Bachelor Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Email Discovery in Modern Applications

Andrii Pastukh
Department of Computer Science
Chair of Computer Science II (Secure Software Systems)

Prof. Dr.-Ing. Alexandra Dmitrienko
First Reviewer

M.Sc. Christoph Sendner
First Advisor

Submission
17. July 2023

Abstract

Email Discovery and Contact Discovery are processes that allow a user to find out which contacts from his phone book are using the same service. In the first case, the email address is used as an identifier, and in the second, the phone number. However, these processes come with privacy vulnerabilities that need to be addressed. One significant vulnerability is the potential for uploading all contacts and data of individuals who have not explicitly given permission for their information to be shared. When a user grants access to their phone book or contact list to a service, the people listed in their contacts have not necessarily consented to their data being uploaded to the server.

This problem opens up the possibility of unauthorized access to sensitive data by third parties or even employees of the service provider. Data leaks and server attacks are also potential risks. Additionally, enumeration and crawling attacks pose threats by exploiting the ability to access data from a user's phone book during the Email or Contact Discovery process. The danger of crawling attacks lies in the fact that they can be carried out without requiring special equipment or in-depth knowledge of the process for each specific service.

As shown in the previous work, almost all modern instant messengers that use Contact Discovery are vulnerable to such crawling attacks. The purpose of our work is to study the possibility of carrying out crawling attacks on services that use an email address as an identifier, i.e., applications using Email Discovery. We consider instant messengers, job search applications, as well as gaming platforms and other applications. Our work shows that there are three applications using Email Discovery that turned out to be vulnerable to such crawling attacks: Xing, LinkedIn, and Google Contacts. To do this, we generate millions of email addresses that are then uploaded to the services to get information about whether the user is registered and to discover what data we can get with this information.

The attacks carried out manually on the LinkedIn and Xing services proved to be ineffective primarily due to the implementation of Rate Limits. These limits restrict the number of requests allowed within a specific time frame, thereby hindering the success of our attacks. Additionally, the inclusion of user IP Address verification further contributed to the inability to extract user data successfully. As a result, the outcome of our attacks yielded insufficient results in terms of accessing and retrieving user data. The attack on the Google Contacts application shows the most devastating results. We develop a crawler, which allowed us to carry out a major crawling attack on this application. As a result, we are able to check 3046656 generated contacts. We also find that there are no Rate Limits enforced by Google to prevent such attacks. Thus, we show that, using a regular user PC, it is possible to carry out a crawling attack and find out such personal data of users as first name, last name, photo, links to other social networks, as well as online status.

Furthermore, we analyze existing methods for protecting the Email Discovery process and propose privacy enhancements for the Google Contacts service.

Zusammenfassung

Email Discovery und Contact Discovery sind Prozesse, die es einem Benutzer ermöglichen, andere Benutzer aus seinem Telefonbuch desselben Dienstes zu finden. Als Identifikator werden E-Mail-Adresse und die Telefonnummer entsprechend verwendet. Diese Verfahren haben die Sicherheitsrisiken und die Schwachstellen in Bezug auf den Datenschutz. Ein der Hauptprobleme ist die Möglichkeit, die Daten des Benutzers aus seinem Telefonbuch ohne dessen Zustimmung teilweise herunterzuladen.

Diese Schwäche ermöglicht den Angestellten des Dienstleisters und den Dritten den Zugriff zu diesen Daten. Datenlecks und Angriffe auf Server sind auch möglich. Eine weitere Bedrohung stellen Enumeration- und Crawling-Angriffe dar, die die Möglichkeit ausnutzen, während des E-Mail- oder Contact Discovery Prozesses Zugriff auf die Daten aus dem Telefonbuch des Benutzers zu erhalten. Die Bedrohung durch solche Angriffe besteht darin, dass man für die Durchführung dieses Angriffs nicht über spezielle Ausrüstung oder besondere Kenntnisse über diesen Prozess in jedem Dienst verfügen muss.

Wie in der früheren Arbeit gezeigt wird, sind fast alle modernen Instant Messenger, die Contact Discovery verwenden, für solche Crawling-Angriffe anfällig. Der Zweck unserer Arbeit ist, die Möglichkeit zu untersuchen, Crawling-Angriffe auf Dienste durchzuführen, die eine E-Mail-Adresse als Identifikator verwenden, d.h. Anwendungen, die Email Discovery nutzen. Wir betrachten dabei die Instant Messengers, Anwendungen für die Stellensuche sowie Spieleplattformen und andere Anwendungen. Es gibt drei Anwendungen, die Email Discovery verwenden und sich als anfällig für die Crawling-Angriffe erweisen: Xing, LinkedIn und Google Contacts. Zu diesem Zweck generieren wir Millionen von E-Mail-Adressen, die dann in die Dienste hochgeladen werden, um die Information darüber zu erhalten, ob der Benutzer registriert ist und welche Daten wir mit dieser Information erhalten können.

Angriffe auf die Dienste LinkedIn und Xing erwiesen sich aufgrund von Ratenbegrenzungen, die die Anzahl der zulässigen Anfragen für einen bestimmten Zeitraum begrenzen, sowie aufgrund der eingebauten Überprüfung der IP-Adresse des Nutzers als unwirksam. Daher waren die Ergebnisse unserer Angriffe nicht ausreichend, um erfolgreich Nutzerdaten zu extrahieren. Der Angriff auf die Google Contacts zeigt die besten Ergebnisse. Für sie entwickeln wir einen Crawler. Er ermöglicht uns, einen großen Crawling-Angriff durchzuführen, wodurch wir mehr als 3 Millionen generierte Kontakte überprüfen können. Wir zeigen auch, dass Google keine ausreichenden Ratenbeschränkungen verwendet, um solche Angriffe zu verhindern. Wir zeigen außerdem, dass es möglich ist, mit einem normalen Benutzer-PC einen Crawling-Angriff durchzuführen und persönliche Daten von Benutzern wie Vorname, Nachname, Foto, Links zu anderen sozialen Netzwerken sowie den Online-Status herauszufinden.

Wir untersuchen auch die neuesten Optionen zum Schutz des E-Mail-Ermittlungsprozesses und schlagen Verbesserungen für den Datenschutz beim Google-Kontaktdienst vor.

Contents

1 Introduction

The era of traditional address books is long gone as we embrace the age of fast computers, mobile phones, and various other electronic devices. However, the fundamental principle remains unchanged: users desire to stay connected and exchange information with others, whether it be for personal or professional purposes such as applying for a job. This connectivity is made possible through applications installed on their devices. According to Statista [1], the average German user has anywhere between 11 and 20 applications on their phone. When users install a new application, they naturally want to identify which contacts in their address book have already installed the same application, enabling them to establish communication effortlessly. To facilitate this process, Contact Discovery and Email Discovery functionalities are employed to aid in user discovery.

Contact Discovery is a crucial feature utilized by various applications, especially messaging services, to identify which contacts from a user's address book are already registered with the respective service. This enables new users to swiftly and conveniently find and connect with their desired contacts, facilitating communication and interaction. Email Discovery, on the other hand, is a protocol specifically designed for discovering email contacts. Typically, the process of Contact Discovery or Email Discovery is seamlessly integrated into the user interface of the application, functioning behind the scenes. During this process, all the user's contacts are uploaded to the server and cross-referenced with the service's user database. This straightforward approach is widely employed by almost all messaging applications, including popular platforms like WhatsApp [2].

These processes have many possible security and privacy problems. If a user gives the service a permission to upload all contacts, the data of people who have not agreed to this will be stored on the server. These contacts are permanently stored, even if there is no match between the contacts [3]. This information about people can later help to find newly registered users from the address book. At the same time, it could also be (mis)used to create a complete social graph of the user [4]. Such sensitive information can be used for spamming: Targeted advertising, phishing, and even for threats to a person. Some companies do not handle such sensitive data carefully enough. Confidential information can be stolen even from large and well-known companies, such as Facebook or WhatsApp [5][6]. For instance, this was demonstrated by the case that happened to Facebook in 2014, when Facebook users received friend recommendations from strangers who had been to the same psychiatrists. This happened because Facebook shared the data with another company[7].

The process of Email Discovery has the same vulnerability as the process of Contact Discovery. Therefore, email addresses can be also stolen from user's email contact book. This information can be then exploited, for instance, to send spam emails. According to the Bundesamt für Sicherheit in der Informationstechnik (German Federal Office for Information Security), the average percentage of spam emails in Germany is 65% [8]. Although the number of spam emails is already very high, the attacker can furthermore use information about the victim's circle of friends to carry out targeted spam attacks. It cannot also be ruled out that the server with this data can be compromised, which will subsequently lead to a massive leak of user data. It is especially important for the Contact Discovery or Email Discovery process to be protected in authoritarian countries where instant messengers or social networks are prohibited, and the use of such a service can be simply dangerous for a person. For example, in Iran, some services, such as WhatsApp, Telegram or Instagram, are blocked or are trying to be blocked [9] [10]. There was also a need for a messenger that would be protected during the protests in Hong Kong [11]. In recent years, the topic of privacy has been in great demand among users, and a process such as Email Discovery must be protected.

The application of simple protective mechanisms before uploading to the service provider, such as hashing telephone numbers or email addresses of contacts, is not particularly helpful. These hashes can be determined using a brute force attack since the numbers have a relatively small range of possible variations. Furthermore, the service provider can still tell whether two users share a contact even a long time after running the discovery routine by storing the received hash values [12]. There are also algorithms that make the matching process much safer, such as Private Set Intersection (PSI) [13] [12] [14]. We will take a closer look at the PSI protocol, and also study the state-of-the-art solutions. However, these protocols are quite far from being used in real life in applications with millions of users due to the high computational costs, as well as communication overhead [15]. However, even using such a protocol, the Contact Discovery or Email Discovery process is not completely protected, because it does not provide against a crawling attack.

Crawling is another way to attack such services. To do this, an attacker needs to generate random phone numbers or email addresses and add them to his contacts in order to initialize the Email Discovery or Contact Discovery process. In our case, crawling is the targeted search for public information about people and their contacts. Since there is no restriction on user registration and each third party can create hundreds and thousands of accounts, such attacks can be carried out. And since all contact information, and not just a phone number or an email address, can be stored during the Contact and Email Discovery process, it is possible for attackers to extract personal information.

Firstly, we find out which services using the Email Discovery process are vulnerable to crawling attacks. For the Contact Discovery process with phone numbers, similar studies were carried out, which showed that all the largest instant messengers are vulnerable to such attacks, and it was possible to find tens or even hundreds of thousands of contacts per day[16][17][15][18];

In this thesis, we make the following contribution:

- In our case, we want to conduct a similar study, but with services that use Email Discovery. The initial list of services that have been studied consists of the following messengers: WhatsApp [5], Telegram [19], Signal Messenger [20] and Facebook [6]. Also, we checked services for job search like Xing [21] and LinkedIn [22] and gaming platforms, such as Steam [23] and PlayStation Network [24]. In addition, we decided to check the Google Contacts service, which is used in almost all Google applications, such as Gmail [25]. We define three main services for us, two of which are job search platforms, as well as the highly specialized social networks LinkedIn and

Xing. After conducting our experiments, we determine that these two services are vulnerable to crawling attacks, and we also get access to such personal information as photos, places of work, places of study, date of birth, names, as well as online status. The attempted manual attacks on the LinkedIn and Xing services proved to be unsuccessful, mainly due to the presence of Rate Limits. These limits impose restrictions on the number of requests permitted within a given time period, effectively impeding the progress of our attacks. Furthermore, the implementation of user IP Address verification also played a role in preventing the successful extraction of user data. Due to these restrictions, we were unable to conduct a large-scale attack;

- For Google Contacts we developed a crawler with which we were able to carry out an automatic attack, and were able to cross-check 3046656 contacts in a 48 hours long experiment based on the email addresses we generated, and found 688742 real contacts (22.61% out of checked 3046656 contacts). We found that there are no adequate Rate Limits enforced to prevent such attacks, and confirmed the hypothesis that user data is at risk. We also checked, how much of 688742 found real users have a profile image, and were able to find 907 (0.132%) with a profile image. We used a regular mid-end PC that a normal user can have for our attack. In order to circumvent potential IP address bans, we use a VPN service and utilized email providers that do not require phone number verification to create multiple accounts. This confirms that such an attack cannot be limited by the complexity of the hardware and can be carried out by an ordinary user;
- We also discuss possible methods that can be applied to secure the Email Discovery process from leaks from the server, as well as from such crawling attacks.

Outline. The remaining of this thesis is arranged as follows: in Section 2 we provide background information necessary for our work. Related works are introduced in the Section 3. In Section 4, we introduce the attacker model, explain the basic design of our attacks, and discuss the attacks on LinkedIn and Xing. In Section 5 we explain the specific implementation of the app, that generates contacts in CSV-format and of the crawler, developed for Google Contacts. In Section 6 we present the results of our work, as well as a discussion about potential methods for protecting the Email Discovery process. In Section 7, we summarize our work and draw conclusions.

2 Background

In this Chapter, the theoretical base of terms and technologies, such as Contact and Email Discovery, spam and phishing attacks and crawling attacks, necessary for our work, are provided. In particular, in Section 2.1 we describe the structure of email addresses. We define the processes of Contact and Email Discovery in Section 2.2. In Sections 2.3 and 2.4 we consider the dangers of Spam and Phishing Attacks and Crawling Attacks respectively. In Sections 2.5 we discuss the protection approaches used for the Email and Contact Discovery processes.

2.1 Email Addresses

An email address is a unique identifier used to send and receive electronic mail messages over the Internet [26]. It typically follows a specific structure, as shown in the following Figure 2.1.

Figure 2.1: Email address structure example.

The first part of the structure, the ”username”, is chosen by the email account holder and is used for logging in into the email account. The ”domain” is the name of the organization or service that is providing the email service. The final part of the structure, the ”top-level domain” (TLD), specifies the type of the organization or service the domain belongs to. For instance, .com for commercial organizations, .edu for educational institutions, .gov for government agencies, .org for non-profit organizations, and so on. There are also country-specific top-level domains, such as .de for Germany or .us for the USA. It is worth noting

that there are different specifications and standards for the structure of email addresses, but this is the most common one [26] [27].

Email addresses are characterized by high entropy due to their structure. According to the specification [26], the username can be up to 255 characters long, and must consist of ASCII characters. We know from the work of M. Marx et al. [28] that the total number of possible emails is 2.2×10^{120}. Since each country has its own most popular domains, by selecting addresses within these domains, it is possible to achieve better results of the attack on this country's services. For instance, the majority of users in Germany use the domain gmx.de [29].

2.2 Contact and Email Discovery

Contact Discovery is a process that allows applications and messaging services to identify which of the user's contacts are already registered with the service. Email Discovery is another protocol that focuses on recognizing email contacts. This enables new users to find the contacts they need to communicate or interact with quickly and easily. Many messaging apps, such as WhatsApp [2], use this method. As a first step, the users install the application on their phone and grant access to their phone book. To differentiate between registered and non-registered users, the service must compare the user's contact list with a comprehensive database of all registered users. This can be achieved by either sending all the user's contacts to the messenger service or vice versa. Since the user has a much smaller set of contacts, usually the user's contacts are sent to the service provider's to find the intersection between the contacts of user and user database of service provider and to keep the user database secret. Subsequently, the service saves the user's contacts and presents the user with a list of their contacts who are already registered on the same service. Additionally, a message is sent to the users from this list, notifying them that the user has joined the service as well, see Figure 2.2.

Figure 2.2: Typical process of Contact Discovery.

However, Contact Discovery methods are vulnerable to various types of attacks, which can compromise user privacy and security.

One of the most common types of attacks on Contact and Email Discovery processes is the leakage of contacts. This type of attack involves analyzing submitted contact books to identify which contacts are using the messenger app. Attackers can then use this information for various purposes, such as spamming or phishing. This attack is a significant threat to user privacy. Such attacks can be carried out by attacking the server or when the malicious access of the employees of service provider to the user data occurs.

Another type of attack is the leakage of information about users using the service and their profile information, commonly referred to as crawling attacks. These attacks involve the

unauthorized scraping of user data from the messenger app. This type of attack is also a significant threat to user security.

To protect the process of Contact Discovery and Email Discovery, some applications use different methods of protection against both types of attacks. Each type of attack requires its own methods. Hashing phone numbers is a commonly used technique for protecting user privacy in mobile messenger apps. This approach involves transforming the phone numbers into a unique string of characters, known as a hash value, before uploading them to the service. This ensures that user phone numbers are not visible to the service or any other third party, making it more difficult for attackers to identify which contacts are using the messenger app. In addition, some messenger services use complex leaky bucket structures to enforce rate limiting. This approach involves limiting the number of requests that can be made to the service within a certain period of time, which helps to minimize damage from possible crawling attacks. For example, Signal uses both techniques for protecting the data [16].

2.3 Spam or Phishing Attacks

Identity theft, a serious crime among Internet users, involves impersonating someone to steal and use their personal information like bank details, social security number or credit card numbers. One method used to commit an identity theft is called a phishing attack. Phishing attacks involve the use of social engineering techniques by the attacker, who sends an email with an embedded link to a malicious website, which the victim is then redirected to after clicking on the link. Phishers have also evolved from sending mass-email messages to more targeted attacks, known as spear-phishing, where they specifically target a single individual or a small group of victims.

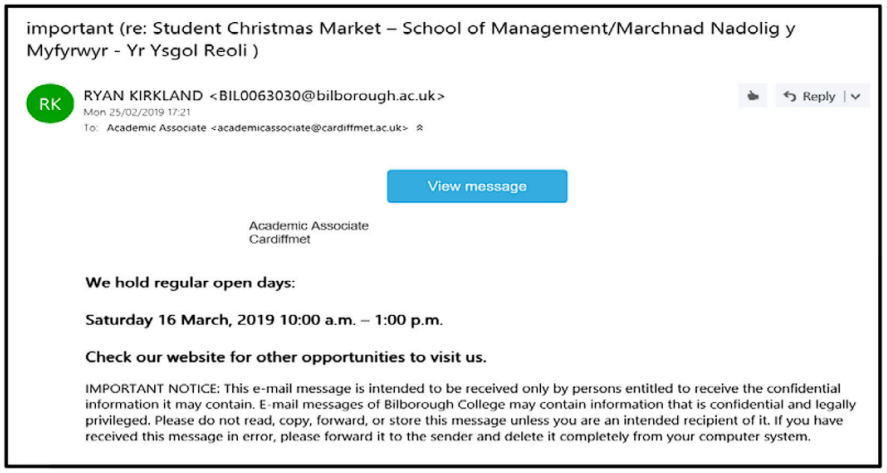

Figure 2.3: Screenshot of a phishing email [30].

Identity theft can have serious consequences, such as a financial loss and damage to credit scores, making it crucial for individuals to take steps to protect their personal information. Additionally, individuals should be aware of the warning signs of a phishing attack, such as emails asking for personal information, or emails that appear to be from a legitimate source but contain spelling or grammar errors. On the Figure 2.3, one can see an example of such a message. Furthermore, the target of 86% of all phishing attacks is to steal confident information from the users of social networks and media, highlighting the importance of protecting information obtained through Contact and Email Discovery [30].

2.4 Crawling Attacks

Crawling or Enumeration attack is a type of cyberattack in which an attacker uses automated software, known as a web crawler or spider, to scan and index web pages and other resources on a targeted website or network. The purpose of this attack is to identify vulnerabilities and gain unauthorized access to the targeted site. The process of a crawling attack typically starts with the attacker identifying a target website or network. They then use automated software to systematically scan and index all pages and resources on the site. During the scanning process, the web crawler also searches for potential vulnerabilities, including unpatched software, weak passwords, and open ports. However, it is important to note that this is not always the case. In certain situations, such as the crawler we develop, it can be utilized to automatically upload generated contacts to the server to verify their existence, thereby gaining access to them.

An attacker, similar to any regular user, can initiate the process of Contact or Email Discovery with the service. As described earlier, this process entails sending the attacker's existing contact list to the service, which then compares it with the service provider's database to identify any matches. The attacker can then modify their contact list and repeat the process. By populating the contact list with specific candidates they want to test, the attacker can iterate the process until they successfully obtain a comprehensive list of all users registered with the service. This is possible if there are no security measures in place to prevent this type of attack. The attacker needs neither special equipment nor good programming skills to carry out a crawling attack misusing Contact or Email Discovery services. Crawling attacks can be carried out using native API, phone emulators, or by exploiting the vulnerabilities of the services that are being targeted. Since most services have no limits on the creation of new user profiles, attackers can create thousands of profiles to crawl user data or accounts [16].

Protection against crawling attacks can be a challenging task. Limiting the possibility to search other users can reduce the usability of an application, and the simplest solution to this problem is to limit the number of requests (rate limit) that a user may send. However, as shown in the work of Hagen et al. [16], it is possible to check thousands of contacts per day in all modern messengers.

Moreover, the information that a person is using a certain application can even put the person at risk if this application is banned in a country where this person currently lives. It is also possible to use such data about users to create a model that can later be used for targeted advertising or scam [31].

2.5 Protection Approaches

In this Section of the work, we describe modern ways to protect the process of Contact Discovery in general terms. We look at the ways to protect against attacks on servers, such as Hashing and Private Set Intersection protocol. We also look at a popular way to limit the rate to counter crawling attacks.

2.5.1 CAPTCHA

CAPTCHA and reCAPTCHA are important security measures that prevents unauthorized access to services and ensures the protection of personal data. The primary function of CAPTCHA and reCAPTCHA is to prevent unauthorized access to services and detect bots and crawling scripts. CAPTCHA requests the user to type a given text or number

in the text field to confirm that he is not a robot. It's appearance can hinder the functionality of crawlers that rely on accessing websites frequently. Figure 2.4 shows a typical reCAPTCHA verification process. ReCAPTCHA provides a set of pictures, from which the user must choose the pictures with specific elements, for instance, crosswalks or bicycles to prove, that he is not a robot. The usual bot or crawler is not able to complete this task.

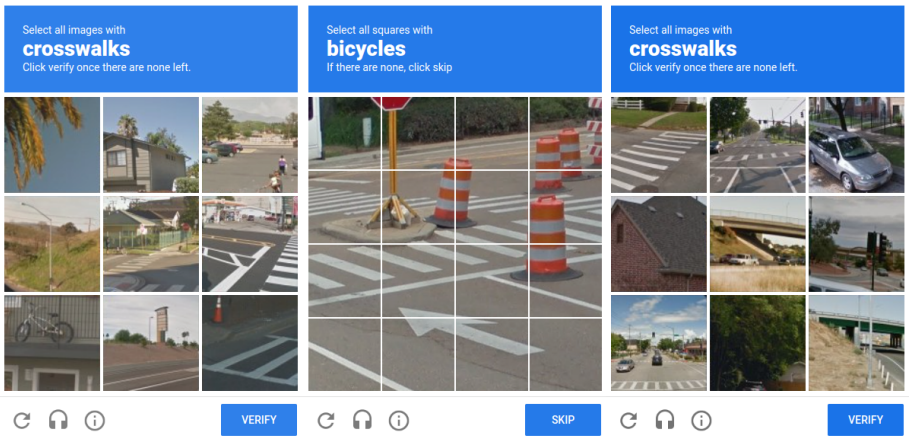

Figure 2.4: reCAPTCHA example

2.5.2 Hashing

Secure Hash Algorithms (SHA) are cryptographic hash functions developed and provided by the National Institute of Standards and Technology (NIST). These algorithms are commonly used by messaging services like WhatsApp [5] and Signal [20] to obscure the phone numbers of users from their own servers [16]. One of the key properties of a hash function is that it must be deterministic, meaning it must consistently produce the same output for a given input. Hash functions are used in this context because of their one-way property, which means that once a phone number is hashed, it is difficult to reverse the process and obtain the original phone number. An example of a hash table can be seen in the provided Figure 2.5

Look-Up Table

One could potentially reverse the hash of a phone number, if the number and its hash were stored as key-value pairs in a database, such as Redis [32]. It is worth nothing that such a database would need to be huge in size. E.g., the estimated amount of valid phone numbers is in the range of 10^{15} [33].

Brute-force

The way to reverse a secure hash is through a process known as brute-forcing, which involves calculating the hash of every possible phone number until the correct one is found. The entropy of phone numbers is relatively low, as they follow a certain international specification [34]. In contrast, the entropy of email addresses is typically much higher, making them harder to reverse through brute-force methods.

Rainbow Tables

Brute-force methods for hash reversal have been found to be time-consuming and resource-intensive. To address this issue, an alternative approach is to use Rainbow Tables [35]. In this method, the attacker pre-calculates hash digests based on a defined input space and then uses a reduction function to reduce the digest back to the input space. This process is

repeated multiple times. As demonstrated in the work of C. Sendner [33], Rainbow Tables can be effectively applied to reverse hashed phone numbers in the context of Contact Discovery.

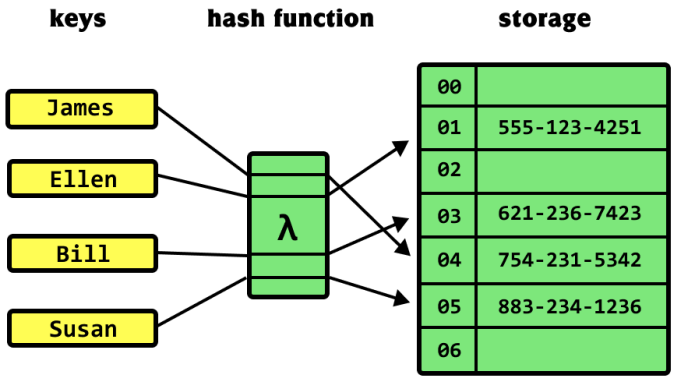

Figure 2.5: Hash table example [36]

Per the General Data Protection Regulation (GDPR) [37], phone numbers are considered personal information that must be protected. As a result, some services use hashing to protect this information. The research by M. Marx et al. [28] and by C. Sendner [33] has shown that it is still possible to quickly reverse a hash of such information as phone numbers and through various brute-force methods. That is why the use of hashing when sending contacts to the server is a practically useless measure and should not be considered as a way to protect data.

2.5.3 Private Set Intersection (PSI)

The Private Set Intersection (PSI) protocol is a method to securely perform Contact Discovery, which allows users to find the intersection of their contacts with those registered on a service without revealing the contents of either set to the other party. The main focus of PSI protocols is to prevent revealing the users data to the service provider. At the same time, the service provider must also have the possibility to implement rate limits, even when PSI is used, to counter crawling attacks.

The process starts with the service provider creating a secret key and using it to encrypt the user database. This encrypted database is then inserted into a Bloom filter or a Cuckoo filter, which is a data structure that allows the efficient checking of whether an element is in a set. The service provider then sends this data to the user.

An example of Oblivious Pseudo-Random Function Protocol (OPRF) process is shown on the Figure 2.6. The user and the service provider use an OPRF to enable the user to obtain encryptions of their own data entries under the secret key from the service provider without revealing the contents of the user's address book or the secret key to the service provider. This way the user can check locally if any of its address book entries exist in the encrypted database provided by the service provider [13] [12] [14]. We can categorize PSI protocols into balanced and unbalanced PSI.

Balanced PSI

In a balanced Private Set Intersection (PSI) protocol, both sets being compared, such as the user's contact list and the service provider's database, are of similar size. This results in

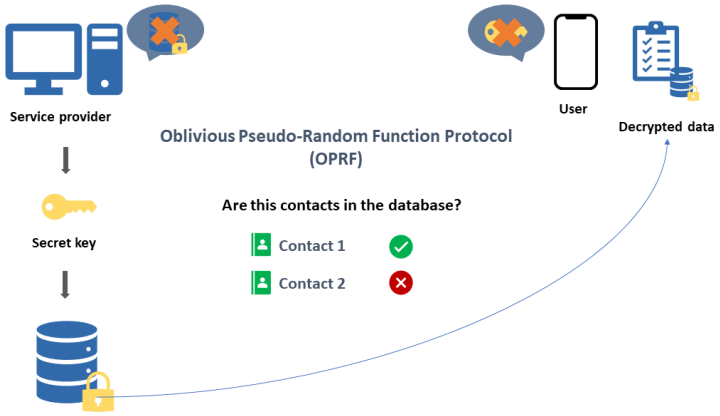

Figure 2.6: OPRF Example

similar computational and communication costs for both the user and the service provider. However, in real life, the database of service providers is usually many times larger than the phone book of an individual user [14].

Unbalanced PSI

In contrast, in an unbalanced PSI protocol, the sizes of the sets being compared differ greatly. In this case, the user's contact list may be significantly smaller than the service provider's database. This results in unequal computational and communication costs, with the user experiencing lower costs as they have less data to process, while the service provider incurs higher costs due to the larger amount of data they must compute.

The PSI protocol is a secure and efficient way to perform Contact Discovery. It ensures that only the contacts registered on the service are shared with the service provider, and that the contents of the user's and service provider's sets remain private. But it has a few important issues that we discuss below.

The most implementations of PSI work great with a small amount of data, but they are far from practical when used at scale. They have high computation and/or communication complexity. They have also lacked optimization for mobile devices. The security of PSI protocols can be difficult to prove. In this paper, we discuss the possibility of using the state-of-the-art PSI protocol to protect the process of Contact and Email Discovery [16] [13] [12] [33].

2.5.4 Leaky Bucket Structure

The leaky bucket algorithm is a congestion control mechanism used to regulate the rate at which requests are made to a contact discovery service. It operates by enforcing a fixed limit on the rate of incoming requests, preventing users from overwhelming the service and causing a degradation in performance or service downtime.

The algorithm is modeled on a "bucket" that can hold a fixed amount of "water" or requests. The bucket is filled at a various rate, but if it becomes full, any additional requests are immediately discarded. The bucket "leaks" requests at a regulated rate. The rate at which the bucket is filled is known as the "token generation rate," while the maximum number of requests that the bucket can hold is known as the "bucket size." Figure 2.7 illustrates an example of such a structure.

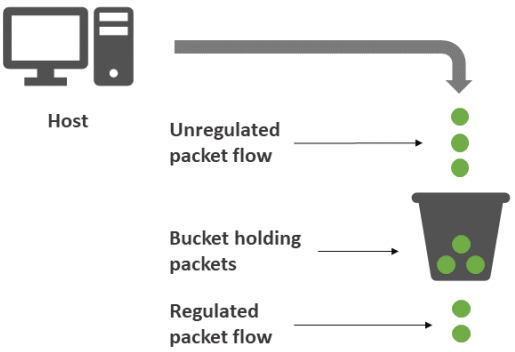

Figure 2.7: Leaky Bucket Structure example

When a user requests contact information from a contact discovery service, a token is removed from the bucket, allowing the request to be processed. If the bucket is empty, no tokens are available, and the request is rejected or queued until a token becomes available. By regulating the rate at which tokens are generated and consumed, the leaky bucket algorithm can prevent users from making too many requests too quickly, ensuring fair usage of the service and preventing overload or abuse. As demonstrated in Hagen et al. [16] work, many instant messengers use this structure in order to limit the number of requests in the context of contact discovery methods.

Overall, the leaky bucket algorithm provides a simple but effective way to control the rate of requests made to contact discovery services, helping to maintain performance and prevent service disruptions [38].

2.6 Services with Email Discovery

In this Section, we consider the selected services and check which of them use Email Discovery. As a result, we needed to find alternative services that utilize email addresses to synchronize contacts. Initially, we planned to focus on studying the most popular instant messaging apps, as these have commonly been used in studies where phone numbers were the primary identifier. However, we soon realized that the majority of these apps don't use email address as an identifier.

This lead us to expand our research to include job search services, as these platforms specialize in finding individuals by their email addresses. Professional networking platforms, such as LinkedIn [22], Xing [21] are of particular research interest due to the nature of their user base. Unlike traditional social networks and messaging platforms, users on these platforms tend to provide accurate personal information, making attacks on such services potentially more harmful for users. The platform contains a wide range of personal information, including an individual's first and last name, place of employment, colleagues, educational institution, as well as a likely representation of the user via a profile photo.

Additionally, we also decide to consider gaming services, as email addresses are typically the main identifier for accounts in these types of services. By including these additional

sources, we are able to gain a more comprehensive understanding of the use of email addresses as user identifiers. The initial list of services can be found in Chapter 1. The overview of all considered services, that use Email Discovery is demonstrated in Table 2.1.

Table 2.1: Services comparison

Logo	Name	Contact Discovery	Email Discovery
	Facebook Messenger		
	Xing		
	LinkedIn		
	Google Contacts		

2.6.1 Facebook Messenger

Facebook Messenger is a messaging app developed by Facebook that was first released in 2011 as a standalone app for iOS and Android devices, separate from the main Facebook app. The goal of the app was to provide users with a faster and more efficient way to message their friends and contacts on the platform. Since its launch, Facebook Messenger has grown rapidly in popularity, becoming one of the most widely used messaging apps in the world. According to Datareportal the app had more than 1.3 billion active users in 2021 [39].

One of the unique features of Facebook Messenger is its integration with other Facebook-owned apps such as Instagram and WhatsApp. This allows users to communicate with their friends and contacts across all three platforms freely. However, this integration also raises some concerns about privacy, as it could potentially create new vulnerabilities.

Unfortunately, Facebook Messenger is a closed-source application and there is no specific information available about what data is stored on the servers and in what form [40]. However, it can be assumed that contacts are encrypted using hashing before being sent to the server, similar to WhatsApp.

Facebook Messenger also has features called Contact Upload and Contact Sync [41], which allows the app to access and upload the names, phone numbers, and email addresses in a user's address book on a daily basis to the servers, including those of users of Facebook, Messenger, and/or Instagram and other contacts who are not users or don't have an account. This means that the contacts are permanently stored, and the email address is also used as a user ID. Therefore is Facebook Messenger suitable for our experiments.

2.6.2 Xing

Xing, also known as OpenBC, is a professional networking platform that has gained widespread popularity in Europe, particularly in Germany. The platform facilitates connections between users, enabling them to explore job opportunities and stay informed

about industry-related news and events. Users can create a professional profile, post job listings, and search for other professionals within their field. The platform was established in 2003 and is headquartered in Hamburg, Germany.

According to the Statista [42], the number of Xing users has exceeded 20 million people in 2022 in the DACH region (Germany, Switzerland, Austria).

This platform was developed in Germany and one of its main achievements is the privacy of personal data. Due to the fact that the company is located in Germany, where one of the strictest data protection laws apply, it's allowed to completely delete contact data and a user's account if he decides to delete an account, according to Xing [21].

As one can see on the Privacy page [43], Xing uses email as user ID and allows to search for users by that ID. Xing allows users to synchronize their contacts from various email services, such as Gmail or Outlook, in order to connect with their contacts on the platform easily. This feature is known as contact sync. One can also use the phone book with saved contacts to synchronize contacts. This service is suitable for our experiments.

Previously, it was also possible to export contacts to third-party applications in the CSV format. It could speed up crawling attacks, but the company closed the ability to export contacts in 2021 [44].

2.6.3 LinkedIn

LinkedIn is a professional networking platform that allows users to connect with other professionals, find job opportunities, and stay informed about industry news and events. It was launched in 2002 and has grown to be one of the largest professional networking sites, with over 800 million members worldwide [45].

LinkedIn has a robust privacy policy that outlines how user data can be collected, used and shared. Users have the ability to control their privacy settings, such as to choose who can see their profile and to limit the information that is shared with third-party advertisers. LinkedIn also uses encryption and other security measures to protect user data. However, as with most social media platforms, there are concerns about the collection of user data and the use of this data for targeted advertising [46].

LinkedIn collects a significant amount of data that may be relevant to our crawling attack. The platform utilizes the email address as a personal identifier and employs contact synchronization, both via personal computers and smartphones, to save contact data and to conduct daily checks for new contacts [46]. Additionally, if synchronization with external services, such as Facebook or Gmail, is enabled on mobile devices, contacts from those services are automatically imported into LinkedIn. Furthermore, the platform saves the email header when conducting contact synchronization [47].

It is also important to note that LinkedIn is a subsidiary of Microsoft, which means that personal data can be shared with Microsoft and its affiliates according to their privacy policy [49].

In 2021 LinkedIn experienced a personal data leak affecting 756 million users, or 92 percent of all users of the service [48]. One can see more information about the data contained in the leak in Figure 2.8. This Figure demonstrates what user data we can obtain during our experiments. For instance, such information as gender, username, work email, mobile phone number, etc. could be leaked.

Due to the huge user base and a lot of personal data stored about users, such services are of particular interest to us in the context of crawling attacks. Therefore, LinkedIn is also suitable for our further investigation.

Figure 2.8: LinkedIn Leak Sample [48]

2.6.4 Google Contacts

Google Contacts is an online service offered by Google that enables users to store and manage their contacts, which can include names, addresses, phone numbers, email addresses, and other contact information. The service has a deep integration with other Google services such as Gmail, Google Calendar and Google Drive, and provides users with an easy access to their contacts from anywhere and on any device. Google Contacts also provides features such as the ability to organize contacts into groups, add notes and labels to contacts and merge duplicate contacts. [50]. The service is available on the web, on Android and iOS devices through the Google Contacts app.

One of the most relevant features of Google Contacts for our research is its integration with Gmail, as we are looking for services that use Email Discovery. In Google Contacts users can access it through their Google account.

Gmail is an email service provided by Google. It was launched in 2004 and has since become one of the most popular email services in the world, with over 1.5 billion active users [51]. Gmail offers a wide range of features: The ability to send and receive emails, organize messages into labels and folders and search for specific emails. Due to the deep integration of Google Contacts with Gmail, this is one of the most interesting services for our work. Gmail contact sync works on all mobile devices [52]. It is also possible to import contacts from other services, such as Outlook. Additionally, users can export their contacts, and search for contacts quickly and easily [53].

Google has extensive privacy settings. It is possible to completely hide your data from other users [54]. However, as a 2019 study demonstrate, that even after major scandals with data breaches and other privacy issues in huge services like Google and Facebook, people are changing their privacy settings. Only 35% of those surveyed have made their account private, and more than 22% have not changed even a single setting [55].

Therefore, based on data from Google, we know that other users can see data such as birthday, gender, job information, personal contact information, address, education, as well as photos, nickname and email address [54].

3 Related Work

In this Chapter, we discuss the related work dealing with the topics of Email and Contact Discovery. In this work, we use many different related works.

As already mentioned in 1, the Contact Discovery process has privacy issues and the data stored by service providers using this process can be stolen or used by third parties to carry out an enumeration attack or phishing attacks. Works demonstrating such possibilities are considered in Section 3.2.

Finding intersections from two sets is one of the main challenges for solving privacy problems for the Contact and Email Discovery processes. A PSI protocol is used to solve it. Works detailing various implementations of this protocol have been reviewed in Section 3.1.

We also overview works that show that hashing data before sending it to the service servers is not a reliable protection in Section 3.3.

3.1 Private Set Intersection

Private Set Intersection (PSI) [13] [12] [14] is a cryptographic technique that is applicable to many privacy-sensitive scenarios, such as contact discovery. However, the majority of existing solutions is inefficient for a large number of inputs, which is common in client-server scenarios. The PSI protocol, while serving as a valuable security measure, does not negate the necessity of implementing rate limits as a safeguard against crawling attacks. In the paper of Kiss et al. [13], PSI was analyzed and the efficiency of existing PSI protocols was optimized to support precomputation so they can efficiently deal with such input sets. Four protocols were implemented, and the experiments showed that a protocol based on securely evaluating a garbled AES circuit achieved the fastest setup time.

In the work by Kales et al. [12], the researchers dealt with the Private Set Intersection (PSI) protocol. The authors developed a new protocol based on two protocols presented by Kiss et al. [13]. Specifically, the authors presented novel precomputation techniques for correlated oblivious transfers, that reduced the online communication by the factor of 2x. The online phase of their fastest protocol was measured to take only 2.92 seconds on a real Wi-Fi connection (6.53 seconds on LTE). This quick processing time allowed

us to efficiently check 1,024 client contacts against a large-scale database containing 2^{28} entries. Finally, the developed protocol was integrated into the Signal application as a proof-of-concept.

Chen et al. [14] investigated the unbalanced PSI setting, where the receiver's set is significantly smaller than the sender's, and the receiver with the smaller set has a low-power device. A new PSI protocol has been developed that supports arbitrary length items and has a small communication complexity. In addition, a pre-processing phase was added to strengthen the security model using an Oblivious Pseudo-Random Function (OPRF). The results of the experiment showed that the protocol was optimized and worked faster than the previous protocol of Chen, Laine, and Rindal and worked with less data to communicate [56].

Pinkas et al. [57] proposed another modification of the PSI protocol. The new approach called Phasing, which uses permutation-based hashing to reduce the length of items mapped to bins without causing collisions, was proposed in this paper to improve the performance of PSI protocols. Phasing is applied to circuit-based and OT (Oblivious Transfer)-based PSI protocols, resulting in protocols that are up to 5 times faster and 20 times more efficient as the OT-based PSI protocol of Pinkas from 2014 [58], respectively. These new protocols aim to be scalable to the demands and constraints of real-world applications and are more secure than currently used insecure PSI protocols.

There has been limited research on using generic Multi-Party Computation (MPC) protocols for Private Set Intersection (PSI) [59]. However, there are many variants of the set intersection functionality that are not addressed by the existing custom PSI solutions and are easy to compute with generic MPC protocols. Pinkas [59] proposed new circuit-based protocols for computing variants of the intersection with almost linear numbers of comparisons. These protocols are based on new variants of Cuckoo hashing in two dimensions and can be extended to multiple parties. One of the protocols is asymptotically efficient, and the other has better efficiency. The proof technique for analyzing Cuckoo hashing can be generalized to standard Cuckoo hashing and other variants.

Weinert [60] explored the PSI protocol. He argues that existing protocol designs and implementations unfortunately incur an impractical computation and/or communication overhead, and proposes a new protocol that is suitable for certain scenarios: Mobile Contact Discovery, mutual authentication for Apple AirDrop, and database intersection analytics. Also, two privacy vulnerabilities in Apple's proprietary offline file sharing service of AirDrop were identified. As a privacy-preserving alternative, "PrivateDrop" was developed based on two optimized PSI protocols.

Hoepman et al. [61] developed a new way to use the Contact Discovery process. The process has a privacy issue: When user joins and enables Contact Discovery, anyone already on the service that has user's number on their contact list gets notified that new user joined, even if the user doesn't know that person. In this work, a mutual Contact Discovery protocol was developed, that only allow users to discover each other when both are in each other's contact list.

3.2 Enumeration, Phishing and Crawling Attacks

In this Section we present related works on enumeration, phishing and crawling attacks in relation to the privacy issues of Contact Discovery and Email Discovery processes. Hagen et al. [16] [17] [15] [18] showed that it is still possible to carry out large crawl attacks. The scientists successfully crawled 10% of the American phone numbers in WhatsApp and 100% of the American phone numbers in Signal. This work also found vulnerabilities in the Telegram API, through which an attacker could get sensitive data of users, even about numbers not registered in the service. Another conclusion of this work concerns the current hashing-based security protocols for Contact Discovery. Such protocols are not sufficient for the security of the Contact Discovery process. Because of this, new hashing-based protocols and strict rate limits on requests have been proposed. Also, Hagen et.al. [16] [17] [15] [18] introduces an innovative approach to the Contact Discovery process, which enables more rigorous rate limiting and can be integrated with PSI for enhanced privacy protection.

In the work of Buchenscheit et al. [31], the authors investigated how to extract the information about daily routines or even about the interlocutors from the data from apps such as WhatsApp. A user study was conducted with two independent groups, in which their presence information was collected and analyzed over four weeks of regular WhatsApp use and followed by follow-up interviews. Finally, they discuss potential solutions to mitigate these issues. For instance, the authors suggest that presence information should only be available to a user's contacts to prevent abuse by third parties.

Bilge et al. [62] investigated how easy it would be for a potential attacker to launch automated crawling and identity theft attacks against a number of popular social networking sites in order to gain access to a large volume of personal user information. The authors also demonstrated two attacks on popular social networks that aimed to automatically clone user accounts to swap "friends" from that person's social networks. As a result of these attacks authors managed to successfully clone accounts. The victims thought that they were talking to friends or acquaintances and thus shared private information.

Gupta [63] explored the abuse of phone numbers for phishing and other types of attacks. Phone numbers are unique identifiers and are verified through the telephony system, making them more trusted than traditional email addresses. Vishing (voice phishing) and SMiShing (SMS phishing) attacks have been carried out using phone numbers, and new technologies such as OTT messaging applications and social networks are now being targeted by attackers. The article outlines several hypotheses related to the abuse of phone numbers in these contexts, including the feasibility of large-scale targeted attacks on messaging applications and the use of cross-platform features to prevent spam campaigns on different social networks.

Gupta et al. [4] explored the feasibility, automation, and scalability of a variety of targeted attacks that can be carried out by abusing phone numbers. These attacks can be carried out on different channels of Over-The-Top (OTT) messaging applications such as WhatsApp, Viber, WeChat, etc. and voice, e-mail, or SMS. As a proof of concept, the authors enumerated a random pool of 1.16 million phone numbers and demonstrated that targeted attacks could be crafted against the owners of more than 250,000 phone numbers by exploiting cross-application features. Cross-application feature is the ability of different applications to share information about users based on their phone numbers. For example, if a user has registered their phone number on multiple applications, each application can

potentially share the user's information with the other applications.

Eunhyn [64] investigated the security issue of an instant messaging application, KakaoTalk, which is popular in South Korea with a focus on automated friends registration based on contacts sync. KakaoTalk uses phone number as an identifier to sync the contacts. The authors demonstrated that there are multiple ways of collecting victims personal information such as their names, phone numbers and photos, which can be potentially misused for a variety of cybercriminal activities. The experiments have shown that a user's personal data can be obtained automatically. The authors also suggested countermeasures to mitigate the discovered attacks, which have been confirmed and patched by the developers.

The issue of personal information and financial transactions being vulnerable to cybercrimes, specifically phishing attacks, is discussed in the work of Alkhalil et al. [30]. The authors state that phishing is a highly effective form of cybercrime that has evolved since the first reported attack in 1990 and is now considered one of the most frequent forms of fraud on the Internet. These attacks can lead to severe losses for victims including sensitive information, identity theft, and company and government secrets theft. The researchers aim to evaluate these attacks by identifying the current state of phishing and reviewing existing phishing techniques, and proposes a new detailed anatomy of phishing that includes different phases, types of attackers, vulnerabilities, threats, targets, mediums, and techniques used in these attacks. The goal of this proposed anatomy is to increase awareness of these attacks and to aid in the development of a holistic anti-phishing system. Countermeasures and new strategies are also discussed in the paper.

In the paper of Hu et al. [65] disposable email services were discussed, which provide temporary email addresses that allow people to register online accounts without exposing their real email addresses. The authors perform the first measurement study on disposable email services, with two main goals: To understand what disposable email services are used for and what risks are involved in common use cases, and to use disposable email services as a public gateway to collect a large-scale email dataset. The researchers collected data from 7 popular disposable email services over three months, containing 2.3 million emails sent by 210K domains. The study finds out that online accounts registered through disposable email addresses can be easily hijacked, leading to potential information leakage and financial loss. Additionally, the authors find out that third-party tracking is highly prevalent, especially in emails sent by popular services, and that trackers use various methods to hide their tracking behavior. The study also finds out that a few top trackers stand out in the tracking ecosystem, but are not yet dominating the market.

3.3 Hash Reversal

In the master thesis by Sendner [33], hash-based security measures were investigated. The developed hash inversion techniques showed that it is possible to invert a hash within 0.1 ms using in-memory databases with 10 TB of RAM. For that, two hash reversal techniques were developed: One using brute-force approach and another one using look-up databases. Furthermore, the authors presented a brute-force approach which could invert a hash in less than 100 seconds with the use of common hardware. Thus, an attacker can easily reverse hash digests of mobile phone numbers and deanonymize personally identifiable information such as phone numbers. This work demonstrates that the use of hashing when uploading contacts to the server almost useless against the leakage of contacts. The results

of this work were used in the work of Hagen et al. [15] [18].

Marx [28] disscussed about common practice of web tracking services to hash personally identifiable information (PII). The practice of web tracking services using hashes to pseudonymize PII is not a sufficient practice to protect user privacy. The researchers used IP and MAC addresses, phone numbers and email addresses as PII for this work. The services argue that they are compliant with privacy laws by pseudonymizing by hashing PII, but the authors show that the finite pre-image space of PII is limited, making it easier to attack the hashes and infer the corresponding PII. An empirical study is conducted to demonstrate that hashing PII is not an effective pseudonymization technique.

4 Methodology

In this Chapter we describe the concepts and methodology of our experiments. In Section 4.1 we discuss the goals and requirements of an attacker. In Section 4.2 we consider the services that we use for our attack, and we make the preliminary test for these services. In Section 4.3 we discuss our approach and architecture of our automated crawling attack. In Sections 4.4 and 4.5 we present the approach and process overview of manual crawling attacks on LinkedIn and Xing respectively.

4.1 Attacker Model

The attacker in this scenario is a malicious individual or group who aims to exploit vulnerabilities and gain unauthorized access to a system or data. The goals of an attacker can be different. For instance, he could try to distinguish real and fake email addresses to use them in Spam or Phishing attacks, or try to gain access to the private information of users. The attacker possesses a certain level of knowledge and skills related to hacking techniques, cybersecurity, and software vulnerabilities. An attacker who uses a crawling attack can be very difficult to recognize, as his actions resemble those of a normal user. Of course, for his attack, he needs certain conditions.

- Account Creation: The attacker can create accounts on the service using fake or normal credentials.
- Data Manipulation or Exfiltration: The attacker can upload, manipulate, or export contacts.
- Exploiting Contacts: The attacker may generate and upload a large number of fake contacts.

If all these requirements are met, then such a user can carry out an attack.

4.2 Establishing Scope

In the Section 2.6, we find out that Facebook Messenger, as well as job search services such as LinkedIn and Xing, are suitable for our purposes. Google Contacts also uses Email Discovery and due to the largest number of users, this service has a priority for us. Preliminary results of the experiments show that crawling attacks on Xing and LinkedIn job search services are possible, and we demonstrate the potential of such attacks. However, we develop a crawler to automate our attacks on Google Contacts, since this service has a huge number of users and preliminary results show that it is vulnerable to crawling.

4.3 Approach

Our proposed approach is a crawling attack that attempts to retrieve user data and verify email addresses. Our method involves generating random email addresses and storing them on a device, such as a computer or a smartphone, from which the attack will take place. Then we upload the contacts to the service provider server. By uploading the contacts with email addresses we generated to the server, we get confirmation, whether a user with a given email address exist, and thereby understand whether this address is real. The information obtained from crawling could be used for other attacks, such as phishing, identity theft, and spamming. The idea of our attack is based on the attacks carried out in the works of other researchers, who concentrated on instant messengers and phone numbers as personal identifiers [16] [64] [62].

4.3.1 Most Popular Usernames in Email Addresses

Due to a high entropy of email addresses, it is important to limit the number of possible address options in order to successfully conduct our experiments. In order to successfully create contacts with an email address as the main identifier, it was necessary to understand what structure of addresses people use most often. Usually, people use their first and last name in various combinations in their email addresses. Having studied articles that recommend different structures for email addresses, we can distinguish the main types that can be divided into several categories: Email addresses with first name only or last name only, email addresses with first and last name, email addresses with initials and names [66] [67].

For ease of demonstration, we will take the first name John, and the surname Smith in our example.

Email Addresses with First Name Only or Last Name Only

Email Addresses with First Name Only are not the most popular options, due to the large number of people with the same first name. The entropy in surnames is higher, and therefore the email addresses with only the surname are more popular.

- john@domain.topleveldomain
- smith@domain.topleveldomain

Email Addresses with First and Last Name

One of the most common options is to use both the first name and the last name in various combinations in the address.

- john.smith@domain.topleveldomain
- john_smith@domain.topleveldomain
- john-smith@domain.topleveldomain
- smith.john@domain.topleveldomain

Email Addresses with Initials and Names

It is also popular to use a combination of person's initials and name.

- j.smith@domain.topleveldomain
- jsmith@domain.topleveldomain
- j-smith@domain.topleveldomain

4.3.2 Usage of Names Database

Once we have decided on the type of email addresses we need, we need to develop a program that generates addresses with the aforementioned categories, and also turn them into contact lists in the format necessary for the service. Since all of the most popular email address types listed previously use people's first and last names, we decide to find the most popular German first names and surnames. This data is available in various sources, such as open databases, as well as statistics on the popularity of first and last names from different cities. In the future, the experiment can be continued with names from other countries, as well as using addresses that do not use first and last names in the username.

Last Names

We found the most popular German surnames in an open database [68]. This data was presented as a CSV file, which, in addition to the surnames, also contained information on the approximate number of people with a particular last name and the frequency given as a percentage. The surnames themselves were sorted in the descending order of prevalence.

First Names

We decided to take the first names from the database of the city of Cologne. This is one of the largest cities in Germany, which may well represent the most popular names [69]. The data was in CSV format and included the names, the year they became popular, the number of people with that name and the gender.

Email Domains

To generate email addresses, we use the most common domains in Germany [29]. By combining found first names, last names and domains, we can generate a large number of email addresses and achieve high entropy. The purpose of our study is to demonstrate the possibility of such attacks and also to evaluate the potential damage and scale. Of course, datasets can be expanded or replaced for other experiments.

4.3.3 Attack Overview

In this Section, we cover the general concepts of the crawling attack.

One can see an example of a full-fledged automatic attack in this Figure 4.1. The attack goes like this: first, contacts are generated on our PC by the program we developed. Further, our crawler reads files with contacts and uploads them to the service provider's server. After the contacts are recognized, crawler saves the found contacts and exports them. Then the crawler deletes uploaded contacts and loads new contacts.

To carry out such an attack, we need to determine the attack technique. Crawling attack can be carried out using native API or an emulated phone application, or a web version of the application. We also need accounts registered in the service, over which experiments are carried out.

The next step is to generate a large number of email addresses and create contacts from them in a suitable format for crawling. Next, after preparing the list of contact candidates, when uploading these contacts, we use the service's tools or built-in functions to search for contacts. Since the addresses are in the attacker's contact list, the service assumes that they are acquaintances of the attacker. By doing so, we can check whether these addresses are registered and obtain information about the users they are registered with.

This information can be the user's first and last name, a profile photo, as well as a date of birth and other information contained in a public account. We don't attack the selected

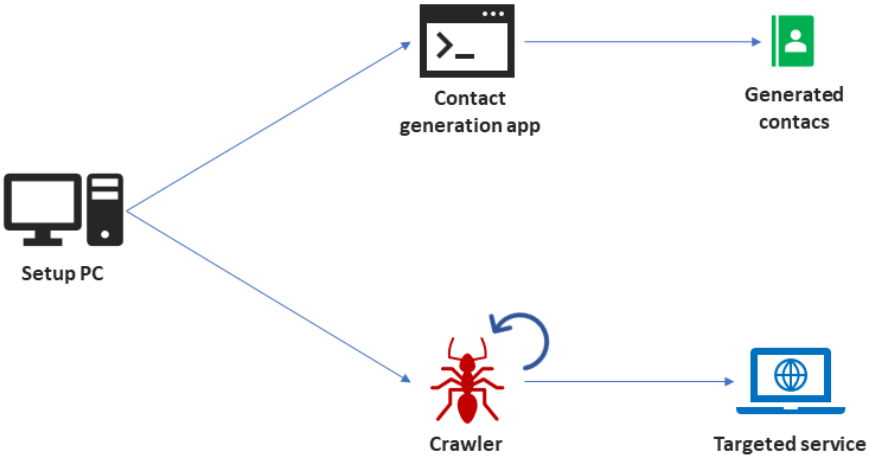

Figure 4.1: High-Level Architecture of Crawling Attack

service's infrastructure, but just use the built-in contact discovery tools. Therefore, each user registered with the service can potentially perform such an attack.

To prevent this type of attack, it is a common practice to enforce rate limits that restrict the number of searches a user can perform in a specific time period. In order to optimize the attack, it is essential to determine the rate limits and work within them.

To increase the efficiency of the attack, a crawler was developed for Google Contacts to automate the replacement of the phone book. The crawler can be designed to perform numerous searches, significantly increasing the number of email addresses that can be checked. By automating the process, we can save time and resources while increasing the success rate of the attack. Potentially, a huge number of email addresses can be checked this way. To carry out such an attack, the attacker needs to send requests to the service's server, and this can be done using the native API. However, accessing user data through such requests usually requires additional authorization and approval from the service, making this option unsuitable for the attacker. Some service providers provide access to API Calls related to user information only to developers authorized by this service. It should be noted that we develop the crawler only for Google Contacts, while for Xing and LinkedIn we only conducted attacks manually.

Thus, in general, the process of our attack is very similar to the interaction of an ordinary user with the service. The only difference is that our contacts were generated by us. We are also trying to repeat the initiation of the Email Discovery process in order to check as many contacts as possible. This is precisely the danger of such an attack, since every registered user can carry it out and use the information found for various needs. As we show later in this thesis, many users leave their privacy settings as default. Because of this, user data can be used by third parties.

4.4 LinkedIn Crawling Attack

We outline the general framework for conducting crawling experiments on the LinkedIn platform. To implement the attack on LinkedIn, we studied official data from the privacy pages [46], contacts synchronization [47], and also information about the API [70].

4.4.1 Attack Approach

One convenient way to carry out a crawling attack is to use the native API to send requests. Such a method could be easily automated for our purposes. To search for users using an API call, LinkedIn has a Profile API [71]. In some cases, services have strict requirements for accessing their APIs, and obtaining the necessary authorization and approval can be difficult or even impossible. For example, the LinkedIn API requires developers to apply for access and provide detailed information about the intended use of the API [70].

In light of these challenges, we decide to use another option to carry out the crawling attack in this scenario. Instead of using the API, we create multiple accounts on the service and send requests to verify email addresses and gather personal data. This approach is more labor-intensive, but it allows us to bypass the authorization and approval requirements for using the API.

To do this, we chose the native Android application LinkedIn, which uses Email Discovery as well. It is necessary to have access to a device or an emulator with the LinkedIn app installed. In that case, the sources of data in this system are the LinkedIn Mobile App and the LinkedIn database. The LinkedIn app contains user profiles and personal data, which are also stored in the LinkedIn database.

4.4.2 Understanding Rate Limits

To gain an understanding of the rate limits enforced by LinkedIn, a series of test experiments must be conducted. This allow us to determine the number of contacts that can be uploaded at once and the frequency at which the requests can be made. If the limits are low or we encounter any difficulties, we can create numerous accounts with fabricated email addresses to bypass any blocking. To gather the necessary data, we utilize third-party services that allow us to export the information we obtain. Once we have collected the data, it is vital to analyze and study the results of the experiments to understand how LinkedIn's platform works and optimize our crawling techniques accordingly.

Since there was no information about the various limits for contact discovery, we've decided to start Email Discovery with a small number of contacts and gradually increase the number. Difficulties with this strategy arose quickly, since when initiating the email discovery process, LinkedIn allowed one such operation to be carried out, and subsequent attempts quickly resulted in blocking this function for the user. We decided to take a break between attempts, and we empirically found that the next day it was possible to re-sync the contacts.

It is likely that the service does not expect that a large number of contacts can be changed in a short period of time. Related works authors came to similar results by conducting similar attacks on instant messengers. They found that, on average, the number of users increases by 0.5 percent per day [16].

In this regard, we were able to set the limit of permissible contacts per day. However, while conducting such experiments for a week, the accounts created for the experiments were blocked several times, and there was also no possibility to initiate an email discovery from this IP address. To bypass this limitation, we used a VPN service.

4.4.3 Attack Process Overview

As a result, the attack process went like this: firstly, various email addresses were created for registration in the LinkedIn service. When the accounts were created, contacts were generated in CSV-files, which were then converted to the vCard format. After that, these files were downloaded to the emulated device and the email discovery process was carried out from there. In this way we obtain the information, which of the generated contacts is registered in the service. Further, the checked contacts were deleted from the LinkedIn server and from the emulated device. The new contacts were then synced again after a certain amount of time. This process is demonstrated on the Figure 4.2.

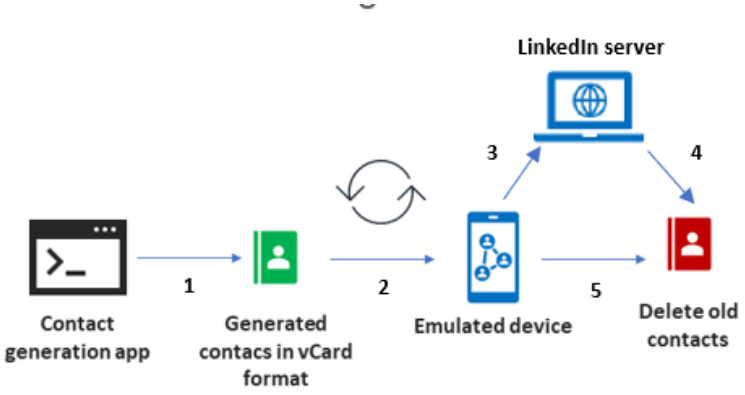

Figure 4.2: Architecture of Crawling Attack on LinkedIn

4.5 Xing Crawling Attack

Similar to LinkedIn, Xing is a professional social network that can also be used for our crawling experiments. The first step is to examine the platform and determine the most suitable tool for carrying out the attacks. To implement the attack on Xing, we also studied privacy pages and API capabilities. Unfortunately, limited capabilities without the rights of an authorized developer did not allow us to use the native API for attacks. There was also no official information about Rate Limits for the Contact Discovery process [72] [44] [43].

4.5.1 Attack Approach

Therefore, it was decided to simulate such an attack, as it was done with LinkedIn. To do this, we installed the Xing mobile application on an emulated phone. Just like with LinkedIn, test experiments need to be conducted to understand the rate limits enforced by Xing. This helps to determine the number of contacts used in a single cycle and the frequency of requests. If the limits are too low, or we encounter any difficulties, we can create multiple accounts with fabricated email addresses to bypass any blocking.

4.5.2 Understanding Rate Limits

When implementing the attack on Xing, we faced similar difficulties as when designing an attack on LinkedIn. An error occurred while resynchronizing contacts, and it was impossible to re-discover other contacts. It was also hard to design the attack because it was impossible to delete contacts that were synchronized earlier from the Xing server.

To solve this problem, we decided to use different accounts to conduct one email discovery process on a specific account. For a new set of contacts, a new account was created and utilized. This method complicates the possible automation of such an attack.

4.5.3 Attack Process Overview

Our attack process involved creating various email addresses for registration on Xing. Once the accounts were created, we generated contacts in CSV format, which were then converted to vCard format. These files were downloaded to the emulated device, and the email discovery process was carried out from there. Then we could obtain the information, which contact was really registered with Xing. Then we created a new account to check another set of contacts, because deleting synchronized contacts from the Xing server was not possible. After a certain amount of time, the new contacts were synced again with a new user account. It was also not possible to synchronize contacts more than twice because of the rate limit.

5 Implementation

In this Chapter we present the specific implementation of the contact generation app and the crawler we developed, starting with the generation of contacts in the Section 5.1. We then discuss the structure, and all the necessary tools for the attack in 5.2. In Section 5.3 we present our crawler in detail and discuss all methods of it, and difficulties we faced while developing the crawler.

5.1 Contact Generation

In this Section, we take a detailed look at how the developing the application for generating contacts went. For this purpose, Java version 17 was chosen from an open-source implementation of the Java Platform OpenJDK [73]. The Java language was chosen due to the ease of use of the built-in libraries from java.io, which allow reading a file from a CSV, as well as creating such files. In particular, we used java.io.BufferedReader to read and java.io.PrintWriter to write to a file. No third party libraries were used.

The program was supposed to perform the following functions:

- Accept CSV-files containing names and surnames, the number of generated contacts, domain, separator character
- Divide the generated contacts into a certain number of files in the CSV format, which should be loaded into applications
- Contact generation, in the format First name, Last name, Email address

Since some first and last names in German use umlauts, we replace them with the correct display in the ASCII format. As a result, the program was written in such a way as to create a contact from each first and last name pair and write it to a CSV-file with following fields: first name, last name, email address. It is possible to change the separator or domain. For more flexibility, we left the ability to adjust the number of contacts in one file and the number of files with contacts.

5.2 Google Contacts Crawling Attack

In this Section, we describe the implementation of the attack on Google Contacts, it's detailed structure, and all the necessary tools. In the Figure 5.1 one can see the general

structure of the attack. On our test computer, we generate contacts in CSV format, then we save these files with thousands of contacts each to a separate folder. Then the crawler we developed uses these files and uploads them to the service and performs the necessary operations, such as importing and exporting contacts, as well as deleting previous contacts.

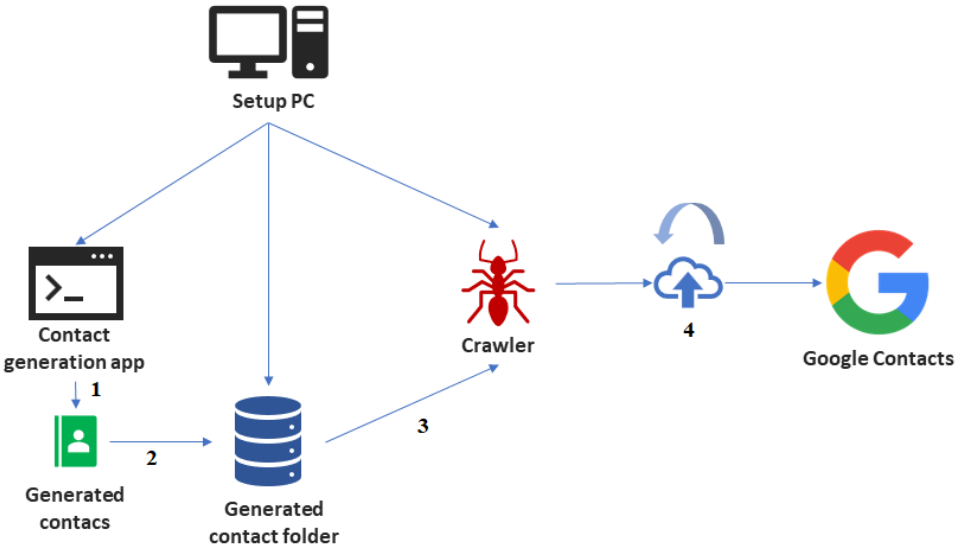

Figure 5.1: Google Contacts Crawling Attack

In preparation for designing the attack, Google's official documents about this service were studied. In particular, pages related to the protection of personal data, the functionality of the service were studied, and official information was found about its limitations for the user [38] [50] [52] [54]. The main target of our attacks is the web version of the Google Contacts application. We choose the web version for the ease of experimentation. In this case, we do not need an emulator of various devices.

5.3 Crawler development

For the greater efficiency of our attacks, we developed a crawler. We decided to use a web crawler because it did not require emulation, as is the case with the Android Emulator. Due to this, we could achieve higher performance and stable operation. Typically, a web crawler is used when it is necessary to scan a page and extract some information from it. It can be email addresses or other information. However, in our case, we develop a crawler that perform the planned attack in a step-by-step manner.

So, now we analyze a preliminary list of the necessary crawler functions for successful attacks. Firstly, the creation of accounts for experiments take place manually and as needed. The use of multiple accounts may be necessary if the previous ones are blocked by the service, and also if the attacker wants to scale up the attack.

5.3.1 Our Setup

As in the experiments with LinkedIn and Xing, discussed in Sections 4.4 and 4.5 respectively, a regular personal computer running the Windows 11 operating system was used. Before the development of the crawler, we review and analyze possible tools to implement the functionality we need and settle on the Selenium service.

Selenium is an open-source tool that automates web browsers [74]. It provides a single interface that lets you write scripts in programming languages like Ruby, Java, NodeJS,

PHP, Perl, Python, and others. We use the functionality of this service to create a crawler with all the necessary functionality for us. In our case, we choose the Python 3 language, in which we use the standard libraries, as well as the libraries of the Selenium service. The script we develop in the PyCharm IDE developed by JetBrains [75].

5.3.2 Developing

To use Selenium, we need to specify a driver. Driver is an executable module that opens up a browser instance and runs the test script. It is browser-specific, for example Google develops and maintains Chromedriver for Selenium to support automation on Chromium/Chrome. While working, we also use two versions (Chrome 110 and Chrome 112) of the Chrome web browser developed by Google [76], for which we use the appropriate driver for the Selenium service.

At the start, we study the interface of the application and identified the most important interface elements that we interact with: 1.Import, 2.Export, 3.Trash. 5.2

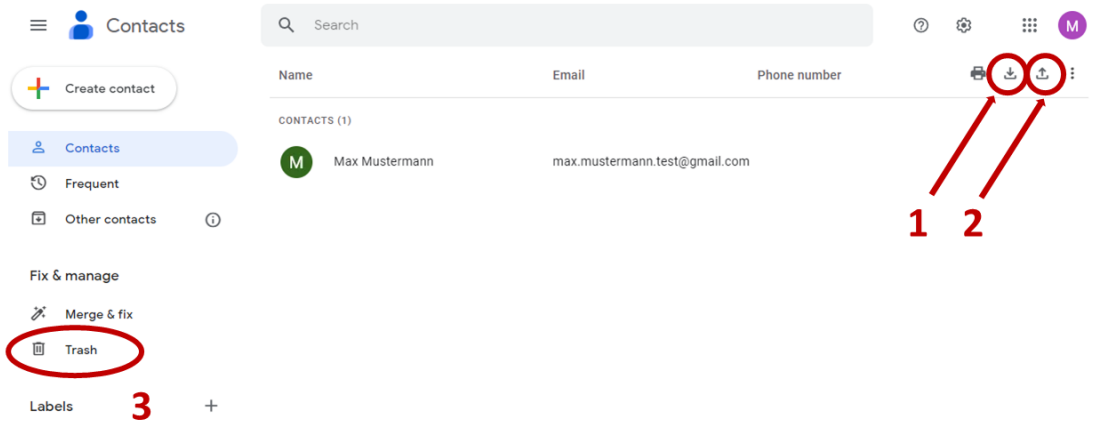

Figure 5.2: Google Contacts Interface

The requirements for the crawler developed for Google Contacts include:

1. Account Access: Login in Google Service. If necessary, bypass built-in protection (checks)
2. Import Functionality: Import contacts through the built-in import function of the service
3. Export Capability: Export found accounts through the built-in export function
4. Contact Deletion: Delete downloaded contacts and empty the contacts trash can
5. Contact Change: Replace (Iterate) the contact list

With these features, we are able to fully automate the crawling process.

In Figure 5.2 one can see the main page of the application, from which the work of the crawler begins. After logging in, we can see this page. On this page, we have to go to the import contacts tab and upload the first file. After waiting for the download, we get the

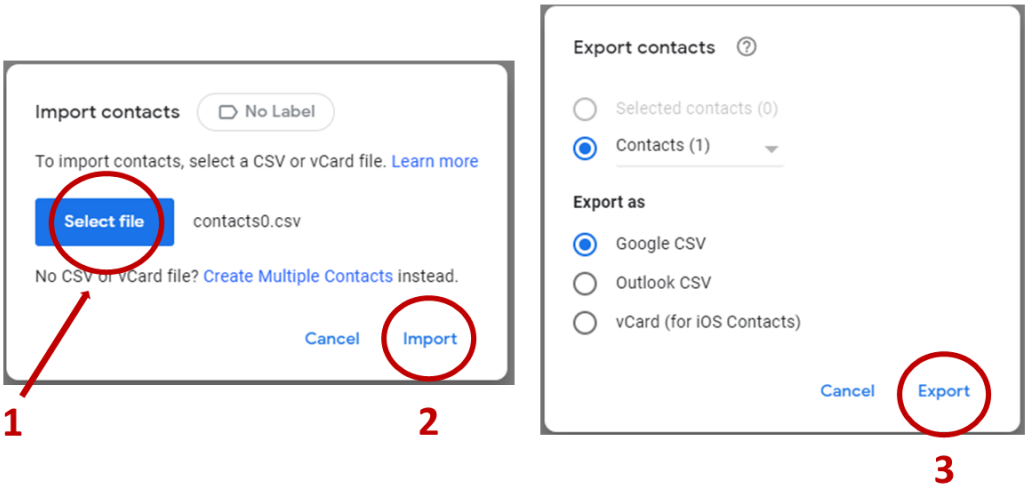

Figure 5.3: Google Contacts Import/Export Interface
1. Select file, 2. Import, 3. Export

result and see the number of contacts that the service recognizes as registered. After that, we can go to the export tab and save the found contacts in the CSV format.

In Figure 5.3, one can see the interface of the import and export functions with which the developed crawler interacts. To write a script, search for elements on the page, we use the built-in developer tools in the Google Chrome browser.

Our crawler consists of 6 methods: login, do_import, do_export, do_remove, clean_trash and a subroutine click_button_by_word.

login. This method allows us to log in to our account by entering our email address and password in the required fields, then submitting the login form.

do_import. This method uploads the CSV file with generated contacts to Google Contacts service. Crawler activates the Import button. Then the first form, depicted in Figure 5.3. The crawler finds the input div, and uploads the CSV file. Crawler then activates the Import button in pop-up menu, uploading the file to the server and starting the Email Discovery process. The subroutine **click_button_by_word** finds and activates the needed buttons. We demonstrate the code of method and subroutine:

```
def do_import(driver, source):
    try:
        buttonImport = click_button_by_word(driver, "Import")
        upload_input =
        driver.find_element(By.XPATH, "//input[@type='file']")
        upload_input.send_keys(source)
        time.sleep(2)
        buttonImport = click_button_by_word(driver, "Import")
    except:
        print("No such elements")
```

```
def click_button_by_word(driver, word):
    buttons = driver.find_elements('xpath', '//button')
    for button in buttons:
        try:
            match = re.search(r'(<span[^>]*>.*?' +
            word + '.*?<\/span>)',
            button.get_attribute("innerHTML"))
            if match:
                button.click()
                time.sleep(2)
                return True
        except:
            print(f"Match {word} Button Error")
    print(f"No {word} Button found")
    return False
```

do_export. After Email Discovery process this method exports the CSV file with found contacts;

do_remove. This method selects all contacts and moves them in the trash;

clean_trash. This method removes all contacts from trash bin.

5.3.3 Difficulties in development

During the development of our script, we faced the challenge of dealing with CAPTCHA, specifically the reCAPTCHA developed by Google [77].

Our script encountered reCAPTCHA in two cases: When attempting to log in and when attempting to delete contacts. These are reasonable scenarios to use protection because they involve personal data. During our experiments, we found out that reCAPTCHA only appeared when logging in with connection to the VPN service. If reCAPTCHA appeared, we completed it and restart the crawler. As we proceed with our work using different IP addresses, we discovered that this protective measure did not appear anymore.

Developing a web crawler to extract data from Google Contacts proved to be challenging due to the difficulty of identifying elements on the page. Google Contacts uses a framework for its front-end, which means that elements in the developer console do not have unique classes or identifiers. As a result, it was necessary to use regular expressions (RegEx) to scan certain HTML elements and search for keywords to identify elements.

The development of the crawler had to be carried out twice because the web interface of Google Contacts was updated during the development process, causing significant changes to the elements and design, which made the previous version of the crawler obsolete. This serves as a cautionary reminder that crawler relying on web page elements can become useless after an update. As a result, attackers should approach such scripts with caution, especially if the service updates frequently.

Additionally, due to the complex identification of objects on the page, certain crawler features that increased reliability were difficult to implement. For instance, to wait for contacts to load before performing certain actions, we use a timer, which could potentially cause the crawler to perform the next action before waiting for the result. This could happen because of unstable network connections or problems with loading on the service side. To avoid such situations, the crawler needs to be improved, especially when a simple wait is necessary to ensure that actions are executed only after the results are available.

6 Evaluation

In this Chapter, we demonstrate our evaluation approach of the experiments we carried out. In Section 6.1 we discuss the metrics used for evaluation of results. In Section 6.2 we discuss in detail, how we have generated contacts for our tests. In Section 6.3 we present the evaluation setup of our manual attacks on LinkedIn and Xing. In Section 6.4 we evaluate the results of the attack on LinkedIn. The results of the attack on Xing are discussed in Section 6.5. Next, we present the results of the automated attack on Google Contacts in Section 6.6. Finally, various methods of protecting the Email Discovery process are also discussed in Section 6.7. Each service has unique Rate Limits, protection methods, as well as the amount of information available about users, so experiments were carried out for each of the services with different parameters, settings and setups.

6.1 Analysis of the Effectiveness of the Experiment

We want to assess the effectiveness of the attack by analyzing the success rate of obtaining personal data, such as email addresses, from the targeted service. We measure the success rate of data collection for a given number of requests or queries made by the attack. To determine the success rate, we divide the number of successful requests by the total number of requests made and express the result as a percentage. To accomplish this, we can design experiments to simulate the attack and gather data on the success or failure of the attack.

In this Chapter, we present detailed results of our experiments with the crawler. Overall, generating a large number of contacts and dividing them into manageable CSV files allowed us to efficiently test the crawler's capabilities and identify any issues that needed to be addressed. Through careful experimentation and testing, we were able to develop an effective and reliable crawler for working with Google Contacts.

6.2 Contact Generation

As already mentioned in the Chapter 5, we made the decision to use the 200 most popular German surnames and thousands of German names. This means that just using our data, domains and various email address structures we can generate about 80 million contacts. However, our goal is only to demonstrate the possibility of crawling. We decided to choose different options for domains and email structures, in order to analyze which contacts are suitable for experiments, which domains are more popular in certain services, and with what frequency we can find a contact with the selected structure.

As a result, we settled on two domains: @gmail.com and @gmx.de. These domains are very popular in Germany. We also use the following email constructions. For ease of representation of names, in this example, the first name will be John and the last name will be Smith:

- john.smith@gmail.com
- john.smith@gmx.de
- john-smith@gmail.com
- john-smith@gmx.de
- j.smith@gmail.com
- j.smith@gmx.de
- smith.john@gmail.com
- smith.john@gmx.de

In our experiment, we identify the optimal designs for analysis and drawing conclusions. However, it is important to acknowledge that attackers can employ a different approach. For instance, they may utilize techniques employed by Hu et al. [65]. The authors explored services offering temporary email address generation. They obtained leaked databases from platforms like LinkedIn, Myspace, Zoosk, Last.fm, and Gmail. In their attacks, they used emails and phones from leaks, since these accounts definitely existed and were registered in the services. By incorporating the most popular email services or mimicking their designs, attackers could potentially achieve even more favorable outcomes. However, we want to prove, that even without knowing in advance, which accounts definitely exist, the attack can also be efficient.

To carry out a successful attack, we need to know all the restrictions, limits and defense mechanisms. Our next step in preparing for the experiments is the designing of trial tests. These tests are necessary to identify all the built-in restrictions, possible protection methods and determine the base rate limits. Also, these tests are the basis for the development of the crawler. Trial tests consist of generating a set of contacts of different sizes and determining the optimal number for the subsequent operation of the crawler. Then we upload the contacts to the service and analyze the data we find. Also, these tests were able to demonstrate the various complexities that must be taken into account when developing a crawler.

6.3 Setup for LinkedIn and Xing Attacks

The setup used for carrying out attacks on LinkedIn and Xing involved a mid-end computer running Windows 11 with Android Studio installed. Android Studio is a free IDE developed by Google that was chosen for its ease of use of the Android Emulator for development. [78].

An Android emulator was used to emulate the normal operation of a mobile application and to access the Android file system for uploading contacts. The Google Pixel smartphone emulator with API version 30 running the Android 11 operating system with a pre-installed Play Market application store was selected for this purpose. Since we choose a mobile application for experiments with the LinkedIn and Xing services, it became necessary to generate contacts in the vCard format.

The built-in Java libraries did not allow creating contacts in this format. However, as our primary objective is to focus on the Google Contacts service rather than to carry out a widespread attack, our intention is to examine the vulnerability of services to this particular type of attack. Consequently, we have decided to utilize the built-in contact converter provided by the Windows operating system. This contact converter is a part of the Microsoft People application. This application was introduced in 2015 and is a native contact manager for the Windows OS. With the help of this converter, we could transfer the required number of contacts (9000 for LinkedIn, and 3500 for Xing) from CSV to vCard format [79]. These contacts were ready to be used in the Android Emulator.

To bypass a potential account blocking, it was necessary to have a supply of email addresses to register new user accounts. For these purposes, we used the free Proton email service, since it did not require confirmation using a mobile phone [80]. Additionally, a VPN service from third-party developers was installed on the computer to bypass potential IP address blocking.

6.4 LinkedIn

In this Section, we discuss the results of the attack on LinkedIn. We also discuss how many real email addresses or other personal data from the targeted service we obtain.

6.4.1 Experiment Limitations

During our experimentation with the LinkedIn service, we utilized 10 different accounts and conducted a thorough analysis of 9,000 contacts in total. Unfortunately, this number is a bit smaller than we had hoped, as we encountered certain limitations and challenges throughout the process. Specifically, we faced difficulties conducting the Email Discovery process multiple times on a single account without risking the account being blocked or restricted, as well as the IP address being blocked.

However, we have some success with one account after a 24 hour wait, but we were not able to replicate this success on subsequent attempts. As a result, we had to create new accounts and use different IP addresses for further testing.

We also found out that the maximum number of contacts that can be discovered is 1000 pieces. In order for the Email Discovery process to be successful and not overcome the maximum possible number of conflicts, we uploaded only 900 contacts. We assume that LinkedIn is using a leaky bucket structure. Therefore, our attempts to discover more than 1000 contacts per day failed. Probably, when this number was reached, a timer was set and access to the discovery function was reopened after 24 hours. Attempts to use phone books with a small number of contacts and trying to download several such phone books were also not successful. In order to be able to download another 1000 contacts the next day, we make the following steps. In the application settings, we go to the privacy item and turn off the Contact Discovery function. After that, it was possible to delete already discovered contacts. Only after that, it was possible to start the synchronization process again.

6.4.2 Experiment Results

Using a phone book with both domains and the firstname.lastname@topleveldomain structure, we were able to find over 10 runs (10 days) an average of about 150 contacts out of 900 pieces. This translates to a success rate of 17%, which is quite promising.

When we used the lastname.firstname@topleveldomain construct, we only found an average over 5 runs of 25 contacts, corresponding to a mere 2-3% of the number of contacts in the phone book.

Finally, we test the email construction with the first letter of the first and last name and found an average over 5 runs of 50 contacts, or 5.5%. As we can see, the most popular email construction for LinkedIn contacts is first name.last name@topleveldomain.

Although we were not completely satisfied with our results, we were impressed with the wealth of personal information available on the LinkedIn platform. We found that a large percentage of accounts contained real, detailed, and personal information about the user. In one of the iterations, we received 154 contacts out of 900 generated. Among them, we found 41 contacts with a photo and all of these photos were portrait photos, which equals 26.62%. This is in strong contrast to other social networks, where profile pictures can be anything.

6.4.3 Data Richness

We found that many profiles were open, which means that it was possible to establish contact with the found users, and we could also view the personal information that the user stored in the service. This allowed us to discover the user's education and place of work, as well as their age. While we may not have been able to create a crawler for this service, the richness of the data available on LinkedIn is certainly noteworthy.

As a conclusion, it can be said that due to the built-in limit on the number of contacts that can be downloaded per day such an attack was mitigated. Also, the blocking or partial restriction of the functionality of the Email Discovery function indicates a built-in protection system against such attacks. But a possible attack vector by creating a large number of accounts using multiple devices is still possible and can be exploited by attackers in future work. Also, as I mentioned earlier, it was possible to delete contacts from LinkedIn and, after a while, run the Email Discovery process again.

6.5 Xing

In this Section, we discuss the results of the attack on Xing. We also discuss how many real email addresses we obtain and which personal data from the targeted service we found.

6.5.1 Experiment Limitation

During our experiments with the Xing service, we utilized 4 accounts to investigate around 3.5 thousand contacts. However, due to low Rate Limits and suboptimal Email Discovery results, we were restricted to 1.5 thousand contacts per account.

In this service, we could not delete already downloaded contacts and thereby reset the found contacts. This led to synchronization limits in a leaky bucket structure when we wanted to upload another set of contacts. Those previously synchronized contacts still remained in the Xing database. Therefore, we had to use different accounts in order to carry out the next synchronization. Thus, we found that after two synchronizations, when there were more than 1500 contacts, an error was thrown and Rate Limits did not allow synchronization. However, when the account had 1500 contacts, it was impossible to carry out the next synchronization even if you wait a day before it. To upload other contacts, a new account was needed.

It is also important to note that when using the Android Emulator and after subsequent discovery attempts failed, it was not possible to create a new contact from this device and we received a network error. To get around this, we registered all subsequent contacts through the web version of the Xing service and then used them on our emulated device.

6.5.2 Experiment Results

We terminated our experiments due to unsatisfactory discovery results. Finding users was the most difficult aspect of the investigation given the service's 20 million users. Our findings indicate that with 3500 contacts with the most popular structure and the most popular German domain, i.e., first name.last name@gmx.de we were able to obtain only 28 matches, 0.8% of the checked contacts.

These outcomes may be satisfactory for an attacker pursuing a specific objective. Consequently, with a plethora of accounts and several devices, it might be possible to achieve as good results, as with LinkedIn.

6.5.3 Data Richness

Despite Xing's positioning as a service that values users' personal information, many profiles are open, and we were able to determine that only 9 of 28 users, 32.14% of users uploaded a portrait photo to their profile, and by 23 of 28 users, 82.14% current workplace was visible. Similar to LinkedIn, the service retains a significant amount of personal information, making it an attractive target for an attack.

6.6 Google Contacts

In this Section, we discuss the results of our main experiment with the main service - Google Contacts. Firstly, we discuss the set-up details of our experiment. Then we describe the limitations and results of the experiment.

6.6.1 Set-up Details

The computer on which the attacks were carried out is running Windows 11 and has 16 GB of RAM, as well as an AMD Ryzen 2600 processor, with an AMD RX 5700xt graphic card.

For our tests, we used one account and the second account was used directly for data crawling using the developed crawler. A Google Services account can be created without any connection to a mobile phone or other additional security measures. To bypass possible blocking of IP addresses, we used VPN.

We need to generate contacts with different domains and different test constructs. We decided to use only one email construction in one file, as this would make the process of evaluating easier.

6.6.2 Experiment Information and Limitation

We generated 9 CSV files of 11902 pieces for each type of 8 aforementioned combinations of email and domain. As a result, our starting set consisted of 856944 contacts.

Each file contained 11902 contacts, which is 47,61% of the maximal allowed number of contacts in an account. According to the official documentation [38], the maximal allowed number of one user's contacts is 25000. Furthermore, maximal number of contacts for one upload is 12000.

In the process, having identified the most successful combinations, we continued the generation and reached 1452044 contacts (122 CSV files) in total. However, the first difficulty we encountered was that Google Contacts only discovered contacts with a domain from Google. Any attempts to download contacts with the gmx.de domain ended with the fact that absolutely all contacts were displayed in the "My contacts" menu-item. However,

nothing could be done with this list, because these contacts were not actually recognized by Google Contacts.

It was because of this that the total number of contacts tested was reduced from 1452044 to 1130690 (from 122 CSV to 95 CSV), since all contacts with other domains had to be removed from the experiments.

Next, the found contacts were re-uploaded to the server to evaluate what data about these accounts was available. Thus, having tested the first 300 thousand contacts, we could already understand in advance which email structures is more popular and what data we can extract.

After this preliminary test, we started a larger experiment. Therefore we have decided to generate more contacts with two most popular email structures: john.smith@gmail.com and john-smith@gmail.com. We chose the domain gmail.com because of the result of preliminary test, and generated 256 files with 11902 each. One can see the exact amount of generated contacts with each structure in Table 6.1. This experiment lasted 48 hours (from 11.06 to 12.06) and as a result, we were able to check 3046656 contacts. Crawling attacks were built as follows: each time a file with the .csv extension was loaded. The email type and construction were different for each file. After that, the accounts found by Google Contacts were exported and removed from the service, including emptying the trash.

During the work of the crawler, we encountered its limitations. During one run, work was interrupted due to an unstable connection, and during the next login, Google requested to enter the phone number to confirm our identity. Using Selenium, we were not able to read data from the Pop-Up Menu, which opened in a completely new browser window. Thus, the experiment was interrupted due to the need for additional verification, when all contacts were deleted from the basket. However, by entering a password, the experiment could be continued. Therefore, the work of the crawler was not stopped, but only the delete action was blocked, which was soon available again. However, at night, when it was not possible to control the work of the crawler, an interruption was recorded, since the number of valid contacts in the basket exceeded 25 thousand. In order to continue working from the same place after the interruption, we saved the index of the last downloaded file in the roofer. In this way, we could always continue the work of the crawler. One possible solution to the problem with additional verification is to use other tools that allow you to read data in new open browser windows.

6.6.3 Experiment Results

Now we present the data from two experiments. In the main experiment with 3046656 contacts, we found, that out of 1574872 contacts there were 368928 real contacts with a similar structure: john-smith@gmail.com. Out of the remaining 1471784 contacts, 319814 real ones had a following form: john.smith@gmail.com. The results of the preliminary manual test for two less popular structures (j.smith@gmail.com and smith.john@gmail.com) can also be found in the Table. We tested 10 CSV files (119020 contacts) for each structure. Detailed data are shown in Table 6.1. As can be seen from the results, the most popular constructions use the first name at the beginning and the last name at the end. In general, this result we expected. Unexpected for us is that the construction with the dash separator is more popular than the dot separator.

The absence of Rate Limits during both experiments is also worth noting. In both cases, one account was used, which sent requests throughout the day and did not meet any restrictions. The only restrictions concerned the maximum number of contacts that could be uploaded at one time, namely 12 thousand contacts, as well as the maximum number of

Table 6.1: Experiments result

Email Constructions	Tested Contacts	Found Contacts	Success Rate
john.smith@gmail.com	1471784	319814	21.72%
john-smith@gmail.com	1574872	368928	23.42%
j.smith@gmail.com	119020	17627	14.81%
smith.john@gmail.com	119020	7333	6.16%

contacts that could be in one account - 25 thousand contacts. It is also worth noting that since we used a standard Mid-End PC with a VPN connection for the experiment, and the pauses between all operations like input, export, remove etc. in the crawler were quite large, this reduced the total number of checked contacts. These pauses were necessary for the stable operation of the crawler, but we guess they could be much lower and thus increase the number of discovered contacts by up to 25-50 percent. It is also possible to use the crawler developed by us on several PCs and accounts in parallel in order to achieve even better results.

6.6.4 Data Richness

You can see the main results showing the number of accounts tested and found, as well as the success rate, in Table 6.1. As for the personal data found, the results of attack on Google Contacts were weaker than those of attacks on LinkedIn or Xing. The contact search service uses Gmail integration, and therefore many accounts do not use personal photos, as well as profile descriptions.

However, exactly because of the deep integration of users into Google Services, we can determine in real time whether a person is online in services such as Google Chat or Google Meet, which can help to build a social graph of the user. One can also send a phishing or a spam message directly to the users using these services. It is advantageous for the attacker, because these services do not provide spam protection, unlike email providers [4].

In order to get and count the number of contacts with photos, we used network monitoring. To do this, when loading the found contacts into Google Contacts, we opened the developer panel in the browser and in the source section we were able to find all the pictures. In the source, we found a folder in which there were two more folders with different names: ”cm” and ”a-”.

In the folder with the name ”a-” were pictures of users, and in the folder with the name ”cm” were Google placeholders. Thus, by scrolling the Google Contacts page to the very bottom, we received a folder with all the images. Then we downloaded this folder with the .har extension, and counted the number of images on the command line. As a result, we found that from 688742 real users with two most popular email address structures 907 users (0.132 % from 688742) have a profile photo. These photos can be either an abstract art or a real photo of the user. And of course, due to the huge number of users, this service

is an excellent target for attackers who want to get user data. It is also possible to send phishing emails to found users directly using these services.

6.7 Possible Protection Methods

In this Section, we will discuss possible options for protecting the Email Discovery process from crawling and other server attacks. To do this, we will consider the most modern versions of the PSI (Private Set Intersection) protocols, as well as the optimization of Rate Limits, to prevent crawling attacks.

6.7.1 PSI Protocols

When using the PSI protocol, the Email Discovery process can be protected from the compilation of the user's social graph by the service provider. To date, there are no efficient protocols that can be used in applications with a huge number of users, since this protocol requires a lot of computing and bandwidth power [13]. Also, these protocols do not protect applications from crawling attacks, and developers need to think about implementing the correct Rate Limits to ensure proper security. For instance, if one would consider applying a PSI protocol for 2 billion of WhatsApp users, the users would need to download about 8 GB of data to their device, as shown by Hoepman [61]. Even if we consider PSI protocols that are optimized for unbalanced sets, matching a thousand contacts against a quarter billion registered users still takes three seconds.

Alternative PSI protocols, such as [14], impose a higher privacy standard. In these protocols, the sender (server) can compel the receiver (user's mobile device) to produce an output containing all the elements in its set, without requiring the sender to guess the set's members held by the mobile device. However, these protocols are unsuitable against crawling attacks.

Mutual Contact Discovery The Mutual Contact Discovery protocol was proposed by Hoepman et al. in 2022 [61] as one of the options to solve the aforementioned problem of the PSI protocol. Mutual Contact Discovery refers to a process that enables users within a social network to find each other when they join a new service. In this context, Alice and Bob are considered mutual contacts if Alice has Bob's contact information and Bob has Alice's contact information. The social graph, which represents the relationships between users, is based on their contact lists stored on their mobile phones. The contact lists consist of mobile phone numbers as identifiers.

The protocol for mutual contact discovery should fulfill these requirements:

- When Alice and Bob are mutual contacts, both use the messaging service, and run the mutual contact discovery protocol simultaneously, they will discover each other.
- If Bob is not a member of the social network, then Alice cannot be forced to believe that he is.
- If the adversary is not a contact of Alice, then he or she will not be able to learn whether Alice is a member.
- An adversary is not able to tell whether Bob is a contact of Alice, provided Alice and Bob are both private.

However, the proposed model has many shortcomings and so far it is more of a theoretical solution to the problem. The aforementioned protocol has not been implemented even as a prototype, which means there are no concrete results or benchmarks to evaluate its performance or effectiveness. Without actual implementation and testing, it's difficult to

evaluate its real-world viability. The protocol draws a comparison between Mutual Contact Discovery and private set intersection. However, due to the nature of the sets involved in Mutual Contact Discovery (concatenation of identifiers), the intersection usually contains at most one element. This unique feature makes applying Private Set Intersection less appealing for Mutual Contact Discovery, especially when multiple members need to compute the intersection. However, the protocol notes that these services typically rely on public keys instead of phone numbers as identifiers. While this makes the aforementioned problem easier to solve, it introduces the challenge of how members obtain and exchange public keys, potentially requiring physical meetings, which undermines the concept of Contact Discovery.

6.7.2 Rate Limits

As already mentioned, there are no such PSI protocols today, that could be applied in applications such as Google Contacts. Moreover, even if the use of such a protocol were possible, the vulnerability against the crawling attacks in the Email Discovery process would remain. One option to mitigate the damage of such attacks is to introduce limits on the number of requests, and we would propose to apply such limits to protect Email Discovery process in Google Contacts.

Incremental Contact Discovery

The approach proposed by Hagen et al. [15] [18] suggests that an average user, when discovering contacts for the first time, does not completely change the phone book. Typically, no more than 0.5 percent of the total number of users joins per day. Thus, a limit of the number of new users with an acceptable margin of error can be set. If the user completely changes his phone book, he will quickly exhaust the available number of contacts for synchronization. Therefore one could control the process of Contact Discovery and Email Discovery.

Also, this synchronization can be limited in time so as not to worsen the user experience. Thus, one can leave the limit on the number of new synchronized users only for those users who repeatedly initiate the Contact or Email Discovery process on the same day. If the user has been offline for a long time, then he will be able to complete the full synchronization of contacts again.

This approach cannot completely prevent above-mentioned attacks, but it can significantly reduce their effectiveness without worsting the user experience. This approach may suit applications such as LinkedIn, Xing, and instant messengers. However, if one leaves a restriction of the number of new contacts for a contact service like Google Contacts, this will lead to a worse user experience. Therefore, other measures are necessary for the security of such a service.

Differential Contact Discovery

In the work of Hagen et al. [15] [18], another approach to Rate Limits has been proposed. This scheme provides a solution to crawling attacks conducted by adversaries who are registered as users. It allows the service provider to limit the changes made to the client's contacts without storing any user contacts on the server. The scheme is efficient in terms of storage, computation, and transmission size. Additionally, it is designed to be used in conjunction with Signal's Intel SGX API [81], offering reasonable privacy for users' social graphs. It is also potentially possible to use the PSI protocol in combination with this approach.

6.7.3 Other Mitigations

One approach that can be used for Google Contacts with Email Discovery are additional verification methods such as CAPTCHA or additional account verification methods.

The scale of enumeration attacks depends on two main factors: The rate limits that determine the crawling rate for individual accounts and the number of accessible accounts for the attacker. Solutions like ICD or DCD can be optimized for this use case and lowering Rate Limits could be a mitigation. However, here one needs to be careful not to harm the user experience, as mentioned before, which allows one to manage a large number of contacts.

Extended Verification Process

Many enumeration attacks can only be conducted on a large scale if attackers can easily register multiple accounts. Implementing extended verification processes, such as identity checks during registration, can make it more difficult for attackers to launch large-scale attacks. For example, Google Contacts could request additional data for authorization to protect the account during the attack. However, it was enough to close this window or refuse to check identity to continue the attack. One possible solution would be to force additional verification.

CAPTCHA

CAPTCHA could also weaken the effectiveness of attacks if it popped up in case of suspicious behavior. For example, when a certain limit of contacts is reached. However, as Bilge et al. [62] already shown, CAPTCHA can be bypassed, but this does not mean that it can be excluded as an additional security measure. It would make the implementation of an attack harder, because one also would need to implement a CAPTCHA bypass.

Data Privacy

One of the most important ways to prevent data leakage is to make that data unavailable to attackers. This can be achieved by not allowing access to personal data, even if the contact was in the user's address book. For example, in our work, we have shown that knowing only the email address, we can get a personal information about users of Google Contacts (cf.Section 6.6.4). One option to make such an attack less attractive is the lack of open personal data by default.

6.8 Ethical Considerations and Responsible Disclosure

Crawling attacks on Google Contacts, Xing and LinkedIn can pose a significant risk to both individuals and organizations. These types of attacks can violate privacy and security, and it is crucial to take the ethical implications of such actions into account.

6.8.1 Ethical Considerations

Crawling attacks on Google Contacts, Xing and LinkedIn raise ethical questions regarding privacy and data protection. Individuals and organizations have the right to protect their personal information and expect it to be treated responsibly and securely.

Conducting such attacks without consent is both unethical and illegal. Therefore, it is essential to carry out such attacks ethically and legally. Furthermore, transparency is crucial in how personal information is handled, collected, and used.

When developing our experiments, we did not have the goal of disrupting the service, slowing down or affecting the quality of the service. We designed our experiments in comply

with the European General Data Protection Regulation [82]. We focused on the principle of data minimization (Article 5c) and used obtained information only for scientific purposes (Article 89). It was also not intended to use public data obtained during the experiment. All information found and received during the experiments, including personal information, is used exclusively locally to evaluate the work performance. No personal data is stored anywhere. When working with such data, data anonymization was carried out, which consisted in leaving only the amount of data received and its type. The experiment was also based on similar experiments conducted by other works before and is fully ethical [16].

6.8.2 Responsible Disclosure

We also sent all the necessary information about our attack to Google and also described and presented the results of our attacks. We indicated in what way and by what means it is possible to carry out the attack. At the time of writing, we have not received any feedback yet.

7 Conclusion

The purpose of this bachelor's thesis is to study the vulnerabilities of the Email Discovery process. The main idea of the work is to perform an enumeration and crawling attacks on the found applications in order to check the vulnerability of the Email Discovery process and specific applications to a similar type of attacks.

To do this, we select and study many messengers, job search applications and gaming platforms. During our research, after conducting preliminary tests, we choose three applications using Email Discovery. These are two job search platforms Xing and LinkedIn, as well as the contact management application Google Contacts. These applications turn out to be vulnerable to crawling attacks.

To design the experiment, we need to generate a set of email addresses. Due to the high entropy of the email address, we decide to generate only the most common address constructs, and also use databases of the most popular German names and surnames to generate contacts. Subsequently, these contacts are loaded into the examined application in order to find which of these addresses are registered in this service. Thus, we verify the addresses, find additional information available about the user and calculate the Rate Limits by the number of possible downloaded contacts.

After conducting preliminary experiments, we find that the Google Contacts service allows to download a large number of contacts at a time and has no limits. Therefore, we developed a crawler for this service that automatically carried out a crawling attack. As a result, we are able to check over three million different generated contacts and got a fairly high percentage of hits. For example, the email address construction with dash as a separator between the first and the last name are the most popular. Out of 1574872 million tested contacts with this construction, we verify 368928 real contacts. Out of 1471784 million tested contacts with the construction john.smith@gmail.com we find 319814 real contacts. We also tested another constructions with the set of 119020 contacts. We verify 17627 real emails from set of emails with j.smith@gmail.com construction and 7333 real accounts with smith.john@gmail.com construction. John Smith is a name, that we use to demonstrate the email constructions. The information found about users make it possible to verify the name, gain access to the user's photo, and also check the online status. Out of 688742 real users with two most popular email address constructions, 907 users had a profile image. In some cases, it is possible to find related profiles and build a broader picture of a particular user. All our experiments are carried out with careful attention to the personal data of users, and also without the goal of harming the operation of the service.

The results of our experiments confirm that Google Contacts service, as well as the Email Discovery process, are vulnerable to crawling attacks. An ordinary Mid-End PC is used to carry out such an attack, and special in-depth knowledge about the operation and structure of the service is not required, which makes such an attack accessible to many attackers.

We also discuss possible measures that can secure the Email Discovery process in the future. Starting from the state-of-the-art PSI protocols and the concept of Mutual Contact Discovery, which protect this process from attacks on the server, as well as advanced limits that allow one to prevent or reduce the damage from crawling attacks. The built-in CAPTCHA as well as limiting the availability of personal information about the user are important methods of preventive protection.

As a conclusion, we can say that the Email Discovery process has privacy vulnerabilities. Crawling attacks are an effective and low-cost type of attack. Services using Email Discovery must implement preventive measures to prevent or mitigate the damage from such attacks.

List of Figures

List of Tables

Bibliography

[1] S. R. Department, "Ergebnisse einer umfrage zwischen den deutschen benutzer." https://de.statista.com/statistik/daten/studie/162374/umfrage/durchschnittliche-anzahl-von-apps-auf-dem-handy-in-deutschland/, 2015. Accessed: November 2022.

[2] "Whatsapp legal info," 2019. https://www.whatsapp.com/legal.

[3] P. Aftab, "Findings under the personal information protection and electronic documents act (pipeda)." https://parryaftab.blogspot.com/2014/03/what-does-whatsapp-collect-that.html, 2014. Accessed: November 2022.

[4] S. Gupta, P. Gupta, M. Ahamad, and P. Kumaraguru, "Exploiting phone numbers and cross-application features in targeted mobile attacks," in *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 73–82, 2016. https://dl.acm.org/doi/10.1145/2994459.2994471.

[5] "Whatsapp official website." https://www.whatsapp.com/.

[6] "Facebook official website." https://www.facebook.com/.

[7] K. Hill, "Facebook recommended that this psychiatrist's patients friend each other." https://splinternews.com/facebook-recommended-that-this-psychiatrists-patients-f-1793861472, 2016. Accessed: November 2022.

[8] B. für Sicherheit in der Informationstechnik, "Erhebung über den e-mail-verkehr mit der bundesverwaltung." https://www.bsi.bund.de/, 2021. Accessed: September 2022.

[9] D. S. magazin, "Nach massiven protesten: Iran schränkt zugang zu instagram und whatsapp ein," 2022. https://www.derstandard.de/story/2000139303914/nach-massiven-protesten-iran-schraenkt-zugang-zu-instagram-und-whatsapp.

[10] M. Serif, "Whatsapp: Ein staat könnte messenger verbieten – und auf heimische alternative setzen," 2022. https://www.futurezone.de/netzpolitik/article233137377/whatsapp-droht-neuer-aerger-ein-staat-koennte-messenger-bald-verbieten.html.

[11] J. Schectman, "Messaging app telegram moves to protect identity of hong kong protesters," 2019. https://www.reuters.com/article/us-hongkong-telegram-exclusive-idUSKCN1VK2NI.

[12] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1447–1464, 2019. https://eprint.iacr.org/2019/517.pdf.

[13] Á. Kiss, J. Liu, T. Schneider, N. Asokan, and B. Pinkas, "Private set intersection for unequal set sizes with mobile applications," *Cryptology ePrint Archive*, 2017. https://eprint.iacr.org/2017/670.pdf.

[14] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," 2017. https://eprint.iacr.org/2018/787.pdf.

[15] C. Hagen, C. Weinert, C. Sendner, A. Dmitrienko, and T. Schneider, "Contact discovery in mobile messengers: Low-cost attacks, quantitative analyses, and efficient mitigations," *ACM Trans. Priv. Secur.*, vol. 26, nov 2022. https://doi.org/10.1145/3546191.

[16] C. Hagen, C. Weinert, C. Sendner, A. Dmitrienko, and T. Schneider, "All the numbers are us: Large-scale abuse of contact discovery in mobile messengers." Cryptology ePrint Archive, Paper 2020/1119, 2020. https://eprint.iacr.org/2020/1119.

[17] C. Hagen, C. Weinert, C. Sendner, A. Dmitrienko, and T. Schneider, "All the numbers are us: Large-scale abuse of contact discovery in mobile messengers," *Network and Distributed System Security Symposium (NDSS)*, 2021. https://www.ndss-symposium.org/ndss-paper/all-the-numbers-are-us-large-scale-abuse-of-contact-discovery-in-mobile-messengers/.

[18] C. Hagen, C. Weinert, C. Sendner, A. Dmitrienko, and T. Schneider, "Contact discovery in mobile messengers: Low-cost attacks, quantitative analyses, and efficient mitigations." Cryptology ePrint Archive, Paper 2022/875, 2022. https://eprint.iacr.org/2022/875.

[19] "Telegram official website." https://www.telegram.org/.

[20] "Signal official website." https://www.signal.org/de/signal/.

[21] "Xing official website." https://www.xing.com/.

[22] "Linkedin official website." https://linkedin.com.

[23] "Steam official website." https://store.steampowered.com/?l=german.

[24] "Playstation network official website." https://www.playstation.com/de-de/playstation-network/.

[25] "Google contacts official website." https://contacts.google.com/.

[26] J. Klensin, "Rfc 5321: Simple mail transfer protocol (smtp)," 2008. https://www.rfc-editor.org/rfc/pdfrfc/rfc5321.txt.pdf.

[27] P. Resnick, "Rfc 5322: Internet message format," 2008. https://www.rfc-editor.org/rfc/pdfrfc/rfc5322.txt.pdf.

[28] M. Marx, E. Zimmer, T. Mueller, M. Blochberger, and H. Federrath, "Hashing of personally identifiable information is not sufficient," in *SICHERHEIT 2018* (H. Langweg, M. Meier, B. C. Witt, and D. Reinhardt, eds.), (Bonn), pp. 55–68, Gesellschaft für Informatik e.V., 2018. https://dl.gi.de/server/api/core/bitstreams/51457f32-af81-4211-9c89-ea045f4cbed5/content.

[29] A. Priebe, "Status quo: Die 10 beliebtesten e-mail-provider in deutschland," 2016. https://onlinemarketing.de/email-marketing/mobilisierung-mails-top-10-e-mail-provider-deutschland.

[30] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy.," 2021. https://doi.org/10.3389/fcomp.2021.563060.

[31] A. Buchenscheit, B. Könings, A. Neubert, F. Schaub, M. Schneider, and F. Kargl, "Privacy implications of presence sharing in mobile messaging applications," in *Proceedings of the 13th international conference on mobile and ubiquitous multimedia*, pp. 20–29, 2014. https://dl.acm.org/doi/10.1145/2677972.2677980.

[32] Redis, "Redis website," 2022. https://redis.io/.

[33] C. Sendner, "Evaluating the privacy of contact discovery: Hash reversal and tees," 2020. https://www.bibsonomy.org/bibtex/2c8eceb93ef865a7af7cba8ee6d8613af/csendner.

[34] I. T. Union, "The international public telecommunication numbering plan," 2010. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.164-201011-I!!PDF-E&type=items.

[35] R. Project, "Rainbowcrack website," 2022. http://project-rainbowcrack.com/.

[36] K. Stemmler, "Hash tables. what, why how to use them," 2022. https://khalilstemmler.com/blogs/data-structures-algorithms/hash-tables/.

[37] T. Parliament, "General data protection regulation," 2016. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[38] G. C. A. Center, "Rate-limiting strategies and techniques," 2023. https://cloud.google.com/architecture/rate-limiting-strategies-techniques.

[39] S. Kemp, "Digital 2021: Global overview report," 2021. https://datareportal.com/reports/digital-2021-global-overview-report.

[40] Meta, "Facebook messenger privacy website," 2021. https://www.messenger.com/privacy.

[41] Meta, "Information for people who don't use meta products," 2021. https://www.facebook.com/help/637205020878504?locale=en_US&cms_id=637205020878504.

[42] L. Rabe, "Statista: Anzahl der mitglieder der xing-plattform in der dach-region vom 1. quartal 2013 bis zum 3. quartal 2022." https://de.statista.com/statistik/daten/studie/481399/umfrage/anzahl-der-xing-nutzer-in-der-dach-region, 2022.

[43] Xing, "Xing privacy policy website," 2022. https://privacy.xing.com/en/privacy-policy.

[44] S. Sester, "Xing schließt api-schnittstelle für kontakt-export," 2021. https://centralstationcrm.de/blog/xing-schliesst-api-schnittstelle-fuer-kontakt-export.

[45] M. Iqbal, "Linkedin usage and revenue statistics," 2023. https://www.businessofapps.com/data/linkedin-statistics/.

[46] LinkedIn, "Linkedin privacy website," 2023. https://www.linkedin.com/legal/privacy-policy#data.

[47] LinkedIn, "Importing and syncing your mobile address book," 2022. https://www.linkedin.com/help/linkedin/answer/a570243?lang=en.

[48] S. Taylor, "New linkedin data leak leaves 700 million users exposed," 2021. https://restoreprivacy.com/linkedin-data-leak-700-million-users/.

[49] Microsoft, "Microsoft buys linkedin," 2016. https://news.microsoft.com/announcement/microsoft-buys-linkedin/.

[50] Google, "What can you do with google contacts?," 2023. https://support.google.com/a/users/answer/9310345?hl=en.

[51] M. Petrova and J. Elias, "Google's rocky path to email domination. cnbs," 2019. https://www.cnbc.com/2019/10/26/gmail-dominates-consumer-email-with-1point5-billion-users.html.

[52] Google, "Back up sync device contacts," 2023. https://support.google.com/contacts/answer/9423168?hl=en&ref_topic=9160153.

[53] Google, "Add, move, or import contacts," 2023. https://support.google.com/contacts/answer/1069522?hl=en.

[54] Google, "Control what others see about you across google services," 2023. https://support.google.com/accounts/answer/6304920?hl=en.

[55] A. Siddiqui, "Nearly 80 percent of social media users have adjusted their privacy settings in the last year!," 2019. https://www.digitalinformationworld.com/2019/10/research-shows-internet-users-taking-action-on-privacy.html.

[56] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption." Cryptology ePrint Archive, Paper 2017/299, 2017. https://eprint.iacr.org/2017/299.

[57] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing." Cryptology ePrint Archive, Paper 2015/634, 2015. https://eprint.iacr.org/2015/634.

[58] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on ot extension." Cryptology ePrint Archive, Paper 2014/447, 2014. https://eprint.iacr.org/2014/447.

[59] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based psi via cuckoo hashing," in *Advances in Cryptology – EUROCRYPT 2018*, pp. 125–157, Springer International Publishing, 2018. https://eprint.iacr.org/2018/120.pdf.

[60] C. Weinert, "Practical private set intersection protocols for privacy-preserving applications," 2021. https://tuprints.ulb.tu-darmstadt.de/19295/1/Dissertation_ChristianWeinert_010921.pdf.

[61] J. Hoepman, "Mutual contact discovery," 2022. https://arxiv.org/abs/2209.12003.

[62] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*, pp. 551–560, 2009. https://dl.acm.org/doi/10.1145/1526709.1526784.

[63] S. Gupta, "Emerging threats abusing phone numbers exploiting cross-platform features," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1339–1341, 2016. https://dl.acm.org/doi/abs/10.5555/3192424.3192668.

[64] K. Eunhyun, P. Kyungwon, K. Hyoungshick, and S. Jaeseung, "I've got your number," in *Information Security Applications*, (Cham), pp. 55–67, Springer International Publishing, 2015.

[65] H. Hu, P. Peng, and G. Wang, "Characterizing pixel tracking through the lens of disposable email services," 2019. https://ieeexplore.ieee.org/document/8835374.

[66] M. DeVault, "5 professional email address ideas examples," 2023. https://fitsmallbusiness.com/professional-email-address-ideas/.

[67] P. Deorukhkar, "Here are some nifty ideas and examples for a professional email address," 2020. https://sparkmailapp.com/blog/create-professional-email-address-format-examples.

[68] Datenbörse, "Die 200 häufigsten nachnamen in deutschland excel-liste," 2020. https://www.xn--datenbrse-57a.net/item/Die_200_haeufigsten_Nachnamen_in_Deutschland_Excel-Liste.

[69] S. Köln, 2022. https://offenedaten-koeln.de/dataset/vornamen.

[70] LinkedIn, "Profile api linkedin website," 2023. https://learn.microsoft.com/en-us/linkedin/shared/integrations/people/profile-api.

[71] Microsoft, "Profile api," 2023. https://learn.microsoft.com/en-us/linkedin/shared/integrations/people/profile-api.

[72] Xing, "Xing rest-api website," 2023. https://developers.xing-events.com/index.php/REST_API.

[73] OpenJDK, "Openjdk website," 2023. https://openjdk.org/.

[74] Selenium, "Selenium website," 2023. https://www.selenium.dev/.

[75] J. Brains, "Pycharm website," 2023. https://www.jetbrains.com/pycharm/.

[76] Google, "Google chrome website," 2023. https://www.google.com/chrome/.

[77] Google, "Google recaptcha website," 2023. https://www.google.com/recaptcha/about/.

[78] Google, "Android studio website," 2023. https://developer.android.com/studio.

[79] Microsoft, "Microsoft people website," 2015. https://apps.microsoft.com/store/detail/microsoft-people/9NBLGGH10PG8?hl=en-us&gl=us.

[80] Proton, "Proton mail website," 2023. https://proton.me/de/mail.

[81] Intel, "Intel sgx documentation," 2023. https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/library.html.

[82] "General data protection regulation," 2018. https://gdpr-info.eu/.

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich keiner anderer als der in den beigefügten Verzeichnissen angegebenen Hilfsmittel bedient habe. Alle Textstellen, die wörtlich oder sinngemäß aus Veröffentlichungen Dritter entnommen wurden, sind als solche kenntlich gemacht. Alle Quellen, die dem World Wide Web entnommen oder in einer digitalen Form verwendet wurden, sind der Arbeit beigefügt.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Würzburg, 17. July 2023

(Andrii Pastukh)