. Jan. 2019. URL: https://telegra.ph/Telegram-Bot-Keys-01-17 (besucht am 25. 05. 2019).

[18] Facebook Messenger Developer. *Facebook for Developers*. URL: https://messengerdevelopers.com (besucht am 24. 05. 2019).

[19] WhatsApp. *WhatsApp Business API*. URL: https://developers.facebook.com/docs/whatsapp (besucht am 24. 05. 2019).