Bachelor thesis
Computing Science

Radboud University

You've Been Previewed - Information Leaks of Content Previews in Mobile Messengers

Author:
Stefan Popa
1027672

First supervisor/assessor:
Asst. Prof. Katharina Kohls
kkohls@cs.ru.nl

Second assessor:
Asst. Prof. Hugo Jonker
hugo.jonker@ou.nl

January 7, 2021

Abstract

Messenger apps have become one of the main methods for people to share and access information, because of their ease of use and extensive functionality. One of the important features enhancing this communication are the link previews. They allow the user to receive a preview of the website being shared within a text message by showing a title, description, and an image. Unfortunately, implementing this feature in mobile messengers means that the information exchanged and the requests made during preview creation can leak information.

In this thesis, we try to identify the issues that exist in the implementation of link previews in various mobile messengers and analyze the respective privacy-critical information leaks. To this end, we design a set of controlled experiments consisting of two devices sharing a link with each other, for a total of 16 regular and end-to-end encrypted instant messaging apps. Our results show that by only making use of the server logs, an adversary can extract sensitive user information from the requests made during link preview generation.

Contents

Chapter 1

Introduction

The way people access information on the Internet has shifted. This is shown by the number of mobile phone users, which is predicted to grow by 630 million between 2015 and 2020 [23], and by the number of Facebook Messenger and WhatsApp users, which has reached 1.6 billion and 1.3 billion respectively in 2019 [30]. This means that more and more people are using mobile messenger apps to share news, stories, and opinions. Implicitly, due to their increasing importance and user base, mobile messengers also often make the news when new vulnerabilities or attacks are brought to light. Quite recently, WhatsApp has been the target of spyware originating from an Israeli company [11] and a bug allowing an attacker to eavesdrop on any Facebook Messenger user has been discovered [15].

Extensive research has been done about the most popular messengers and their potential issues. For instance, previous work has discussed and analyzed the different protocols implemented by the most popular messengers [12][25], their security [13][19] or frameworks for analyzing security of mobile messengers [26]. Additionally, the usability of the encryption protocols implemented in mobile messengers [20][27] and possible attack vectors resulting from a large number of required privileges or information stored on the local storage [18] has also been considered.

While past work on mobile messengers has mostly focused on cryptographic, privacy and usability aspects, smaller features have not been extensively discussed. One such feature are the link previews. When a user sends a link to a website, most mobile messenger apps will display a preview of the website inside the app, such that the receiving user can decide whether or not it is of interest. This implies that a request will be made to the server hosting the content without any of the users opening it. The request might contain information that allows an attacker to find something sensitive about the user.

For example, a possible scenario is that the information obtained from that request is enough to conduct a localization attack on users using IP Geolocation. There are many ways for performing geolocalization, such as trilateration and Geofeeds [9][14]. Furthermore, existing work has looked at different attacks resulting from localization vulnerabilities, such as prefix hijacking [8] or exploiting proxies [28]. Thus, if we are able to obtain a user's location by only sending some messages, it would be quite a bad vulnerability. This is just one of the possibilities. An attacker could also be able to deduce what messenger app has sent the request to the server, based on the number and type of headers the request contains [22]. Therefore, we center our work around the following research question:

"What privacy-critical information leaks can be found in the link preview feature of modern messengers?"

In other words, the focus of the thesis is to get a better understanding of privacy-critical information leaks found in the link preview generation and to document the results. To this end, we (i) use a remote server for hosting an experimental website, (ii) exchange text messages containing the URL to the website between two or more devices, using various messenger apps, (iii) search the server logs for the relevant requests and (iv) parse the information found in these requests and documenting and discussing our findings. This approach differs from previous work on link previews, which either focused on how the lack of standardization and countermeasures results in link previews being used for malicious purposes [24] or on possible attack vectors arising from monitoring the traffic generated by the link preview generation [22][7].

The content of the following chapters is structured as follows: **Chapter 2** presents the technical information necessary for understanding the experiments we conduct in our work and the related results. **Chapter 3** describes the concept of link preview generation and its different parts as well as the attacker model. **Chapter 4** discusses the technicalities of our experiment and the selection process of the candidate messenger apps. The chapter also outlines the experimental and analysis process. **Chapter 5** summarizes the various results obtained for 16 different mobile messengers. In **Chapter 6** our findings and their implications are discussed, e.g., discovering the originating device based on its IP address, fingerprinting the various messenger apps or being an important step of a larger attack. The limitations of our experimental process are also presented here. Finally, we conclude this work in **Chapter 7** and summarize our analysis and findings.

Chapter 2

Technical Background

Before we present and discuss our experiment and findings, we introduce important background information. Here, we first describe the basis of the HTTP protocol and the features relevant for our study. We continue by shortly explaining the Same Origin Policy, followed by an introduction to the role that the server and server logs have in our thesis. We also outline the link preview generation process and we end the chapter by comparing in-transit and end-to-end encryption.

2.1 Hypertext Transfer Protocol

Throughout the thesis we make use of the HTTP protocol, or Hypertext Transfer Protocol. Therefore we introduce the context in which HTTP is used, explain important concepts of the protocol's architecture and conclude the section with a description of the format used by the requests. HTTP responses are not relevant for our experiment and hence we do not explain their format.

HTTP is an application layer protocol which represents the building block for the communication of data on the World Wide Web. In other words, it specifies how computers should communicate with one another: how to retrieve data, how to transfer HTML and media files, what format should be used for the requests etc.

2.1.1 Architecture

HTTP is built using the client–server computing model. This means that communication is done using a request-response format. For example, a web browser (client) tries to access some content, such as a website, hosted on another computer (server). This web browser makes an HTTP request message to the server, and the server replies with a response message. This

response message contains status information about the request and may also include the content that was requested, such as an HTML webpage.

Often the content that needs to be accessed is hosted on a computer from a different network. This means that there needs to be a way for the client to specify to the protocol where can this content be found. This is done using the Uniform Resource Locator (URL). An URL is simply a string which contains information about the location of the resource that is being requested. A resource can be, for example, a website or a web service. The following is a general syntax for URLs:

```
"http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]
```

The format of the URL can be summarized as follows [2]:

(1) *host*: The host identifier can be either in the form of a canonical name (e.g., *"thesisstefan.science.ru.nl"*) or in the form of an IP address (e.g., *"134.21.3.4"*);

(2) *port*: The port which is attributed to the HTTP protocol on the server where the requested content can be found. It is often left unspecified, which means that port 80 is assumed. This is the default port for HTTP;

(3) *abs_path*: The absolute path on the server of the resource that is being requested;

(4) *query*: It can contain certain parameters that are needed for accessing the content, e.g., *"par1=value1&par2=value2"*;

2.1.2 Request Messages

When an HTTP client wants to communicate with an HTTP server, it sends a request containing the following information [2]:

- Request line;
- Zero or more header fields;
- An empty line, indicating the end of the header fields;
- An optional message body.

What exactly can be found in the header fields or the body of the message is not important here, therefore we only focus on the request line. The request line contains the request method, the request URL and the HTTP version in use, as follows:

Table 2.1: URL origin comparisons example

URL	Outcome	Reason
http://example.com**/dir2/other.html**	Same Origin	Path differs
http://example.com**/dir/onelvl/another.html**	Same Origin	Path differs
https://example.com/page.html	Failure	Different protocol
http://example.com:**81**/dir/page.html	Failure	Different port
http://**example.news.com**/dir/page.html	Failure	Different host

```
Request-Line = Method " " Request-URI " " HTTP-Version
```

There are various possible methods that can be used in a request. In this case, we focus on the **GET** method, which is seen throughout our work. The method is used to retrieve information from the server specified by the URL. This type of request only retrieves data and does not have any other effects on this data.

The request URL is simply an identifier for the resource that is being requested. For example, *"https://thesisstefan.science.ru.nl"* is the URL we use throughout our experiment. Finally, the HTTP version just specifies what HTTP implementation is the server using, which in our case is **HTTP/1.1**.

2.2 Same Origin Policy

The Same Origin Policy (SOP) is a critical mechanism for the security of web applications. The policy specifies that a web browser should allow a script loaded from one web page to access data from another web page only when the two web pages share the same origin. The main purpose of this policy is to isolate malicious third-party code, thus reducing the risk of sensitive data being accessed by potential attackers. SOP is the reason why most requests for generating link previews are done server-side, as explained in Section 2.4.

Two web pages have the same origin when their URLs share the protocol used, the port and the host. Table 2.1 offers some example origin comparisons for the URL **http://example.com/dir/page.html** [3].

2.3 Server

Because our analysis revolves around the concept of client-server communication, an important part of the experiment is the server. The server

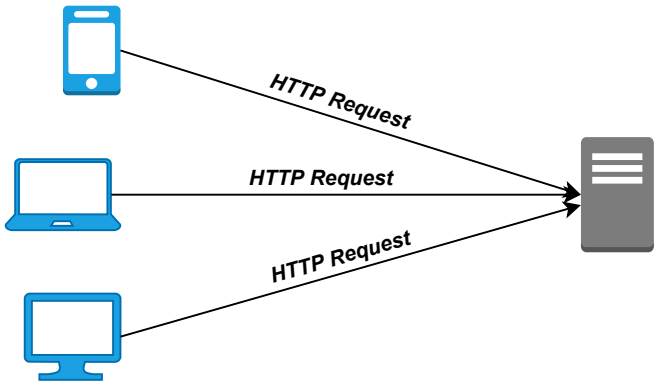

Figure 2.1: Different clients make HTTP Requests to a server for accessing some content

is nothing but a piece of computer hardware (or in some cases, computer software), which provides certain features to other devices (clients). These features include, but are not limited to, the sharing of information and resources, or performing certain tasks and computations.

2.3.1 Experimental Use

We use a remote Apache server to host some content (a website). Whenever an user device wants to access the website, it will make one or more HTTP requests to the server. The process is illustrated in Figure 2.1.

These requests are automatically recorded in the server logs. These logs can only be accessed by an administrator and are used for finding out spam content, broken external links and incorrect server responses, amongst others. For our experiment, we use the server logs as a means to gather information about the requests made by messengers for generating link previews.

2.3.2 Logs Format

Depending on the type of server, the logs can have various formats. The Apache server that we use formats these logs as follows:

```
92.110.122.85 thesisstefan.science.ru.nl - - [18/Nov/2020:18:57:26 +0100]
↪  "GET / HTTP/1.1" 200 9473 "-" "WhatsApp/2.20.205.16 A"
```

The format of the logs is explained below [1]:

(1) *92.110.122.85*: The IP address of the client which made the request to the server;

(2) *thesisstefan.science.ru.nl*: The canonical name of the server responding to the request;

(3) *[18/Nov/2020:18:57:26 +0100]*: The time at which the request was received;

(4) *"GET / HTTP/1.1"*: The request line from the client, given in double quotes;

(5) *200*: The status code returned by the server to the client;

(6) *9473*: The size of the server response in bytes, not including the headers;

(7) *"-"*: Referrer header of the HTTP request, in case the client has been reffered from another site;

(8) *"WhatsApp/2.20.205.16 A"*: The User-Agent HTTP request header.

2.4 Link Previews

A link preview or URL preview is a feature implemented in most modern messengers. It allows the app to send a preview of a website when typing the respective link in a message. The generation of link previews can be done either client-side or server-side. A client-side implementation consists of the messenger app using a script to fetch resources from the web page, while a server-side implementation has the messenger app send HTTP GET requests to the server hosting the content. Nevertheless, most implementations use server-side requests due to the restrictions arising from the same-origin policy (SOP) [16], [24].

The process of generating link previews works as follows [24]:

(1) The user types a URL in the input field of a mobile messenger platform;

(2) The platform fetches the URL and its resources using a series of HTTP GET requests;

(3) A preview is generated by the platform based on the fetched resources.

While link previews can be generated without the use of any special meta tags, usually web pages use such tags since they offer more control of what is displayed and allow for more detailed previews. Two popular meta tags

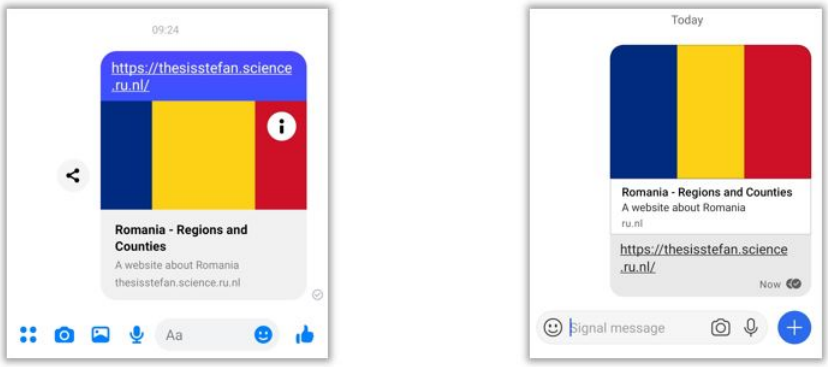

Figure 2.2: Example of a link preview as implemented in Facebook Messenger (left) and Signal (right)

protocols are OpenGraph [4] and Twitter Cards [5], with the former being more broadly supported [24].

The following is a simple example of OpenGraph and Twitter Cards meta tags. Here, the tags are used to provide a title and a description for the website, which will be shown in the link preview.

```
<meta property="og:title"       content="Romania - Regions and Counties">
<meta property="og:description" content="A website about Romania">
```

```
<meta name="twitter:title"       content="Romania - Regions and Counties">
<meta name="twitter:description" content="A website about Romania">
```

2.5 In-Transit & End-to-End Encryption

Throughout our work we distinguish between regular and end-to-end encrypted messengers. The former implement in-transit encryption, while the latter implement end-to-end encryption. This section aims to describe and explain the differences between both options.

Encryption is used to ensure that the data that is being sent over the Internet can only be read by the parties participating in the communication, and not by any bystander or adversary. Without encryption, it would be trivial to access and tamper with any kind of information, such as citizen information, passwords, online transactions etc. How exactly does encryption work in practice is not relevant for our work. What is important to understand is that when two parties communicate, the sending device uses an algorithm

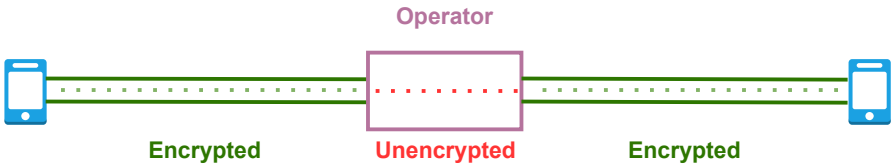

Figure 2.3: In-Transit Encryption in mobile messenger apps

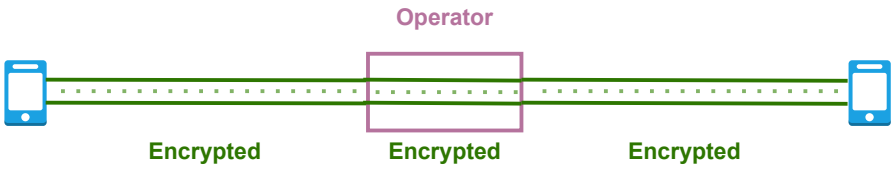

Figure 2.4: End-to-End Encryption in mobile messenger apps

and a key to encrypt the information, while the receiving device uses the same algorithm and a different key to decrypt the information. In our work we distinguish between in-transit encryption and end-to-end encryption.

2.5.1 In-Transit Encryption

In the case of In-Transit Encryption, the communication data is only being encrypted between the user device(s) and the operator itself (e.g., messenger app operator). This means that the operator handles the data internally in an unencrypted form. Figure 2.3 illustrates this type of encryption.

Because the service provider is able to access all the data exchanged, in-transit encryption implies that the user needs to trust them to handle this data safely and appropriately. Therefore, if the operator is compromised by an attack, or the information gets corrupted, communication will be affected.

2.5.2 End-to-End Encryption

In this case of End-to-End Encryption, the communication data is encrypted throughout the whole path between the sending user device and the receiving user device(s). The operator is only capable of forwarding the messages to the correct recipients, but it cannot read what the messages contain. Figure 2.4 illustrates this type of encryption.

Compared to in-transit encryption, end-to-end encryption reduces the risks of information being compromised or corrupted. This is because it eliminates the need of trusting a service provider or operator with the data - the messages exchanged are encrypted between the devices themselves.

Chapter 3

Concept

We begin our work with a description of the building blocks of our analysis. To this end, we offer a high level overview of the concept of link preview generation. We continue by describing in more detail some of the steps involved in this process. We end the chapter by explaining the different test cases we consider for the experiment and by outlining the assumed attacker model.

3.1 Link Preview Generation Framework

The analysis discussed in our work is based on the concept of link preview generation. That is, when Client A sends a message containing an URL (e.g., https://thesisstefan.science.ru.nl/) to Client B using a messenger app, the user devices make one or more HTTP GET requests to the server hosting the website. The server returns the web page content and it records these requests in its logs. By fetching the special meta tags (e.g., OpenGraph meta tags) from the returned web page, the messenger app is capable of creating a link preview for it. This process is illustrated in Figure 3.1.

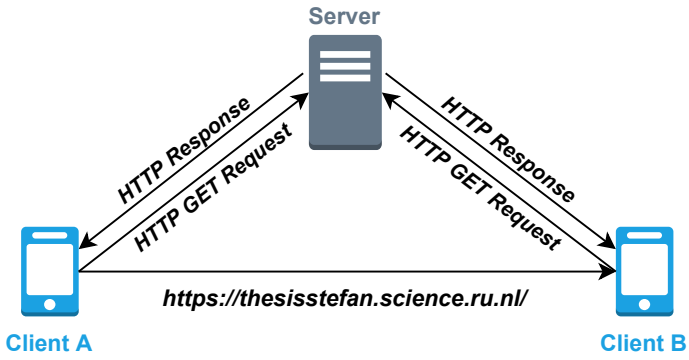

Figure 3.1: Overview of the Link Preview generation process

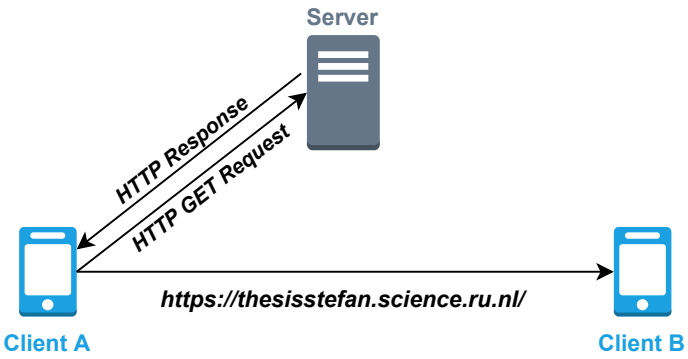

Figure 3.2: The sender's device generates the link preview request

Based on this high-level description, we can sum up the building blocks of our thesis:

- A small experimental setup - a remote web server hosting a simple website suitable for preview
- Attacker model - maintaining a server and having access to the logs
- Two or more Android devices - used for running the different messenger apps which receive a link with the preview content

3.2 Generation Methods & Server Logs

In Section 3.1 we mentioned that the user devices make one or more HTTP GET requests in order to generate a link preview. However, in practice, only one of the parties involved in the communication needs to make these requests. We can now distinguish between two options: (i) the messenger app creates HTTP GET requests from the sending device and (ii) the messenger app creates HTTP GET requests from the receiving device(s). This is illustrated in Figure 3.2 and Figure 3.3.

There is also a third approach, which is mostly implemented by the regular messengers. The app first sends the link to an external server which will generate the preview, and then the server returns this preview to both the sender and receiver. This is illustrated in Figure 3.4.

The requests made by the client devices are recorded on the server logs. Therefore, we can use these logs to find out more about the infrastructure of the messenger app that is being used, or about the users themselves. This includes, but is not limited to, the IP address of the sender and/or receiver, the app that is being used and the operating system of the user device(s).

Figure 3.3: The receiver's device generates the link preview request

3.3 Test Cases

Based on the different features and encryption methods implemented by each messenger app selected for our thesis, we distinguish the following test cases:

- *Regular vs. End-to-End Encryption Messengers*: In this case, regular messengers implement in-transit encryption, while the others implement end-to-end encryption. We distinguish between the two because there are different expectations for our experiment. For example, we do not expect end-to-end encrypted messengers to use an external server for generating the preview, since this goes against the idea of the operator not accessing and tampering with the messages that are being exchanged.

- *One-to-One vs. Group Chats*: It is important to test between both types of chats because the results will have different implications. For example, if a messenger app generates the link previews using the sending device, there would not be much of a difference between one-to-one and group chats if the IP address included in the request is the actual IP address of the device. However, if the messenger app uses the receiving device(s) for generating the preview, this means that an adversary would be able to record all the IP addresses of the users involved in the group communication. Moreover, past research has shown that it is possible for differences to exist between the security of one-to-one chats and the security of group chats [17].

- *Mobile vs. Desktop Apps*: For some messengers it is not quite clear how their desktop version is implemented. Therefore it is interesting to test the applications on both platforms and document the differences.

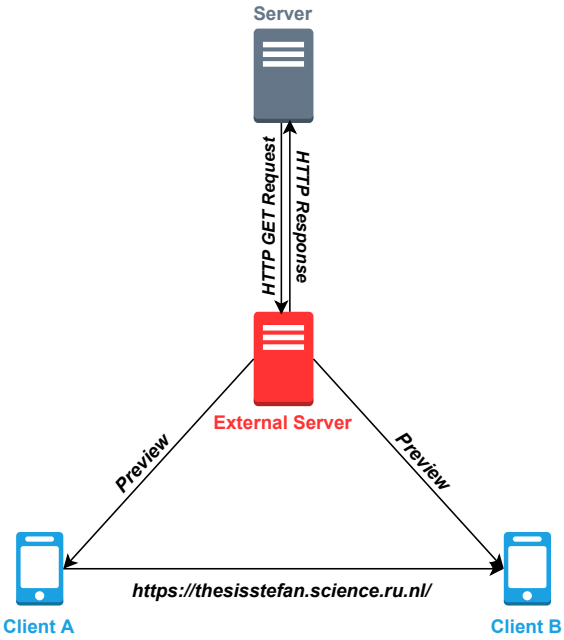

Figure 3.4: An external server generates the link preview request

3.4 Attacker Model

There are various ways in which different actors can access and use the information generated by the link previews. For example, an adversary could monitor all the traffic travelling through the app's infrastructure. If the messenger does not offer end-to-end encryption, this means that the adversary can also read the information that is being exchanged, and whom is communicating with whom. A concrete example is the operator of the app itself. However, even in this case, this means that the operator needs to constantly monitor its worldwide infrastructure, which is a costly and highly demanding process.

Another possibility is that the adversary is hosting a malicious website on a remote server and can monitor the requests made to this server. Therefore, by only making use of the logs, they are capable of parsing the requests made during link preview generation for any privacy-critical piece of information. The only infrastructure needed here is a remote server capable of hosting some content and thus it would not require a lot of resources and effort.

A final approach is that of the adversary impersonating a normal user. By itself, this adversary is not capable of performing any type of monitoring or attack. Therefore, it needs to be combined with one of the previous two attacker models. The goal here is to stimulate the receiving user(s) to further share the URL with their own contacts, and monitor the requests or the traffic created by those conversations.

Chapter 4

Experimental Setup

We continue our work by explaining the experimental setup in more detail. To this end, we first offer a description of the different devices involved. We continue by motivating our choice of messenger apps and outlining the experimental process. We conclude the chapter with a detailed overview of the different steps incorporated in our analysis.

4.1 Setup

For the analysis presented in this thesis, we prepare a set of controlled experiments, for which we conduct a number of trials on each of the selected messengers. The experiments consist of a user submitting a link to the web page we are hosting on a remote server, and the other user(s) opening the message containing the link and the link preview. Consequently, we register two user accounts for each mobile messenger app.

4.1.1 User Devices & Remote Server

We conduct our experiments using two different mobile devices, i.e., an LG G6 device (Model H870S, Android 8) and a Huawei Mate 20 Lite device (Model SNE-LX1, EMUI version 10.0.0, Android 10).

For testing the desktop versions of the selected apps, we use an ASUS TUF A15 laptop (Model FA506IV) with Windows 10 Pro (Version 20H2) installed.

For hosting our web page content, we set up a remote Apache web server, version 2.4.29, using the Radboud University infrastructure. The web page contains a page title, a logo, a navigation bar, and a table together with some images. Moreover, Open Graph meta tags were added to ensure the link preview generation. In our case, we use meta tags for page type, page

title, description, and preview image. Please check Appendix A for the HTML soruce code and Appendix B for the CSS source code.

4.1.2 Messenger Apps

We conduct our experiments on 16 mobile messenger apps. In this section we present the criteria we use for selecting these messengers.

We use an overview of the most secure messenger apps in our selection [21], which resulted in 17 possible candidates. In this case, we consider an application to be secure if it uses end-to-end encryption for the message exchange. From this batch, we eliminate 8 apps because of the following reasons: (i) low amount of downloads in the Google Play Store (<1M), (ii) no link preview implementation, (iii) requires subscription for full functionality and (iv) not available on Android.

Because we also want to test messenger apps that do not implement end-to-end encryption, or which do not implement end-to-end encryption by default, we select 11 candidate apps by going over the most popular apps - based on the number of downloads and the average review score - listed under the Communication category in the Google Play Store. From this list, we again have to remove 4 apps because of the following two reasons: (i) no link preview implementation and (ii) the application will soon be retired. Table 4.1 lists the 16 apps we use for our study.

4.2 Experimental Process

Here, we will go through the experimental process step by step. Firstly, there are different expectations for the two types of messengers and therefore we have divided the selected apps into End-to-End Encrypted and Regular messengers. We have then ordered them alphabetically.

Regarding our classification of the different apps, it is worth noting that some messengers not offering encryption by default are included in the End-to-End Encrypted category (e.g. Telegram) and some messengers in the Regular category also offer End-to-End Encryption (e.g. Facebook Messenger). This is because of how the applications are being marketed. For example, even though Telegram does not offer End-to-End Encryption by default, it is advertised as being an end-to-end encrypted messenger.

Afterwards, we run the experiments on the different messengers. This consists of one user device sending a message containing the link to the website hosted on our remote server, and the other user device(s) opening the message containing the URL and the preview of the website. We conduct two

Table 4.1: List of apps, including the number of downloads in the Play Store, the type of encryption implemented by the app and the build version

App	Downloads	Regular/E2E	Version
WhatsApp	5B+	○/●	2.20.205.16
Google Messages	1B+	○/●	7.0.039
Instagram	1B+	●/○	167.1.0.25.120
Facebook Messenger	1B+	●/●	291.2.0.22.114
Skype	1B+	●/●	8.66.0.76
Snapchat	1B+	●/○	11.7.0.62
imo	500M+	●/○	2020.11.1051(5058)
Line	500M+	●/●	10.20.1
Telegram	500M+	●/●	7.2.1(2139)
Viber	500M+	○/●	14.3.0.5
Discord	100M+	●/○	50.2
KakaoTalk	100M+	●/●	9.1.3
Signal	10M+	○/●	4.79.3
Slack	10M+	●/○	20.11.20.0
Wickr	5M+	○/●	5.66.8
Wire	1M+	○/●	3.60.960

trial runs for each messenger and for each of the test cases presented in Section 3.3. That is, we run the experiment twice for the end-to-end encryption implementation and the regular implementation in the case of single chats. We follow the same pattern for the group chats and the desktop apps. We direct two trials to (i) check for inconsistencies in the format or number of requests made and (ii) aid in the analysis of the results as explained in Section 4.3. A few times during the experiments we had to change the image shown in the preview and switch between *http://thesisstefan.science.ru.nl/* and *http://thesisstefan.science.ru.nl/index.html*, due to the fact that some messengers do not make a request each time the link is sent in a message.

The final step of the experiment is to parse the logs based on the division discussed at the beginning of this section, and then analyse the results following the steps we outline next.

4.3 Analysis Process Outline

In Chapter 2 we have explained the structure of the server logs and what important bits of information can be extracted from each part. Therefore, we use a step-by-step approach for the analysis of our results, with each step focusing on a different part of the logs and its implications for our experiments.

- *Step 1*: We look at the IP addresses used in the requests in order to conclude whether one of the following situations apply: (i) the request contains the IP address of the sender, (ii) the request contains the IP address of the receiver, (iii) a request which includes the real IP address is made for each user involved in the communication or (iv) the messenger app uses a proxy to hide the user's IP address. Moreover, in certain cases, we can use the IP address of the proxy server to distinguish between the different messenger apps.

- *Step 2*: We analyze the user agent included in the server logs for the purpose of searching for other sensitive information about one of the users participating in the communication. This includes, but is not limited to, phone model, app version and operating system version.

- *Step 3*: The next step was to look for any other piece of information we can gather from the server logs, such as the number of users receiving the link. In this step, we do not only use the information shown in the logs, but also the number of requests made, for example.

Chapter 5

Results

In this chapter, we describe the results obtained after running the experiment on each of the apps. There are different expectations for the two types of mobile messengers, and therefore the chapter is divided into two sections: (i) a section dedicated to presenting the results acquired from end-to-end encrypted messengers and (ii) a section dedicated to presenting the results acquired from regular messengers.

5.1 End-to-End Encrypted Messengers

The main building block of end-to-end encryption is that it prevents possible eavesdroppers from being able to decrypt and read the messages exchanged between two or more users. Moreover, it is also advertised to be the most secure type of encryption implemented. In the case of our analysis, we expect the information leaks to be reduced to a minimum for the messengers implementing this type of encryption. The results are as follows.

5.1.1 Google Messages

Messages is an app developed by Google specifically for their Android mobile operating system. It offers both the SMS and instant messaging functionalities, with the latter having to be manually activated on all the devices involved in the communication. Only recently Google has introduced end-to-end encryption for their instant messaging feature, in an effort to speed up the process of replacing SMS. Messages is available on both mobile and desktop devices, with the latter requiring the mobile counterpart to be connected to the internet.

```
//One-to-One & Group Chat - Mobile
66.249.93.94 thesisstefan.science.ru.nl - - [23/Nov/2020:18:46:11 +0100]
 ↪  "GET / HTTP/1.1" 200 9444 "-" "Mozilla/5.0 (X11; Linux x86_64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
 ↪  Google (+https://developers.google.com/+/web/snippet/)"
66.249.93.112 thesisstefan.science.ru.nl - - [23/Nov/2020:18:46:11 +0100]
 ↪  "GET /images/romania-counties.jpg HTTP/1.1" 200 421607 "-" "Mozilla/5.0
 ↪  (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
 ↪  Chrome/56.0.2924.87 Safari/537.36 Google
 ↪  (+https://developers.google.com/+/web/snippet/)"

//One-to-One & Group Chat - Desktop
66.249.93.49 thesisstefan.science.ru.nl - - [23/Nov/2020:21:40:21 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "Mozilla/5.0 (X11; Linux x86_64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
 ↪  Google (+https://developers.google.com/+/web/snippet/)"
66.249.93.51 thesisstefan.science.ru.nl - - [23/Nov/2020:21:40:26 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "Mozilla/5.0 (X11; Linux x86_64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
 ↪  Google (+https://developers.google.com/+/web/snippet/)"
```

Using the steps presented in the previous chapter, we make the following observations about Google Messages.

- *Step 1*: Google Messages is using an external server for HTTP requests made by the app when a user sends a message containing the link to the website. Therefore neither the sender's IP address nor the receiver's IP address has been forwarded to the server.
- *Step 2*: The fact that the message was sent within Google Messages can be observed using the user agent string.
- *Step 3*: We can distinguish between the desktop and mobile version based on the format of the requests. That is, the desktop app does not fetch any images from the page itself, while the mobile version does (*/images/romania-counties.jpg*).

  Another possibly important privacy-critical information is the number of recipients of the URL. Google Messages makes a request each time the link to the website is sent in a chat, and thus an attacker can have an idea about the number of people a user is communicating with.

5.1.2 Line

Line is a South-Korean messenger app and it is the main competitor of KakaoTalk. It is available on multiple platforms and it allows for the exchange of texts and media files and audio and video calls. Alongside the

messaging features, Line also offers an online wallet, taxi platform and online shopping, amongst others. The app implements both in-transit and end-to-end encryption.

```
//One-to-One & Group Chat - Mobile (Regular)
147.92.179.115 thesisstefan.science.ru.nl - - [27/Dec/2020:21:19:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.115 thesisstefan.science.ru.nl - - [27/Dec/2020:21:19:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.114 thesisstefan.science.ru.nl - - [27/Dec/2020:21:19:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.109 thesisstefan.science.ru.nl - - [27/Dec/2020:21:19:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"

//One-to-One & Group Chat - Mobile (E2E)
147.92.179.112 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.118 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.106 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.107 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.111 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"
147.92.179.111 thesisstefan.science.ru.nl - - [05/Dec/2020:12:55:36 +0100]
 ↪  "GET / HTTP/1.1" 200 8451 "-" "facebookexternalhit/1.1;line-poker/1.0"

//One-to-One & Group Chat - Desktop
Desktop version not working.
```

Line is the only messenger that offers both types of encryption **and** implements link previews for both types of encryption. It is also important to mention that while the app does have a desktop version, unfortunately we were not able to get it working on our device. The results for Line are as follows:

- *Step 1*: The app is using an external server for making the HTTP requests necessary for generating the link preview. Therefore, none of the devices involved in the communication has their IP address forwarded to the remote server. This is the case for both in-transit and end-to-end encryption.
- *Step 2*: The user agent of the requests can be used for concluding that the message was sent within a Line conversation.
- *Step 3*: In this case, we can distinguish whether the message was sent within a conversation that uses in-transit encryption or end-to-end encryption based on the number of requests made. That is, for the former four requests are made, while for the latter six requests are made.

5.1.3 Signal

Signal is a cross-platform instant messaging app developed by the Signal Foundation. The app allows for both one-on-one and group chats and calls, and it is available on mobile and desktop devices. Unlike most other messengers, it is not required for the mobile app to be connected to the internet when the desktop variant is used. Signal implements end-to-end encryption by default, using their own, open-source protocol. Other features include, but are not limited to, setting timers to messages and locking the app with a passphrase or biometric authentication.

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [18/Nov/2020:18:52:09 +0100]
 ↪  "GET / HTTP/1.1" 200 9231 "-" "WhatsApp"

//One-to-One & Group Chat - Desktop
Link previews implementation is limited.
```

Before discussing our observations, it is important to mention that Signal has a Desktop version with link previews, however it only generates them for a limited number of well-known websites. Therefore, the Desktop version of Signal has not been tested. The results obtained from the Signal app are along these lines.

- *Step 1*: Our analysis of Signal has shown that the app uses the IP address of the sender when making the request for generating the preview.
- *Step 2*: The user agent is unusual in the sense that it mentions WhatsApp. This might be the case because WhatsApp and Signal share the same encryption algorithm. However, it is still possible to conclude that the request was made by the Signal app because the user agent does not mention an app version, while for WhatsApp it is included.
- *Step 3*: Another possibly important privacy-critical information is the number of recipients of the URL. Signal makes a request each time the link to the website is sent in a chat, and thus an attacker can have an idea about the number of people a user is communicating with.

5.1.4 Telegram

Telegram is a messenger that was introduced in 2013 after the revelations brought forward by Edward Snowden. It is open source and cross-platform, and it allows for instant messaging, video calling and VoIP communication. Telegram implements in-transit encryption as well as end-to-end encryption in their **Secret Chat** functionality, using a custom protocol.

```
//One-to-One & Group Chat - Mobile
149.154.161.17 thesisstefan.science.ru.nl - - [18/Nov/2020:19:07:10 +0100]
 ↪  "GET / HTTP/1.1" 200 8441 "-" "TelegramBot (like TwitterBot)"

//One-to-One & Group Chat - Desktop
149.154.161.17 thesisstefan.science.ru.nl - - [18/Nov/2020:19:10:23 +0100]
 ↪  "GET / HTTP/1.1" 200 8441 "-" "TelegramBot (like TwitterBot)"
```

An important observation is the fact that the requests made by Telegram when generating a link preview are the same for both the in-transit and end-to-end encrypted chats. Therefore, we make no differentiation between the two here. We have decided to add it under this section since the messenger is mainly promoted for its end-to-end encryption implementation. The following observations have been made.

- *Step 1*: Telegram is using an external server for HTTP requests made by the app when a user sends a message containing the link to the website. Therefore the sender's IP address was not forwarded to the server.
- *Step 2*: Again, the user agent shows that the message was sent within a specific messenger, in this case Telegram.
- *Step 3*: During our analysis of Telegram, we have observed that regardless of the amount of time between the messages, the app does not make another request to the server for generating the preview. This is because the only way to have it fetch the content of the website again is by using the **@webpagebot**.

5.1.5 Viber

Viber is a Japanese instant messaging and VoIP application, available on multiple platforms such as Android, iOS and Microsoft Windows, amongst others. It offers end-to-end encryption by default since 2016, using a custom implementation of the Signal Protocol. The app uses phone numbers for user registration and identification, however the desktop version can be used without the need of mobile connectivity.

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [18/Nov/2020:19:20:16 +0100]
 ↪  "GET / HTTP/1.1" 200 9473 "-" "Viber"

//One-to-One & Group Chat - Desktop
92.110.122.85 thesisstefan.science.ru.nl - - [04/Dec/2020:15:28:01 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngine/5.15.2
 ↪  Chrome/83.0.4103.122 Safari/537.36"
92.110.122.85 thesisstefan.science.ru.nl - - [04/Dec/2020:15:28:01 +0100]
 ↪  "GET / HTTP/1.1" 200 9444 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngine/5.15.2
 ↪  Chrome/83.0.4103.122 Safari/537.36"
```

By following the steps outlined in Section 4.3, we can present various findings.

- *Step 1*: The experimental trials on Viber have shown that the app uses the IP address of the sender when making the request for generating the preview. This is the case for both the mobile and desktop versions.
- *Step 2*: In the case of the mobile version of the app, it is possible to conclude that the message was sent within Viber based on the user agent string. The user agent used by the desktop version does not offer any concrete information about the user.
- *Step 3*: Another piece of important information is the number of users receiving the message, since Viber makes a request each time a user sends a message containing the link. It is not possible to distinguish between the mobile and desktop apps, due to the user agent of the latter not containing any useful information about the user.

5.1.6 WhatsApp

Bought by Facebook in 2014, WhatsApp is a free, cross-platform instant messenger and VoIP service. With more than 5 billion downloads worldwide, it is the most popular and widely used messenger app. It offers text and voice messages, voice and video calls and sharing of media files. It is available on both mobile and desktop devices, with the latter requiring the mobile counterpart to be connected to the Internet. In 2016, WhatsApp announced that all devices running the app were using end-to-end encryption by default, which was implemented using a custom implementation of the Signal Protocol.

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [18/Nov/2020:18:57:26 +0100]
 ↪  "GET / HTTP/1.1" 200 9473 "-" "WhatsApp/2.20.205.16 A"

//One-to-One & Group Chat - Desktop
92.110.122.85 thesisstefan.science.ru.nl - - [24/Nov/2020:19:43:44 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "WhatsApp/2.20.205.16 A"
92.110.122.85 thesisstefan.science.ru.nl - - [24/Nov/2020:19:43:44 +0100]
 ↪  "GET / HTTP/1.1" 302 571 "-" "WhatsApp/2.20.205.16 A"
92.110.122.85 thesisstefan.science.ru.nl - - [24/Nov/2020:19:43:44 +0100]
 ↪  "GET / HTTP/1.1" 200 9475 "-" "WhatsApp/2.20.205.16 A"
```

After running the experiment on WhatsApp, we can outline the following findings based on the steps discussed in Section 4.3.

- *Step 1*: During our testing of WhatsApp we have observed that the app forwards the sender's IP address to the server, when making the request for generating the preview. This is the case for both the mobile and desktop versions of the app. Furthermore, the desktop variant forwards the IP address regardless of whether a VPN is used or not. This is because WhatsApp uses a mirroring functionality. In other words, messages sent from the desktop device will first be forwarded to the mobile device, and only then they will be transmitted to the intended recipient.
- *Step 2*: We can see that the HTTP request also includes the app version, in this case *2.20.205.16*, and whether it is sent from Android or iOS device, in this case *Android (A)*.
- *Step 3*: We can also differentiate whether the requests are made from the mobile version or the desktop version by using the numbers of requests made. In this case, the mobile app makes one request during link preview generation, while the desktop variant makes three.

5.1.7 Wickr

Wickr, or Wickr Me, is an American instant messaging app available on Android and iOS. Even though it is less known, the app is still a popular choice, with more than 5 million downloads on the Play Store alone. Wickr offers end-to-end encryption by default, and allows users to set an expiration time for their encrypted communications. Moreover, Wickr also claims to strip metadata from all the content transmitted through the network [10].

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [19/Nov/2020:16:03:13 +0100]
 ↪  "GET / HTTP/1.1" 200 9475 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
 ↪  10_11_1) AppleWebKit/601.2.4 (KHTML, like Gecko) Version/9.0.1
 ↪  Safari/601.2.4 facebookexternalhit/1.1 Facebot Twitterbot/1.0"

//One-to-One & Group Chat - Desktop
Link previews not implemented.
```

The Wickr messenger app does not implement link previews in the desktop version, and thus it has not been included in our analysis. Various findings are discussed here.

- *Step 1*: Wickr does not use a proxy and therefore the app forwards the IP address of the sender when making the request for generating the preview.
- *Step 2*: The user agent shows that the app uses the same bot as Facebook Messenger and Instagram. We are able to distinguish between the two based on the addition that the user agent of Wickr has, which is *"Facebot Twitterbot/1.0"*.
- *Step 3*: In the last step of our analysis, we have seen that Wickr generates a request to the server each time a message containing the link to the website is sent in the app. Therefore, the number of users receiving the message can be deduced.

5.1.8 Wire

Wire is a Swiss communication and collaboration app, available on multiple platforms such as Android, iOS and Microsoft Windows. It uses end-to-end encryption by default and the collaboration suite includes instant messaging, voice and video calling and file sharing. Wire comes in three different versions. In this thesis we have focused on Wire Personal, which is the secure messaging app dedicated to personal use.

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [19/Nov/2020:16:08:38 +0100]
 ↪  "GET / HTTP/1.1" 200 9475 "-" "Mozilla/5.0 (X11; Linux x86_64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36"

//One-to-One & Group Chat - Desktop
92.110.122.85 thesisstefan.science.ru.nl - - [23/Nov/2020:18:20:55 +0100]
 ↪  "GET / HTTP/1.1" 302 534 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100
 ↪  Safari/537.36"
92.110.122.85 thesisstefan.science.ru.nl - - [23/Nov/2020:18:20:55 +0100]
 ↪  "GET / HTTP/1.1" 302 534 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100
 ↪  Safari/537.36"
92.110.122.85 thesisstefan.science.ru.nl - - [23/Nov/2020:18:20:55 +0100]
 ↪  "GET / HTTP/1.1" 200 8662 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 ↪  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100
 ↪  Safari/537.36"
```

Using the step-by-step analysis describes in Chapter 4, the findings for the Wire messenger app are summarized below.

- *Step 1*: Our analysis of Wire has shown that the app uses the IP address of the sender when making the request for generating the preview.
- *Step 2*: No useful information about the user can be extracted from the user agent.
- *Step 3*: Wire acts in a similar way as other end-to-end encrypted messengers, in the sense that the app makes an HTTP request to the server each time a message which includes the link to the website is sent in a chat. However, in this case, the effects of this observation are negated by the fact that the user agent does not offer any useful information about the user.

5.1.9 Summary End-to-End Encryption Messengers

Our findings show that most end-to-end encrypted messengers use the sender's IP address in the HTTP GET requests made for generating link previews. The exceptions are **Google Messages**, **Line** and **Telegram**, which use external servers for this. It is also possible to (i) pinpoint within which messenger app was the URL shared, based on the user agent string, (ii) estimate the number of receiving users based on the number of requests and (iii) distinguish between the mobile and desktop implementations of certain apps. Another interesting case is that of **WhatsApp**, which also gives information about the app version and the operating system of the device.

5.2 Regular Messengers

Regular messengers do not use end-to-end encryption and therefore the messages that are sent within the app are first forwarded to the operator's servers and only afterwards relayed to the intended recipient. This means that one expectation for the apps classified under this category is the use of external or proxy servers for the requests made for generating link previews. Moreover, we also expect to be able to collect a larger amount of sensitive user information. The results are as follows.

5.2.1 Discord

Discord is a VoIP and instant messaging software application which is designed for creating communities. It is available on multiple platforms and it allows its users to communicate via text messages and voice and video calls. Discord started as a gaming communication platform where users can create servers or groups for various topics. In this analysis we have focused on the one-to-one and group chats only.

```
//One-to-One & Group Chat - Mobile
35.237.4.214 thesisstefan.science.ru.nl - - [20/Nov/2020:12:42:54 +0100]
 ↪  "GET / HTTP/1.1" 200 9418 "-" "Mozilla/5.0 (compatible; Discordbot/2.0;
 ↪  +https://discordapp.com)"

//One-to-One & Group Chat - Desktop
35.227.62.178 thesisstefan.science.ru.nl - - [24/Nov/2020:19:26:29 +0100]
 ↪  "GET / HTTP/1.1" 302 515 "-" "Mozilla/5.0 (compatible; Discordbot/2.0;
 ↪  +https://discordapp.com)"
35.227.62.178 thesisstefan.science.ru.nl - - [24/Nov/2020:19:26:29 +0100]
 ↪  "GET / HTTP/1.1" 200 9418 "http://thesisstefan.science.ru.nl/"
 ↪  "Mozilla/5.0 (compatible; Discordbot/2.0; +https://discordapp.com)"
```

The conclusions drawn from the step-by-step analysis are summarized in the next few paragraphs.

- *Step 1*: The requests made by Discord when a user sends a message containing the URL of our website are forwarded to an external server and therefore the sender's IP address is not included.
- *Step 2*: Here we are again able to deduce that the message was sent within Discord using the user agent string. This applies for both the mobile and desktop versions.
- *Step 3*: We can use the number of requests made to the server to distinguish between the two versions of the app. Discord on mobile devices will only make one request, while Discord on desktop will make two requests.

5.2.2 Instagram & Facebook Messenger

Facebook Messenger is the instant messaging app developed by Facebook Inc. Launching in 2008 as Facebook Chat, the app now has more than 1 billion downloads on the Play Store. It is available on both mobile and desktop devices, and it offers a large amount of features such as instant messaging, voice and video calls, money transfer and third-party integration. In October 2016 end-to-end encryption was introduced to Facebook Messenger using the Signal Protocol [29].

Instagram is an American social networking service owned by Facebook Inc, which is available on multiple operating systems, such as Android and iOS. While it is mainly designed as a social media platform, it also includes a chat feature, which allows for one-to-one or group communication.

```
//One-to-One & Group Chat - Mobile
31.13.115.24 thesisstefan.science.ru.nl - - [21/Nov/2020:20:18:46 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.115.10 thesisstefan.science.ru.nl - - [21/Nov/2020:20:18:46 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"

//One-to-One & Group Chat - Desktop
173.252.87.28 thesisstefan.science.ru.nl - - [24/Nov/2020:19:38:39 +0100]
 ↪  "GET / HTTP/1.1" 302 534 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
173.252.87.28 thesisstefan.science.ru.nl - - [24/Nov/2020:19:38:40 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
```

```
//One-to-One & Group Chat - Mobile
31.13.103.20 thesisstefan.science.ru.nl - - [20/Nov/2020:12:15:11 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.103.14 thesisstefan.science.ru.nl - - [20/Nov/2020:12:15:12 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"

//One-to-One & Group Chat - Desktop
31.13.103.14 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:43 +0100]
 ↪  "GET / HTTP/1.1" 302 534 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.103.14 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:43 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.103.23 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:43 +0100]
 ↪  "GET / HTTP/1.1" 302 534 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
```

```
31.13.103.23 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:43 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.103.5 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:45 +0100] "GET
 ↪  / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
31.13.103.112 thesisstefan.science.ru.nl - - [24/Nov/2020:19:30:46 +0100]
 ↪  "GET / HTTP/1.1" 206 8494 "-" "facebookexternalhit/1.1
 ↪  (+http://www.facebook.com/externalhit_uatext.php)"
```

It is important to note that while Facebook Messenger does have a **Secret Chat** functionality which uses end-to-end encryption, it does not implement link previews. The following conclusions were drawn after multiple experimental trials were ran on both apps.

- *Step 1*: Both Instagram and Facebook Messenger are using a proxy for HTTP requests made by the app when a user send a message containing the link to the website. Therefore the sender's IP address was not forwarded to the server. However, our analysis has shown that Facebook Messenger and Instagram use the same servers for generating the link previews, and it is possible to differentiate between the two based on the addresses used. For example, in our testing, Instagram has been using *31.13.115.0* and *173.252.87.0* addresses, while Facebook Messenger has been using *31.13.103.0* addresses.
- *Step 2*: Instagram and Facebook Messenger have the same user agent, and thus no concrete information can be extracted.
- *Step 3*: We can also distinguish between the mobile and desktop versions. In the case of Instagram, the mobile app uses *31.13.115.0* addresses while the desktop app uses *173.252.87.0* addresses. In the case of Facebook Messenger, the desktop version generates more requests for the preview.

5.2.3 imo

imo is a cross-platform instant messaging app owned by the Chinese company YY Inc. With more than 210 million monthly active users worldwide [6], the app shares a lot of similarities with WeChat. It offers audio and video calling, text messaging, stories and file sharing.

```
//One-to-One & Group Chat - Mobile
92.110.122.85 thesisstefan.science.ru.nl - - [27/Nov/2020:15:45:42 +0100]
 ↪  "GET / HTTP/1.1" 200 9472 "-" "Dalvik/2.1.0 (Linux; U; Android 10;
 ↪  SNE-LX1 Build/HUAWEISNE-L21)"
92.110.122.85 thesisstefan.science.ru.nl - - [27/Nov/2020:15:45:42 +0100]
 ↪  "GET / HTTP/1.1" 200 3953 "-" "Mozilla"
92.110.122.85 thesisstefan.science.ru.nl - - [27/Nov/2020:15:46:33 +0100]
 ↪  "GET / HTTP/1.1" 200 9472 "-" "Dalvik/2.1.0 (Linux; U; Android 8.0.0;
 ↪  LG-H870S Build/OPR1.170623.032)"
92.110.122.85 thesisstefan.science.ru.nl - - [27/Nov/2020:15:46:34 +0100]
 ↪  "GET / HTTP/1.1" 200 9472 "-" "Mozilla"
31.187.215.182 thesisstefan.science.ru.nl - - [27/Nov/2020:15:58:58 +0100]
 ↪  "GET / HTTP/1.1" 200 9472 "-" "Dalvik/2.1.0 (Linux; U; Android 10;
 ↪  SM-G960F Build/QP1A.190711.020)"
31.187.215.182 thesisstefan.science.ru.nl - - [27/Nov/2020:15:58:59 +0100]
 ↪  "GET / HTTP/1.1" 200 3953 "-" "Mozilla"

//One-to-One & Group Chat - Desktop
Desktop version not working.
```

An important observation here is that imo does have a Desktop version. Unfortunately, we were not able to get it working during our experiment. The results of our analysis of the app can be summarized as follows.

- *Step 1*: The results for imo are quite interesting. A request for generating the preview is made on the sender side as well as on the receiver(s) side. Because the app does not use an external server, this means that the IP addresses of all the users involved in the communication are included in the requests.
- *Step 2*: Information about the user devices is included in the user agents, such as the operating system version (*Android 10*), operating system build (*Build/OPR1.170623.032*) and phone model (*LG-H870S*).
- *Step 3*: The app does not make a request each time the URL is sent in a chat, however we can still learn about the number of users receiving it due to imo making a request when a user opens the message.

5.2.4 KakaoTalk

KakaoTalk is a South Korean instant messaging app. Starting as purely a messenger service, it has now become a platform where also a large amount of third-party content and apps can be accessed and downloaded. It is available on mobile as well as on desktop devices, and it allows users to communicate via text and voice messaging, initiate voice and video calls and also share media files, location and URL links, amongst others.

```
//One-to-One & Group Chat - Mobile
211.249.201.238 thesisstefan.science.ru.nl - - [19/Nov/2020:14:09:13 +0100]
 ↪  "GET / HTTP/1.1" 200 9172 "-" "facebookexternalhit/1.1;
 ↪  kakaotalk-scrap/1.0; +https://devtalk.kakao.com/t/scrap/33984"

//One-to-One & Group Chat - Desktop
121.53.84.4 thesisstefan.science.ru.nl - - [23/Nov/2020:21:52:56 +0100] "GET
 ↪  / HTTP/1.1" 302 572 "-" "facebookexternalhit/1.1; kakaotalk-scrap/1.0;
 ↪  +https://devtalk.kakao.com/t/scrap/33984"
211.249.204.140 thesisstefan.science.ru.nl - - [23/Nov/2020:21:52:58 +0100]
 ↪  "GET / HTTP/1.1" 200 9172 "http://thesisstefan.science.ru.nl/"
 ↪  "facebookexternalhit/1.1; kakaotalk-scrap/1.0;
 ↪  +https://devtalk.kakao.com/t/scrap/33984"
```

Together with the regular chats, KakaoTalk also offers end-to-end encryption in their **Secret Chat** functionality. Due to link previews not being implemented in these chats, they were not included in the analysis. The following conclusions can be made.

- *Step 1*: From the IP address used in the request made to the server we can safely deduce that KakaoTalk is using a proxy for the HTTP requests made by the app when a user sends a message containing the link to the website. In this way the sender's IP address is not revealed.
- *Step 2*: The user agent string offers enough information to allow an attacker to identify the app that has made the request, in this case KakaoTalk.
- *Step 3*: The app does not make a request each time the URL of the website is used in a message, therefore we cannot estimate the number of users receiving it. However we can still distinguish between the mobile and desktop version. The mobile app only makes one request to the server when generating the link preview, while the the desktop app makes two.

5.2.5 Skype

Skype is a proprietary application which specializes in cross-platform voice and video calling. The app also offers instant massaging, allowing the users to send texts and media files. Users of the app are identified with a Skype username in most features, however the app uses another identifier - a Skype online number - to allow the receiving of conventional calls on computer devices. Skype has more than 1 billion downloads on the Play Store alone.

```
//One-to-One & Group Chat - Mobile
52.114.77.26 thesisstefan.science.ru.nl - - [20/Nov/2020:12:53:43 +0100]
 ↪  "GET / HTTP/1.1" 200 8700 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64)
 ↪  SkypeUriPreview Preview/0.5"
52.114.77.26 thesisstefan.science.ru.nl - - [20/Nov/2020:12:53:43 +0100]
 ↪  "GET /favicon.ico HTTP/1.1" 404 479 "-" "Mozilla/5.0 (Windows NT 6.1;
 ↪  WOW64) SkypeUriPreview Preview/0.5"

//One-to-One & Group Chat - Desktop
52.114.75.71 thesisstefan.science.ru.nl - - [24/Nov/2020:20:57:57 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64)
 ↪  SkypeUriPreview Preview/0.5"
52.114.75.71 thesisstefan.science.ru.nl - - [24/Nov/2020:20:57:57 +0100]
 ↪  "GET / HTTP/1.1" 200 8700 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64)
 ↪  SkypeUriPreview Preview/0.5"
52.114.75.71 thesisstefan.science.ru.nl - - [24/Nov/2020:20:57:57 +0100]
 ↪  "GET /favicon.ico HTTP/1.1" 404 479 "-" "Mozilla/5.0 (Windows NT 6.1;
 ↪  WOW64) SkypeUriPreview Preview/0.5"
```

Alongside the regular chat, Skype has end-to-end encrypted communication in the form of **Private Conversations**. They do not implement link previews and thus they were not analyzed. The conclusions that can be drawn after running the experiment on the Skype app are outlined below.

- *Step 1*: Skype is using a proxy for HTTP requests made by the app when a user sends a message containing the link to the website. Therefore the sender's IP address was not forwarded to the server.
- *Step 2*: In this case, we again notice that based on the user agent string (*"SkypeUriPreview Preview/0.5"*) we can conclude that indeed the request has been made by the Skype app.
- *Step 3*: The desktop and mobile versions can be distinguished based on the number of requests - Skype on mobile generates two requests, and on desktop generates three requests.

5.2.6 Slack

Slack is a business communication platform which is proprietary to the American software company Slack technologies. The app is known for offering multiple IRC-style features, such as persistent chat rooms and direct and group messaging. It only offers in-transit encryption, and it is available on mobile and dekstop devices.

```
//One-to-One & Group Chat - Mobile
54.235.57.223 thesisstefan.science.ru.nl - - [21/Nov/2020:20:21:51 +0100]
 ↪  "GET / HTTP/1.1" 206 8490 "-" "Slackbot-LinkExpanding 1.0
 ↪  (+https://api.slack.com/robots)"
```

```
54.235.57.223 thesisstefan.science.ru.nl - - [21/Nov/2020:20:21:52 +0100]
 ↪  "GET /favicon.ico HTTP/1.1" 404 5179 "-" "Slackbot 1.0
 ↪  (+https://api.slack.com/robots)"

//One-to-One & Group Chat - Desktop
54.164.7.236 thesisstefan.science.ru.nl - - [05/Dec/2020:12:43:31 +0100]
 ↪  "GET / HTTP/1.1" 302 572 "-" "Slackbot-LinkExpanding 1.0
 ↪  (+https://api.slack.com/robots)"
54.164.7.236 thesisstefan.science.ru.nl - - [05/Dec/2020:12:43:31 +0100]
 ↪  "GET / HTTP/1.1" 206 8487 "-" "Slackbot-LinkExpanding 1.0
 ↪  (+https://api.slack.com/robots)"
54.164.7.236 thesisstefan.science.ru.nl - - [05/Dec/2020:12:43:32 +0100]
 ↪  "GET /favicon.ico HTTP/1.1" 404 5179 "-" "Slackbot 1.0
 ↪  (+https://api.slack.com/robots)"
```

In the next few paragraphs we will present the results obtained from running the experiment on the Slack mobile and desktop apps.

- *Step 1*: The IP address included in the request received by the remote server can be traced back to an external server owned by KakaoTalk. Therefore, neither the IP address of the sender nor the IP address of receiver are exposed.
- *Step 2*: From the user agent used by the app, an attacker can deduce that the URL to the website has been sent within KakaoTalk.
- *Step 3*: We can distinguish between the mobile and desktop apps by looking at then number of requests made to the server. The desktop version makes three requests for generating the link preview, while the mobile version only makes one. Requests are not made multiple times and therefore a conclusion regarding the number of receiving users cannot be drawn.

5.2.7 Snapchat

Snapchat is an American messaging service which has as central feature the fact that messages sent within the app are only available to the intended recipients for a short amount of time before disappearing. Other features include the sharing of stories, media files and the editing of pictures, just to name a few. It is only available on mobile devices, and it one of the more popular instant messaging apps, with over 1 billion downloads on the Play Store alone.

```
//One-to-One & Group Chat - Mobile
107.178.194.122 thesisstefan.science.ru.nl - - [20/Nov/2020:12:25:16 +0100]
 ↪  "GET / HTTP/1.1" 200 9444 "-" "AppEngine-Google;
 ↪  (+http://code.google.com/appengine; appid: s~snapchat-proxy)"
```

```
//One-to-One & Group Chat - Desktop
Desktop version not available.
```

Following the analysis outline in Section 4.3, the results of the experiment are laid out below.

- *Step 1*: Snapchat is using a proxy for HTTP requests made by the app when a user send a message containing the link to the website. Therefore the sender's IP address was not forwarded to the server.
- *Step 2*: The only piece of information we can gather from the server logs, by looking at the user agent, is that indeed the request originates from the Snapchat app - *"appid: s~snapchat-proxy)"*.
- *Step 3*: In the case of Snapchat, no piece of information could be gathered in this step. There is no desktop version, and the application does not make subsequent requests when the URL of the website is sent multiple times.

5.2.8 Summary Regular Messengers

Our findings show that most regular messengers use an external server for the HTTP GET requests made for generating link previews. It is also possible to pinpoint within which messenger app was the URL shared, based on the user agent string and to distinguish between the mobile and desktop implementations of certain apps. One app that stands out is **imo**, which not only makes a request for all the users involved in the communication, but also includes the IP address of the respective device in this request.

5.3 Summary

The analysis of information leaks we conduct on the 16 messenger apps unveils that there are similarities, such as how the user agent string can be used to pinpoint the originating app of the request, as well as differences, such as the IP addresses used in these requests, between regular and end-to-end encrypted messenger. A few apps also offer interesting results, such as **Google Messenger**, **Line** and **Telegram**, which are end-to-end encrypted messengers using external servers for the requests, or **imo**, which exposes extensive information about the users involved in the conversation.

Chapter 6

Discussion

We conclude our work by discussing the important implications of the results outlined following our analysis. Consequently, we present two possible larger attacks in which our results can be used. We end the section with a brief description of the limitations considered throughout the thesis.

6.1 Findings

This section is dedicated to the implications that arise from using only the information we gathered from our experiment. We first discuss why knowing the IP address of a device is an important "information leak" and then we describe how the user agent string and number of requests can be used to create a clearer image about the communication that is being monitored. We finish our discussion with a small overview of the other findings of the analysis.

6.1.1 Originating Device & IP Addresses

Based on the timing of the HTTP GET requests and the IP addressed used, we can draw a few conclusions. Firstly, it is clear that for the majority of messengers the sending device is the one initiating the request. This is a good approach, because this means that the receiver is protected from any possible malicious link. The only exception in this regard is **imo**, which makes a request for all the devices involved in the communication - that is, both the sending device and the receiving device(s) participate in the link preview generation.

Building upon the previous point, we can conclude that most end-to-end encrypted messengers forward the IP address of the sender to our remote server (with the exception of **Google Messenger**, **Line** and **Telegram**), while regular messengers use external servers or proxy servers for the requests

made for generating link previews (with the exception of **imo**). Therefore, in the case of end-to-end encrypted messengers, it is possible to find the location of the sender using IP Geolocation. This technique allows for localization at the region or city level. **imo** is an unfortunate example, because not only it uses the actual IP address of the device, it does so for both the sending and receiving users.

Google Messages, **Line** and **Telegram** also stand out as end-to-end encrypted messengers, because they use an external server for generating the preview of the website. This defeats the purpose of end-to-end encryption, since the service provider is able to see what website is being accessed and which users are communicating about it.

6.1.2 Messenger Fingerprinting

Our results show that the user agent can offer quite a lot of information about the user device making the request. With a few exceptions, the user agent always specifies the originating messenger app. In the case of **WhatsApp** the user agent also specifies the app version and the operating system used (*Android* or *iOS*). **imo** is again the 'black sheep' of mobile messengers, since it sends out information about the operating system version, operating system build and the phone model. An adversary could use this information to exploit possible vulnerabilities of the specific messenger app involved, or even of the operating system of the device (in the case of **imo**).

6.1.3 Conversation Sketch

Based on our analysis we can also conclude that for some messengers it is possible to sketch out the number of people that receive the message containing the link. By itself this does not say much about an user, however it plays a more important role in a larger context, which will be explained in Section 6.2.

imo stands out here as well due to the previously mentioned information leaks. Since the app makes a request for each device opening the message, that means that an adversary can sketch out how many people are communicating and whom is communicating based on the number of requests and the IP addresses respectively.

6.1.4 In-Transit vs. End-to-End & Mobile vs. Desktop

There are a few other interesting findings resulting from our experiment. Firstly, in the case of **Line**, it is possible to distinguish between the two types of encryption based on the number of requests made. Moreover, for some of the other messengers, it is also possible to distinguish between the desktop and mobile versions.

6.2 Larger Context

In the previous section we have discussed about the potential implications of our experiment when only looking at the logs themselves. However, some of the findings can also play a role in a larger context.

For example, when the receiving user opens the link included in the message, some of the apps, such as Line or Facebook Messenger, will include in the request that the link is being opened from within the app. Since the user agent used in the HTTP GET request for generating the link preview mentions from within which app was the message sent (as explained in Section 6.1.2), an adversary is capable of making connections between sending and receiving users.

```
//Facebook Messenger
92.110.122.85 thesisstefan.science.ru.nl - - [03/Jan/2021:20:49:01 +0100]
 ↪  "GET / HTTP/1.1" 200 9475 "http://m.facebook.com/" "Mozilla/5.0 (Linux;
 ↪  Android 10; SNE-LX1 Build/HUAWEISNE-L21; wv) AppleWebKit/537.36 (KHTML,
 ↪  like Gecko) Version/4.0 Chrome/87.0.4280.101 Mobile Safari/537.36
 ↪  [FB_IAB/Orca-Android;FBAV/294.0.0.24.129;]"

//Line
92.110.122.85 thesisstefan.science.ru.nl - - [05/Dec/2020:13:00:20 +0100]
 ↪  "GET / HTTP/1.1" 200 9475 "-" "Mozilla/5.0 (Linux; Android 10; SNE-LX1
 ↪  Build/HUAWEISNE-L21; wv) AppleWebKit/537.36 (KHTML, like Gecko)
 ↪  Version/4.0 Chrome/86.0.4240.185 Mobile Safari/537.36 Line/10.20.1/IAB"
```

Another example is that of WhatsApp. The app implements a feature which displays the activity of an user. That is, it shows when the user was last active or if the user is currently online. The app allows for the former to be deactivated, however it will still display information about the current status.

WhatsApp also allows for an enumeration attack, which means that an adversary can find out which numbers have an WhatsApp account and which numbers do not by just adding them in the contact list of a mobile device. This is because the app automatically scans this list for potential WhatsApp contacts [18].

Based on this information, a larger attack can be theoretically outlined as follows:

- An adversary can monitor a server and the server logs for any WhatsApp requests made for the link preview generation using the user agent string, as explained in Section 6.1.2;
- When such a request is being recorded, based on the location given by the sender's IP address (Section 6.1.1), the adversary can create a script for generating numbers for a certain country and/or region and thus implementing an enumeration attack [18];
- The numbers that match with active users can be monitored using software such as WhatsSpy [18]. This means that the adversary can now monitor the WhatsApp activity of these users;
- The final step is to match requests made when the website is being accessed with the activity times of the monitored users. The IP addresses can aid in this since it will tell the adversary whether the user is from the country of interest or not.

This type of attack can theoretically be used against any messenger app implementing similar features as WhatsApp, however it might be harder to conduct without an existing monitoring software.

6.3 Limitations

It is important to mention that for our experiment we limited ourselves to only using the information we can gather from the server logs themselves. However, there are many other possibilities when it comes to studying link previews. For example, an option would be to check the HTTP GET request in more detail and see if any other information can be extracted from the headers or the message itself. Moreover, there exists some work about the possible vulnerabilities of link previews, such as being able to drain the battery of the device or succeeding in running malicious code on the external/proxy servers that the messenger apps are using for generating the previews [7].

Another limitation is related to the messenger applications chosen for the analysis. This is because it was not possible to test every single app that exists due to time constraints. Moreover, link previews is not a feature unique to messenger applications. It is implemented in social networks, mail clients, video conference software etc. Therefore, there are a lot more software applications that can be analysed in a similar manner.

Chapter 7

Conclusion

In this thesis, we offered a detailed analysis of link previews in mobile messaging platforms. To this end, our concept considers the requests made by an app when the link preview is being generated. In an experimental study, we have set up a remote server hosting a small website with content suitable for preview, and we have analyzed 16 mobile messengers by documenting what information can be extracted from the respective requests. Our findings have shown that it is, in theory, possible for an adversary that only has access to the server logs to localize the sender, find out the originating app of the request, estimate the number of users receiving the link or distinguish between the different versions of a messenger. One platform also exposes the IP addresses of the receiving users, or information about the user device such as the operating system version and the phone model. We concluded our work with a discussion about the different implications our findings have in a larger context.

Bibliography

[1] "Apache Log Files". https://httpd.apache.org/docs/2.4/logs.html.

[2] "HTTP Protocol". https://tools.ietf.org/html/rfc2616.

[3] "Same Origin Policy". https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy.

[4] "The Open Graph protocol". https://ogp.me/.

[5] "The Twitter Cards protocol". https://developer.twitter.com/en/docs/twitter-for-websites/cards/overview/abouts-cards.

[6] "YY Reports First Quarter 2019 Unaudited Financial Results". https://ir.yy.com/news-releases/news-release-details/yy-reports-first-quarter-2019-unaudited-financial-results.

[7] Mysk Bakry. "Link Previews: How a Simple Feature Can Have Privacy and Security Risks", Oct 2020. https://www.mysk.blog/2020/10/25/link-previews/.

[8] Hitesh Ballani, Paul Francis, and Xinyang Zhang. "A Study of Prefix Hijacking and Interception in the Internet". *ACM SIGCOMM Computer Communication Review - SIGCOMM '07*, 37(4):265–276, August 2007.

[9] Massimo Candela. "Current Status of IP Geolocation", August 2020. https://www.lacnic.net/innovaportal/file/4695/1/ip_geolocation_status.pdf.

[10] CNBC. "Snapchat rival hopes to pounce on security breach". https://www.cnbc.com/2014/02/06/snapchat-rival-hopes-to-pounce-on-security-breach.html.

[11] CNBC. "WhatsApp confirms it's been targeted by spyware. That's exactly its customers' fear". https://www.cnbc.com/2019/05/14/whatsapp-confirms-its-been-targeted-by-spyware.html.

[12] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. "A Formal Security Analysis of the Signal Messaging Protocol". *IEEE European Symposium on Security and Privacy - EuroS&P '17*, pages 451–466, April 2017.

[13] Amir Herzberg and Hemi Leibowitz. "Can Johnny Finally Encrypt? Evaluating E2E-Encryption in Popular IM Applications". *Proceedings of the 6th Workshop on Socio-Technical Aspects in Security and Trust - STAST '16*, page 17–28, December 2016.

[14] Katharina Kohls, Kai Jansen, David Rupprecht, Thorsten Holz, and Christina Popper. "On the Challenges of Geographical Avoidance for Tor". *Network and Distributed System Security Symposium - NDSS '19*, February 2019.

[15] Security Magazine. "Facebook fixes Messenger bug that allowed Android users to spy on each other". `https://www.securitymagazine.com/articles/94000-facebook-messenger-bug-allowed-android-users-to-spy-on-each-other`.

[16] Giancarlo Pellegrino, Onur Catakoglu, Davide Balzarotti, and Christian Rossow. "Uses and Abuses of Server-Side Requests". *International Symposium on Research in Attacks, Intrusions, and Defenses - RAID '16*, pages 393–414, September 2016.

[17] Paul Rösler, Christian Mainka, and Jörg Schwenk. "More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema". *IEEE European Symposium on Security and Privacy - EuroS&P '18*, 2018.

[18] Christoph Rottermanner, Peter Kieseberg, Markus Huber, Martin Schmiedecker, and Sebastian Schrittwieser. "Privacy and Data Protection in Smartphone Messengers". *Proceedings of the 17th International Conference on Information Integration and Web-Based Applications & Services - iiWAS '15*, pages 1–10, December 2015.

[19] Sebastian Schrittwieser, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, and Markus Huber. "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications". *Network and Distributed System Security Symposium - NDSS '12*, February 2012.

[20] Svenja SchroDer, Markus Huber, David Wind, and Christoph Rottermanner. "When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging". *Network and Distributed System Security Symposium - NDSS'16*, August 2016.

[21] Heimdal Security. "The Best Encrypted Messaging Apps You Should Use Today", 2019. https://heimdalsecurity.com/blog/the-best-encrypted-messaging-apps/.

[22] Justin Seitz. "How To Blow Your Online Cover With URL Previews", Jan 2019. https://www.bellingcat.com/resources/how-tos/2019/01/04/how-to-blow-your-online-cover-with-url-previews/.

[23] Statista. "Number of mobile phone users worldwide from 2015 to 2020". https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/.

[24] Giada Stivala and Giancarlo Pellegrino. "Deceptive Previews: A Study of the Link Preview Trustworthiness in Social Platforms". *Network and Distributed System Security Symposium - NDSS '20*, February 2020.

[25] Tomáš Sušánka and Josef Kokeš. "Security Analysis of the Telegram IM". *Proceedings of the 1st Reversing and Offensive-Oriented Trends Symposium - ROOTS '17*, November 2017.

[26] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. "SoK: Secure Messaging". *IEEE Symposium on Security and Privacy - EuroS&P '15*, pages 232–249, July 2015.

[27] Elham Vaziripour, Justin Wu, Mark O'Neill, Jordan Whitehead, Scott Heidbrink, Kent Seamons, and Daniel Zappala. "Is that you, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications". *Thirteenth Symposium on Usable Privacy and Security - SOUPS '17*, pages 29–47, July 2017.

[28] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. "How to Catch When Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation". *Proceedings of the Internet Measurement Conference - IMC '18*, page 203–217, October 2018.

[29] Wired. "You Can All Finally Encrypt Facebook Messenger, So Do It". https://www.wired.com/2016/10/facebook-completely-encrypted-messenger-update-now/.

[30] Digital Information World. "Statistics Reveal Facebook's Social Media Monopoly". https://www.digitalinformationworld.com/2019/10/monthly-active-users-of-top-social-media-platforms-and-messaging-apps.html.

Appendix A

HTML Source Code

```
  <!DOCTYPE html>

<html lang = "en">
  <head>
    <title>Romania - Regions and Counties</title>
      <meta property="og:type" content="website">
      <meta property="og:title" content="Romania - Regions and Counties">
      <meta property="og:description" content="A website about Romania">
      <meta property="og:image"
       ↪  content="https://cdn.webshopapp.com/shops/94414/files/52415442/
                  flag-of-romania.jpg">
      <meta name = "viewport"           content = "width=device-width">
      <link rel = "stylesheet" href="styles.css">
  </head>

  <body>
    <header>
      <h1><img src = "images/romania-logo.jpg" width="600" height="200"
       ↪  alt=""></h1>
    </header>

    <nav class = "navbar">
      <p><a href = "index.html">Home</a></p>
          <p><a href = "sources.html">Sources</a></p>
    </nav>

    <article>
      <div id = "main">
            <figure class = "ro-map">
              <img src = "images/romania-counties.jpg" width = "586" height
               ↪  = "398" alt = "">
                </figure>

        <table style = "width:40%">
          <tr>
            <th>Region</th>
            <th>Counties</th>
```

```
</tr>

<tr>
  <td rowspan = "2">Bucovina</td>
                <td>Botosani</td>
        </tr>
        <tr>
                <td>Suceava</td>
        </tr>

        <tr>
  <td rowspan = "6">Moldova</td>
                <td>Iasi</td>
        </tr>
        <tr>
                <td>Neamt</td>
        </tr>
        <tr>
                <td>Bacau</td>
        </tr>
        <tr>
                <td>Vaslui</td>
        </tr>
        <tr>
                <td>Vrancea</td>
        </tr>
        <tr>
                <td>Galati</td>
        </tr>

        <tr>
  <td rowspan = "2">Dobrogea</td>
                <td>Tulcea</td>
        </tr>
        <tr>
                <td>Constanta</td>
        </tr>

        <tr>
                <td rowspan = "10">Muntenia</td>
                <td>Braila</td>
        </tr>
        <tr>
                <td>Ialomita</td>
        </tr>
        <tr>
                <td>Calarasi</td>
        </tr>
        <tr>
                <td>Buzau</td>
        </tr>
        <tr>
                <td>Prahova</td>
        </tr>
```

```
<tr>
      <td>Ilfov</td>
</tr>
<tr>
      <td>Giurgiu</td>
</tr>
<tr>
      <td>Dambovita</td>
</tr>
<tr>
      <td>Arges</td>
</tr>
<tr>
      <td>Teleorman</td>
</tr>

<tr>
      <td rowspan = "5">Oltenia</td>
      <td>Valcea</td>
</tr>
<tr>
      <td>Olt</td>
</tr>
<tr>
      <td>Gorj</td>
</tr>
<tr>
      <td>Mehedinti</td>
</tr>
<tr>
      <td>Dolj</td>
</tr>


<tr>
      <td rowspan = "10">Transilvania</td>
      <td>Hunedoara</td>
</tr>
<tr>
      <td>Alba</td>
</tr>
<tr>
      <td>Sibiu</td>
</tr>
<tr>
      <td>Brasov</td>
</tr>
<tr>
      <td>Covasna</td>
</tr>
<tr>
      <td>Harghita</td>
</tr>
<tr>
```

```
                    <td>Mures</td>
                </tr>
                <tr>
                    <td>Cluj</td>
                </tr>
                <tr>
                    <td>Salaj</td>
                </tr>
                <tr>
                    <td>Bistrita-Nasaud</td>
                </tr>

                <tr>
                    <td rowspan = "2">Crisana</td>
                    <td>Arad</td>
                </tr>
                <tr>
                    <td>Bihor</td>
                </tr>

                <tr>
                    <td rowspan = "2">Maramures</td>
                    <td>Satu Mare</td>
        </tr>
                <tr>
                    <td>Maramures</td>
                </tr>
        </table>
      </div>
    </article>

    <footer>
      <p>Come visit and enjoy an eastern latin paradise!</p>
    </footer>
  </body>
</html>
```

Appendix B

CSS Source Code

```
/* reset styles */
html {
  font-size: 16px;
}

body, header, h1, nav, p, article, div, figure, img, table, th, td, footer{
  border: 0;
  padding: 0;
  margin: 0;
}

img {
  max-width: 100%;
  height: 15;
  width: 15;
}

table {
  border-collapse: collapse;
}

th, td {
  border: 1px solid #dddddd;
  text-align: center;
}

/* body */
body {
  margin: 0 auto;
  font-family: Andale Mono, monospace;
}

p {
  font-size: 1.3em;
  line-height: 1.6em;
}
```

```css
/* header section */
h1 {
  text-align: center;
}

/* site navigation bar */
nav.navbar {
  background-color: chartreuse;
  color: white;
  text-align: center;
}

nav.navbar p {
  margin: 0.3em 0.3em;
  display: inline-block;
}

nav.navbar a:link {
  color: white;
  text-decoration: none;
  font-weight: bold;
}

nav.navbar a:visited {
  color: darkgreen;
}

nav.navbar a:hover, nav.navbar a:focus {
  color: slategrey;
}

/* main content */
article {
  margin: 0 auto;
  height: 837px;
  padding: 1.4em;
  background: #7eccec;
  background: url("images/romania-landscape.jpg") no-repeat;
  background-size: 100%;
  font-size: 11px;
}

article div {
  max-width: 1100px;
  margin: 12em auto;
  padding: 1% 1%;
  background-color: white;
  overflow: auto;
}

article figure {
  max-width: 100%;
  margin-left: 0.5em;
  margin-right: 0.5em;
```

```
  margin-top: 7em;
  float: right;
}

/* footer section */
footer {
  padding: 0.6em;
  background-color: chartreuse;
  color: white;
  text-align: center;
}

footer p {
  font-size: 1em;
}
```
