MASTER'S THESIS

Do iOS applications respect your privacy?

A case study on popular iPhone apps in Belgium.

De Laender, J

Award date:
2023

Do iOS applications respect your privacy?

A case study on popular iPhone apps in Belgium.

by

Jelle De Laender

Master of Science
in Software Engineering

at the Open University, Faculty of Science
Master Software Engineering

Course code:	IM9906	
Thesis committee:	Dr. Greg Alpár (supervisor, chair),	Open University
	Dr. Fabian van den Broek (supervisor),	Open University
Date:	March 28, 2023	

Contents

1. Summary

Figure 1:
© Apple Inc

Every new version of iOS adds new security and privacy features and improvements. iOS 15 added a privacy report and options to hide email addresses when creating accounts by using anonymised email forwards. Application Tracking Transparency, App Store Privacy Labels and Local network restrictions are examples of new features added in iOS 14.

Apple does review every application version submitted to the App Store and verifies if the app meets the latest guidelines. Does this mean that all applications respect our privacy? Apple added App Transport Security (ATS) in 2016, which enforces the use of encrypted network connections. It is, however, still possible to add exceptions and use unencrypted connections.

Apple Tracking Transparency (ATT) is a privacy feature added by Apple to prevent unwanted tracking via iOS applications. Software vendors can track users by collecting data and metrics such as how the application is used, the user's location, the type of other applications used, the user's interest and more. This information can help to profile users and to increase revenue by showing relevant advertisements and personalised content, especially when data from multiple applications are combined. ATT has been enforced since iOS 14.5 and disallows tracking without the explicit consent of the user, as defined in the Apple Developer Guidelines[1].

The main objective of our research is: Do iOS applications respect your privacy? This is researched via a case study on popular iPhone apps in Belgium.

50 iPhone applications are selected and tested. We used a proxy server to intercept data exchanged by the applications and servers. This data allows us to verify if all communication is encrypted, which trackers are used, which data is exchanged and more.

We found that one-fourth of the applications still use unencrypted connections. Almost 10% of all intercepted requests were unencrypted. All applications use encrypted connections. One application is not performing any validation on the validity of the encrypted connection. Eight apps have additional security measures implemented via TLS certificate pinning.

Looking at which data is sent unencrypted, we find quite some applications linking the developer's website legal documents, like the privacy policy over plain text HTTP. Some applications load static files from CDN servers using the unencrypted HTTP protocol, which is mostly harmless as no personal information is involved in the detected cases. Surprisingly, our research shows that most servers are configured for HTTPS, which would make it easy to adapt the applications for the secure variant.

It was also found that some applications execute JavaScript downloaded via unencrypted connections, which introduces security risks as the JavaScript can be tampered with, e.g. by injecting bad code, such as trackers or keyloggers. One app asks the user to upload selfies and sends those in plain text to the developer's server, which adds a privacy risk.

Apple provides a basic tracker that is implemented deeply in the iOS system. Users are asked to opt-in to share analytics when configuring the iPhone, and can be configured in the system preferences. Our study shows that every application in scope has at least one additional tracker installed. Every application has, on average, five extra trackers active, even with ATT set to "Do Not Allow to Track". It was also detected that applications collect data such as cities and carrier names of the users.

[1] https://developer.apple.com/app-store/user-privacy-and-data-use/

2. Introduction

> "Privacy is a fundamental human right. At Apple, it's also one of our core values. Your devices are important to so many parts of your life. What you share from those experiences, and who you share it with, should be up to you. We design Apple products to protect your privacy and give you control over your information. It's not always easy. But that's the kind of innovation we believe in." (Apple Inc[2])

The quote above is found on the website of Apple. Privacy is an important focus and selling point of Apple. As a result, privacy is a big topic every year at the World Wide Developer Conference (WWDC), where Apple presents the latest privacy guidelines and requirements to the developers. Security and privacy are related. Without security, it is hard to ensure privacy, as data can more easily be intercepted, spoofed, altered and more. At WWDC, Apple presents the latest security frameworks[3] and changes that developers need to use and follow to make the applications more secure and to ensure the privacy of the users.

A recent example is the obliged use of App Transport Security (ATS)[4] which enforces HTTPS, the secure variant of HTTP. Another security example is the App Sandbox[5]. The App Sandbox runs applications in an isolated container and controls at the kernel level which resources and actions can be executed. The App Sandbox can control which folders applications can read and write to, which user data is accessible such as photos and address-book, which parts of the RAM it can access, and which sensors and hardware, such as camera, microphone, and GPS sensor, can be accessed. The access is granted via security profiles or explicit user approval.

On January 28 2021, the Data Privacy Day, Apple announced[6] more privacy controls for iOS applications and the app store. Including new labels in the App Store, also known as privacy nutrition labels, summarising which data is used for tracking, which data is processed that is linked to you, and which data is processed but not linkable to the user. Apple also announced AppTrackingTransparency (ATT), an extension to the sandbox. The ATT framework is used to request consent, as shown in Figure 2, to track and access the user's Identifier for Advertisers (IDFA). Apps are not allowed to track a user or access the IDFA ID without this explicit consent. Apple also states that collecting device and usage data, intending to derive a unique user representation, violates the Apple Developer Program License Agreement without explicit consent[7].

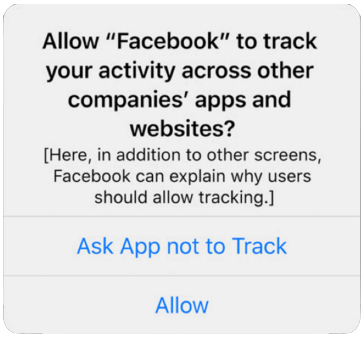

Figure 2: ATT: Example of 'Ask App not to Track' prompt © Apple Inc.

Tracking users in applications is done by collecting data of the users and user behaviours,

[2] https://www.apple.com/privacy/
[3] https://developer.apple.com/security/
[4] https://developer.apple.com/documentation/security/preventing_insecure_network_connections
[5] https://developer.apple.com/app-sandboxing/
[6] https://www.apple.com/newsroom/2021/01/data-privacy-day-at-apple-improving-transparency-and-empowering-users/
[7] https://developer.apple.com/news/?id=ecvrtzt2

e.g. how the application is used, when the application is used, from which locations, type of networks, type of devices, the type of videos that are watched, the search queries entered by the user, which other applications the user uses and more. This information can help software vendors and companies to optimise applications and revenue by showing advertisements adjusted for the current user and personalised content. Especially when data from multiple applications can be linked to one user, a big data set exists with valuable insight. Multiple tools and software providers exist that offer this type of tracking as a service, especially as part of ad service providers.

iOS users can easily install software with only a few clicks via the Apple App Store. Apple works hard on privacy guidelines and rules like the ATS and ATT. All applications submitted to the App Store are reviewed and verified by Apple. However, knowing how much an application respects the user's privacy is difficult. For example, are all network connections encrypted, as the ATS allows exclusions? Which level of encryption is used? Is certificate pinning used to reduce risks of data interceptions? Another privacy issue is that consent to data, such as the user's location via GPS, does not limit what is happening with this data. The location could be processed at the device and sent to the developer's server or shared with other companies. The App Store does not reveal this kind of information. This research aims to get an insight into this and the current state of the popular applications regarding respecting our privacy.

2.1. Privacy

While Apple claims to focus on privacy and security, users of iOS still need to trust the good intention of the developer of applications and third-party frameworks. While access to the camera, GPS, the microphone, e.g., is handled at the operating system level, the user has no insight into what data is captured, collected and sent back to the developer. For example, a user could give a VOIP application access to the address book to call friends easily. However, it is not always clear if the full address book content is uploaded and sent to the developer server to make this possible. Some applications do upload the full address book, or hashes of phone numbers, to determine connections between people, which is not always ideal and can be a privacy concern. Papadopoulos et al. [2018] wrote an interesting article about this issue and proposed a possible solution in this example regarding how users can be identified on one platform without the need to upload all personally identifiable information (PII). This is done by decentralising the information and storing data in DNS records to keep track of relationships and metadata. This setup prevents that one vendor knows everything and limits exposure.

Another case of unwanted processing and exposure of data is given by Mysk Inc.[8], a software company. They reported on February 19 2023, on Twitter[9] that a 2FA application was sending the content of the QR code when setting up 2FA to Google Analytics. This is a security concern as this data contains the secret that can be used to generate 2FA time-based codes.

Developers can also track the user's behaviour and track all actions. A patent of Momin and Moledina [2016] shows how useful tracking can be in the case of tracking fuel levels and (predicted) routes of cars. For application developers, tracking users can give a lot of insight to help improve applications' usability, efficiency, popularity and profit. Especially

[8] https://mysk.co
[9] https://twitter.com/mysk_co/status/1627097291063435264

when developers are tracking data of multiple applications that can be linked to specific users could result in a big and valuable dataset. While Apple no longer gives access to a persistent device ID, developers can identify users using fingerprinting.

When a fingerprinting technique is used, a user will be linked to a 'combination of device properties' such as the device name, screen resolution, timezone, installed fonts, browser agent, software version, and hardware model, eg..., which gives a high chance of uniquely identifying a user from other users in a dataset. This will allow the developer to identify one user in multiple applications and combine the users' data profiles across applications, resulting in wide tracking and analysis. An ad publisher could gain insight into which apps a user likes and uses and personalise the ads for the user. While fingerprints can be used to track people, they can also be used to improve security. For example, fingerprinting can increase the level of authentication when you try to log in on a new device. Websites and applications could require 2FA or send alerts when you try to log in on a new device or browser that is not used before with this account.

Apple does not require developers to inform users which tracking and crash-analytic frameworks and tools are used. However, the European GDPR law[10], art 30 "Records of processing activities" requires companies to keep a public list of all providers and tools that are used to process personal user data. Apple does not require developers to inform users directly which user data will be collected and sent to the developer's servers. This makes it impossible for users to know which data is collected, processed and sent.

While Apple added the privacy nutrition labels, which give a high-level insight into which data types are collected and processed, it is unclear which providers and tools are used. Those labels are only visible on the app store product page, and users are not notified when other data is used to track or processed when an application is updated.

2.2. Paying with privacy

Related to privacy, it is interesting to investigate if there is a difference in collecting user data and tracking users between free and paid software. Is free software possible because 'we pay by giving our data and identity', and is this less the case for paid software? Are all applications collecting as much data possible from users unrelated to the commercial model? To get insight into this, the applications in scope for this research will consist of a distributed set of free and paid applications.

Han et al. [2020] performed research about the differences in privacy in paid and free applications. This was done via an online survey to get an insight into the user expectation and by analysing existing applications to investigate to which extent privacy is respected. During the research, Han et al. confirmed that users expect privacy to be more respected in paid apps than in free applications. Han et al. also examines 5877 applications with a free and paid version to check for differences between both versions. They conclude that paid applications are not guaranteeing better privacy for the users. In Chapter 4 the study of Han et al. is discussed in more detail.

[10] https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

2.3. Objective

This research aims to research to what extent application developers respect the users' privacy. This will be done by researching iPhone applications that are popular in Belgium, are using secured and encrypted connections, if and which trackers are used while the ATT is set to Do Not Track, and by investigating which data is exchanged.

Apple is well known for its attention and care for users' privacy. For example, the strict verification and tests before the software is allowed in the App Store, the required sandbox system, ATS, ATT and more. In my experience, people trust Apple and are installing new software without big privacy concerns. This research aims to verify whether this level of trust is justified by analysing the application implementations and data exchanges.

There are multiple studies available regarding Android and the usage of trackers, TLS encryptions and more. However, for iOS, this is less the case, which was my main motivation for this research, especially because of Apple's statement about user privacy. As an iPhone user, I wanted to verify if all new privacy features of Apple are actually implemented and used like the ATS and ATT. For this reason, the main focus is limited to iPhone and iOS.

3. Technical Background

This section introduces technical aspects such as TLS certificates, validations of certificates like OCSP and CRL lists, and proxies with TLS resigning and interception options. Other parts of this study will continue on this baseline.

3.1. Encrypted connections

With the evolution of the internet, the internet of things, and cloud computing, more and more applications require access to the internet and send user data to multiple servers and third-party services. For example syncing data between different devices of a user, between different people and team members, sharing and viewing content on social media, collecting analytics of application usage and more.

This data is typically sent using the HTTP[11] protocol, defining how a client requests and sends data to a web server. HTTPS[12] is an extension of HTTP by using an encryption layer SSL/TLS[13]. TLS secures communication by using symmetric encryption with a session key. This session key is generated during the TLS handshake, which uses a public key infrastructure. Data encrypted with the public key can be decrypted via the private key, and data encrypted with the private key can be decrypted with the public key. As the name reveals, the private key is only known by the server's owner, while the public key is publicly known. The public key is distributed via TLS certificates, also known as public-key, digital, or identity certificates.

Authenticity and integrity

One of the advantages of the public/private keyset is the ability to verify data integrity and authenticity, as the other key is required to tamper with or change information. To ensure the public key is the actual public key of the other party and not a key of, for example, a hacker, validations are required. This is one of the main advantages of TLS certificates, as it allows for checking the trust level of a certificate and the embedded public key.

[11] https://tools.ietf.org/html/rfc2616
[12] https://tools.ietf.org/html/rfc2818
[13] https://tools.ietf.org/html/rfc5246 & https://tools.ietf.org/html/rfc8446

TLS validation

Multiple attributes should be verified to know if a TLS certificate can be trusted. For example, by checking the date range that the certificate is valid for, defined by the issued date and expiration date. By checking the common name, which limits for which domains the TLS certificate can be used. To verify if the attributes of a certificate are not changed, a certificate is signed by a different certificate, called the issuer, and contains a signature that can be verified. The issuer certificate used to sign the certificate can also be signed via another certificate to ensure no data is tampered with and forms the TLS certificate chain. The chain can be multiple levels deep.

The top issuer is self-signed, so not signed by a different certificate and is called the root certificate or CA certificate. Every operating system has a default set[14] root certificates that are trusted by default by the OS.

A certificate is valid if the certificate succeeds all validations, including each certificate in the chain, and the root certificate needs to be trusted on the device. It is important to state that users can bypass this verification by marking a certificate as trusted on the device. Once one validation fails in the certificate or chain, the certificate will be marked as untrusted and insecure.

Revocations of certificates

Certificates can also be revoked at any time, for example, when abuse is detected. Every certificate in the chain is linked to an online list called the 'certificate revocation list'[15], also known as the CRL list. This list is published by the certificate owner of the root certificate and contains the revoked certificates signed by this certificate. This will make all certificates that are signed by this revoked certificate invalid. Part of the validation is a check on the CRL list, starting at the root certificate to verify that none of the certificates in the chains is revoked.

OCSP

Besides the CRL list, the 'Online Certificate Status Protocol', in short OCSP, can be used to verify remotely whether a given certificate is revoked. This process requires less memory and is faster as no CRL list needs to be downloaded and processed.

It is important to notice that most OCSP requests are often performed over plain text HTTP as no sensitive data is sent, only a hash of the current certificate, and all data is public. This avoids a recurrent loop where the OCSP request triggers another OCSP lookup request when the request is performed over HTTPS. OCSP requests may be over HTTPS when required for privacy or security reasons, also defined in the RFC as:

> Where privacy is a requirement, OCSP transactions exchanged using HTTP MAY be protected using either TLS/SSL or some other lower layer protocol.
> Myers et al. [1999]

3.2. Capturing data via a proxy

Validating TLS certificates can ensure a user the data is encrypted in transit. However, when a proxy is used on the network, the data could still be intercepted, harming privacy and trust. By using a proxy, all traffic on a network or device can be rerouted via the proxy server,

[14] For example, https://support.apple.com/en-us/HT210770 for the default set on iOS 13
[15] https://tools.ietf.org/html/rfc5280

allowing the proxy to intercept all data. When the data is sent unencrypted, plain text, the proxy has insight into all transmitted data. When the data is encrypted, the proxy server will not be able to get details of the data sent as the data cannot be decrypted in theory as the private key used in the encryption is unknown. Illustrated by Figure 3.

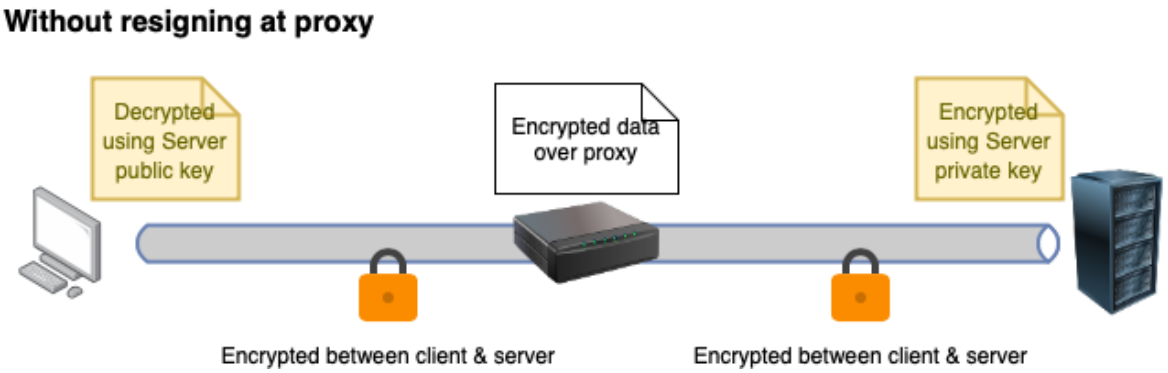

Figure 3: Visualisation of a proxy without TLS resigning. The proxy cannot decipher data sent between the client and server, as the encryption keys are unknown.

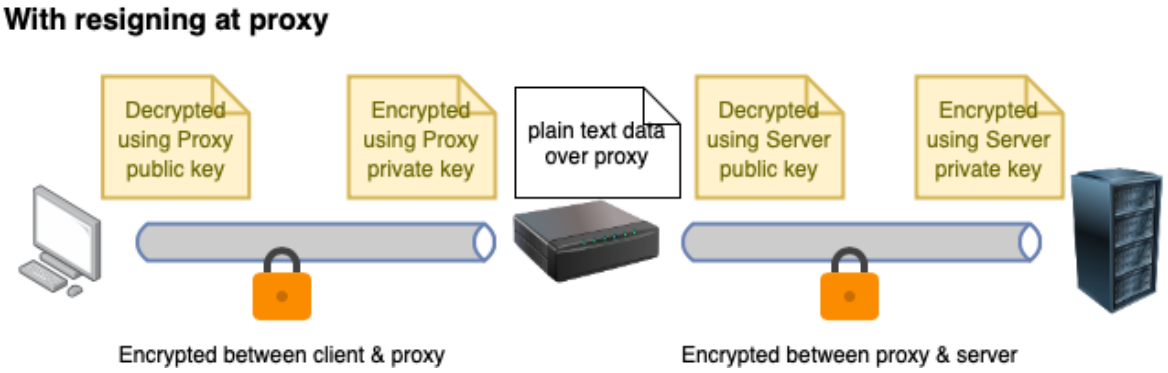

Figure 4: Visualisation of proxy with TLS resigning at the proxy level. The proxy has insight into data going over the proxy server, as it acts as an endpoint and creates two different HTTPS connections, one acting as the server and one acting as the client.

A proxy could get all details when it resigns the connections, known as TLS interceptions. To do this, it will need to spoof the original server. For example, assume an application requests data from a server and a proxy server resigns the connection. The proxy will act as the server of the application and will create a new encrypted connection between the proxy and the application. The proxy server will act as the client to the server and will set up an encrypted connection between the proxy and the server. See Figure 4 for a visualisation.

One of the issues with this setup is the TLS certificate used between the proxy and the client, as the proxy cannot use the original TLS certificate since it does not have the original server's private key. The proxy must make and use a new TLS certificate to encrypt the connection between the proxy and the client. The TLS validation on this certificate will fail as a well-known CA root certificate will not sign this new certificate. The proxy cannot properly prove the ownership of the intercepted domain, which is required to get a trusted root certificate to sign your certificate.

As a result, the proxy will use a new root CA certificate and use this CA certificate to sign new certificates for the requested domain. The client device does not know this new

CA certificate, so this connection will still fail and be marked as untrusted. Luckily, for our research, it is possible to mark (root) certificates as trusted when you have full admin access to the device. Once the CA certificate is marked as trusted, the certificate validation will succeed on any newly created certificate by the proxy.

Researchers and corporate networks often use this proxy setup to analyse the network traffic in deep detail. This is also used to block unwanted content by web content filters on networks.

3.3. Certificate pinning

Certificates can be verified if valid and trusted by checking the attributes and chain as explained in Section 3.1. However, as explained in the previous section, this does not guarantee that the used certificate is the real original certificate. It only tells us the certificate is valid for the current context. For example, a proxy could resign connections without the user's awareness or consent in a corporate network where the company manages the devices, especially when the proxy CA certificate is installed and marked as trusted, e.g. by installing the network configurations requested by the IT department, installing software and more.

The certificate pinning technique can be used to avoid or reduce this risk. Certificate pinning is verifying if the given certificate is the expected certificate. All other certificates will be rejected. This verification can be implemented at different levels and via different methods. This extra check can be implemented in client software and applications.

In most cases, the certificate pinning is performed by checking the certificate attributes. For example, if the used root certificate is the expected root certificate by storing a hash or ID of this certificate in the app. Or by storing the hash of the issuer certificate or 'issued at' or 'expiring date stamps. The expected values are often hard coded in the application and part of the compiled binary. The downside is that the application needs to be updated whenever the server certificate changes, based on how strictly the certificate pinning is implemented.

Pradeep et al. [2022] performed research in 2021, with a collection of 2515 iOS applications and concluded that roughly 11% had a type of certificate pinning enforced.

4. Related work

Research on security and privacy on mobile applications is not a new subject but is evolving heavily. New weaknesses and possible hack and attack methods are frequently found and abused. Best practices for the discovered security problems are frequently evaluated and adjusted. Multiple other types of research are related to our research which are described in this section.

4.1. Tracking methods

Multiple studies about tracking users on iOS and other mobile devices and systems exist. An example is the older study of Egele et al. [2011], which focuses on finding privacy leaks in iOS applications, which is relevant to our research. The research of Egele et al. investigates which threats applications are posing to users by introducing a tool called PiOS (Privacy iOS). PiOS is used to analyse more than 1400 iPhone applications for possible privacy leaks and sensitive information by performing static analysis on the binary of the applications. The research inspects applications and looks for variable names and system calls to sensitive personal data. Egele et al. conclude that most applications respect privacy. However, more

than half of the applications are using and leaking the unique device ID (also known as UDID), which can be used to track users, which can be seen as a breach of the user's privacy.

It is important to state that the research of Egele et al. is dated. For example, Apple is not giving access to the UDID anymore. Since iOS 6, a developer can only use an *'identifierForVendor'* which is different on one device for multiple software vendors but shared between all applications of the same vendor on that device. The ID is also not guaranteed to be saved and reused when a user deletes all vendor applications. This improves privacy for users as it is harder to track users between applications of different vendors, for example, by ads service providers or application analytic services. Another important fact is that in the study of Egele et al. the sandbox use was not enforced. Apple did enforce this on all app submissions in the App Store starting in March 2012. This enforcement resulted in iOS requiring user permission when sensitive data such as photo albums or GPS sensor is requested. Applications could do this secretly before the use of the sandbox was required.

While Apple did increase security and made it harder for developers to use sensitive data of users secretly, the study of Egele et al. is still relevant as they describe an interesting method to analyse applications that can be automated. Egele et al. investigate which advertisements and tracking libraries are used by applications, which is common with our research. In contrast to Egele's research, our study will detect this by intercepting network traffic and reviewing the requested domains versus inspecting binaries for the existence of frameworks. The study of Egele et al. is also limited to free applications and a set of applications available via jailbreak stores (which can more easily bypass any security enforcement). In our research, we will include paid applications to verify if there is a noticeable difference regarding privacy between free and paid software.

Kurtz et al. [2014] performed a related study of Egele et al. in 2014 and introduced a new framework called DiOS, where iOS applications can be analysed based on the called system API functions. While Egele et al. is performing research on a static binary that won't change, the research of Kurtz et al. is considered dynamic. It is dynamic as the applications run, and user interaction is simulated via random actions. System calls and network traffic are then analysed to inspect and search for privacy breaches. The study of Kurtz et al. focuses on free applications. The conclusion is that almost one-fifth of the investigated applications (1136 applications in total) are asking for or using the user his location, one-third of the applications in the social network category want to access the address book of the user, and almost half of all applications are tracking the user app usage and behaviour via tracking services.

The research of Kurtz et al. is similar to this and lists which advertisements and tracking frameworks/services are detected. The biggest difference is the way how the applications are tested. Kurtz et al. test via an automated way, a large scale and a set of free applications. Our research has a smaller set of applications and will include paid and free applications. It will be possible to investigate if any differences between free and paid applications are noticeable. Automated testing tools use an algorithm to detect all possible flows and paths in apps and execute those. The applications in our research will be tested manually as the human heuristics, know-how and logic, which is hard to implement, can help detect sensitive areas and features in applications.

4.2. Intercepting and deciphering TLS network traffic

By using a proxy server, it is possible to get insight into which domains and servers are contacted by an application. In cases when there is no certificate pinning implemented, we can get insight into which data is sent over the network even when encrypted connections are used. Shah [2012] suggests a possible proxy setup that can help to collect the required data.

Pradeep et al. [2022] researched how many applications have certificate pinning implemented on iOS. When certificate pinning is implemented, our proxy setup will not allow resigning connections and get details of the data sent over the network. Luckily for our research and sadly less for the privacy protection level, he found that only about 11% of 2515 iOS applications had certificate pinning enforced in 2021.

D'Orazio and Choo [2017] describes a method to disable TLS validation at the OS level, which could allow us to still use our proxy and get more insight into the data sent by applications with TLS certificate pinning enforced. This is done by using the *SSL Kill Switch project*[16], which is only possible on a jailbroken device which is not the case for our research and does not work for every application, depending on how the TLS certificate pinning is implemented.

4.3. Use of encrypted connections

One part of the research verifies if all connections are encrypted. Razaghpanah et al. [2017] recently performed a study about the adaption rate and usage of encrypted connections on Android. The study of Razaghpanah et al. is based on data collected by Lumen.

Lumen[17] is an academic initiative that provides an Android app that people can install. The app will then run a local proxy on the Android device, intercept all data, and analyse this data. It will then send reports to the database of Lumen with anonymised statistics. No personal user data is collected. Lumen collects statistics about which data types are sent by applications, if encrypted connections are used, which encryption techniques/algorithms are used, and more. The Lumen database is limited to Android data. I couldn't find a similar project with iOS data yet, possibly because the guidelines and strict rules on iOS make this not possible or complex to implement.

Researchers can access the Lumen database to run queries and draw conclusions on the datasets. While Android users generate the Lumen data, the dataset can be interesting and relevant for our research to compare findings. However, as Lumen is not well known and is mostly used by people that want to get insight into their privacy, it is not clear if the Lumen dataset is representative of all users and popular applications like in our research.

Razaghpanah et al. research focuses on possible vulnerabilities based on which framework is used to validate the TLS connections. 84% of the selected applications using TLS use the default frameworks and configuration provided by the OS to validate TLS connections. The other applications use third-party or custom versions to validate TLS connections. The frameworks provided by the OS offer a suitable validation but can easily be outdated as the OS is updated less frequently than applications. It is also possible that older devices cannot update the OS anymore while the applications can still be updated. Using a third-party or embedded TLS validation framework in the application can help to solve this problem. However, Razaghpanah et al. found that widely-used applications had alarming vulnerabilities

[16] https://github.com/nabla-c0d3/ssl-kill-switch2
[17] https://www.haystack.mobi

and weaknesses in their implementation. They find that almost no applications are implementing certificate pinning. If certificate pinning is implemented, it is mostly done for applications of larger companies or privacy-sensitive applications such as banking software.

A similar study is performed by Chothia et al. [2017] with a focus on 15 applications of UK banks. Chothia et al. investigated if the banking applications had any known TLS flaws. They find that 8 of the 15 applications are using certificate pinning. 5 applications have serious vulnerabilities. 2 applications with certificate pinning implemented failed to validate the certificate's hostname, which poses risks of abuse. This is possible because the certificate pinning makes it more complex to test all validation requirements and can give a false impression of being more secure. Chothia et al. also found that some applications still requested data over plain text, which could endanger security and privacy. The research of Chothia et al. proves that even applications that are assumed to have top-level security can have vulnerabilities and not always protect the users' privacy. The difference between our research and the research of Chothia et al. is that our research focuses on which data is collected, which trackers are used, if data is sent encrypted and if TLS certificate pinning and if TLS validation is used, and not searching for TLS flaws or implementation issues.

Orikogbo et al. [2016] performed a static research on iOS applications. Due to the static testing, only URLs in the app binary were detected, and URLs returned by API calls were not detected, which is the benefit of the dynamic testing. This research was performed in 2016, when Apple did announce the ATS, where HTTPS usage was *enforced*. The finding of Orikogbo et al. is that 9.32% of the network connection endpoints fail to offer HTTPS endpoint correctly. Only 2% of the applications are fully encrypted and free of unencrypted network calls. The results are not completely relevant to our research, as the study was conducted six years ago, and the introduction of ATS, which objective is to have all calls via HTTPS, was only recently announced.

4.4. Privacy

Kollnig et al. [2022a] performed a study on 24.000 Android and iOS applications from 2020, with privacy as focus point. The focus is on detecting third-party tracking, and detecting if unique user identifiers are shared. The study was performed by analysing the binary (static testing), as well dynamically by intercepting traffic. They present a methodology for large-scale testing and automatisation. A tool was developed and made available at platformcontrol.org[18]. Kollnig et all. intercepted network data exchanged between the application and servers by opening the application and leaving it open for 30 seconds. No interactions or actions were performed on the app. Our study has fewer applications in scope, but the applications will be tested in more detail, as the objective is to go over each flow and action in the application. This will reveal any traffic that is performed by using the application.

Binns et al. [2018] performed a similar technique by statically scanning 959,000 Android application binaries. The conclusion of Binns et al. is that the median number of trackers per app is 10. 90.4% of the selected Android apps have at least one tracker embedded, and 17.9% has more than twenty trackers embedded.

Kollnig et al. [2022b] performed a second study, extending the previous research of Kollnig et all. The new focus is to verify if any changes could be detected after the announcement and enforcement of ATT, the App Tracking Transparency. The conclusion is that the top

[18] https://www.platformcontrol.org

tracking services have not changed since the introduction of ATT and are still used. The average number of trackers dropped a little bit from 3.7 to 3.6. The number of applications with at least one tracker did increase from 86.39% to 87.52%. So there were no notable changes since the introduction of ATT.

Privacy and monetisation model

Han et al. [2020] investigates if there is a difference between free and paid applications regarding respecting users' privacy. The study of Han et al. includes an online survey to understand better the expectations regarding privacy on free and paid applications. This survey shows that users expect paid applications to be more privacy-friendly than free ones.

Han et al. also perform a static and dynamic analysis of applications. The applications in scope are all Android applications with free and paid variants. This is important as Han et al. investigate the differences between the free and paid application variant to conclude if the paid application respects privacy more than the free variant. The list is selected based on the top free applications in the Google Play Store, filtered on applications with a paid variant and applications compatible with the test devices and available in the current region. Applications that are too expensive, more than $10, are also discarded. They ended up with 5877 applications in scope.

Those applications were inspected statically by checking which frameworks and libraries are used and which permissions are declared. Han et al. then investigated if free and paid applications used the same frameworks and requested the same permissions. The applications were also inspected dynamically using a proxy setup to check which data was sent by the applications and if any data was marked as sensitive.

Han et al. conclude that 45% of the paid applications have the same third-party libraries as free variants. 74% of the paid versions request the same, considered dangerous, permissions as the free version. The main conclusion is that paid applications do not guarantee better user privacy.

The study of Han et al. is similar to our research. The scope of applications is different as Han et al. focus on Android, while in our research, popular iPhone apps in Belgium are selected. It is also unclear if there is a difference between Android and Apple regarding privacy, as Apple has a stricter review process before applications are accepted in the App Store. The research of Han et al. also filters on applications with a free and paid variant. This can give an incorrect representation, as those applications have a higher chance of sharing code between both variants, including the used frameworks and data being collected by users. Applications that are only available for free or only available paid are ignored in the research. Applications that are more expensive are also excluded, which can have an impact on the findings. If more expensive apps provide a better security and privacy baseline, this will not be detected or impact the study. In our research, the selection is based on popularity as described in Chapter 7 and mixed on the app category and monetisation model.

4.5. Conclusion

While multiple research studies have been performed, our study adds value as it researches the current state of iOS applications on security and privacy. While Apple defines more strict best practices, developers can still use exceptions in some cases. For example, the ATS enforces HTTPS, but still allows unencrypted requests on a predefined set of domains. Our research tries to verify if developers are taking this loophole and taking the easiest way to

meet the new security guidelines. Our study will also check if trackers are loaded and used while ATT is set to Do No Track. The study is also limited to popular iPhone applications in Belgium. This also includes paid applications which are often excluded from other research.

5. Research question

It is possible to download and install applications from unknown developers with a single touch while being connected to potentially untrusted and uncontrolled networks, making privacy more important than before.

iOS users often assume their privacy is ensured and guarded by iOS, Apple and the developers, especially by the marketing and privacy focus of Apple. A user cannot easily know which data is collected and tracked, how this data is tracked, processed and sent to the developer, and for which purpose. Are all movements and clicks of users tracked? Do iOS applications send personal and identifiable data? In this research, the aim is to get an insight into which trackers exist and which trackers are used. As users of those applications, are we aware that we are tracked? Besides being tracked, which data is sent to the developer servers? Is this data sent securely or over plaintext? When data is sent in plain text, this could cause privacy issues when untrusted networks are used as data could be intercepted by apps and network devices.

Research Question: To what extent do iOS applications, that are popular in Belgium, respect the privacy of the users?

To get a detailed answer to the research question, six subquestions are defined:

- RQ1: Which tracking techniques exist to track users on iOS?
- RQ2: Which tracker services exist to track users on iOS?
- RQ3: Which tracking services can we detect in our sample set?
- RQ4: To what extent do iOS applications use secured connections?
- RQ5: Are TLS validation and TLS certificate pinning used?
- RQ6: To what extent are differences observable between the monetisation models regarding user privacy?

Those subquestions are all related to the main question. RQ1-3 are about how users are tracked and which trackers exist. This has a connection with privacy as this will expose the end-user behaviour and possibly exposes personal data and behaviour. RQ4-RQ5 verifies the used connections to research if the data can easily be intercepted on eg. public networks. RQ6 focuses on the monetisation model and its impact on RQ3-5.

In the following section, the research method is explained in more detail. Chapter 7 defines which applications are selected for this research and how this list was created. In Chapter 8, the results and findings are discussed and analysed. The security aspect of the connections (RQ4 & RQ5) are first investigated, followed by the research on tracking techniques, services and detected trackers (RQ1-RQ3). The findings of RQ1-RQ5 are then compared based on the monetisation model for RQ6, and Chapter 9 contains the conclusions.

6. Research method

This research will be done via a quantitative method. A list of 50 popular iPhone apps in the Belgian App Store will be defined and tested. Chapter 7 explains how this selection will be made and which applications are selected.

The selected applications will be tested using dynamic analyses. This means that we will run the applications and analyse the results, whereas static analyses are performed on source code or binaries of not running applications.

The advantage of dynamic testing is that the code is executed. The intercepted traffic will contain all data identically as when users would be using the application. While with static testing, it is harder to determine which attributes and data will be actually sent on the network and which functions are used, especially as we don't have access to the source code but only the compiled binary. It is possible that a binary has references to a tracker, which are never called or used when running the application, for example.

The downside of dynamic testing is that we need to go over all flows and run the applications in detail. To cover this, each application will be tested manually by clicking all buttons and going over all detectable flows and screens, which is time-consuming.

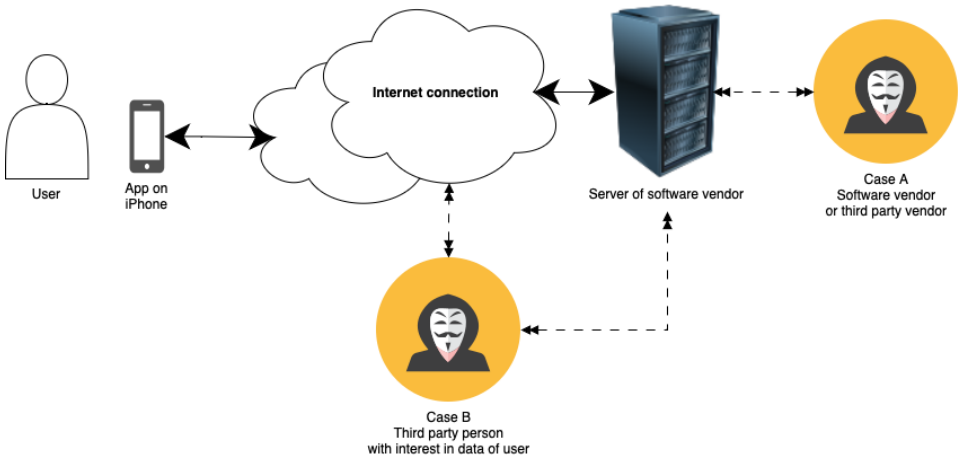

Figure 5: Overview attacker model

To research to which extent the privacy of a user is respected, two different criteria will be researched: Firstly, we will investigate which trackers and tracker software are used in applications, and which data is exchanged. Secondly, we will verify how data, such as the tracking data and other data exchanged to, for example, the software vendor's server, is secured and protected. We will verify if any encryption is used, if the TLS certificate is properly verified and more.

The combination of both criteria will give insight into how much the user's privacy is respected. When the user is tracked in detail, the software vendor, but also the tracking software vendor, will have insight into the usage and data of the user, which may harm the privacy, as shown in Figure 5 case A. When the data is sent insecurely, this exposes a risk that unwanted people can get insight into which data is exchanged, which may contain personal data, as shown in case B. To ensure privacy is respected, no personal data should be collected and exchanged without consent, and security should be adequate.

The applications will be tested manually due to multiple reasons. One reason is that iOS applications cannot easily be tested automatically without access to the source code. The

iOS sandbox prevents applications from accessing and communicating with other applications. The sandbox also prevents applications from controlling the device by simulating user interactions. This could be possible via VoiceOver commands, by jailbreaking the device or by (ab)using the mouse support[19] added by Apple in iPad OS 13.4. However, the current user interface must be interpreted and processed to make automated tests logical and meaningful. This research is considered out of scope and is described in the future work section.

The number of applications in scope is calculated based on the estimated time to test applications and process and analyse the data sets manually.

A proxy server collects and intercepts which data is exchanged, to which domains and via which methods. The following section explains the technical setup that will be used to get the required data in more detail. The data collected via the research will give insight and support answering the research subquestions and questions. This will be discussed further in Section 6.2.

6.1. Technical setup

This research will be performed by using a dedicated test network. This network is managed via a router with only two devices connected: The iPhone used to run the applications in scope and a computer running the proxy server to intercept all requests. The iPhone is set to proxy all traffic via the proxy server. See Figure 6.

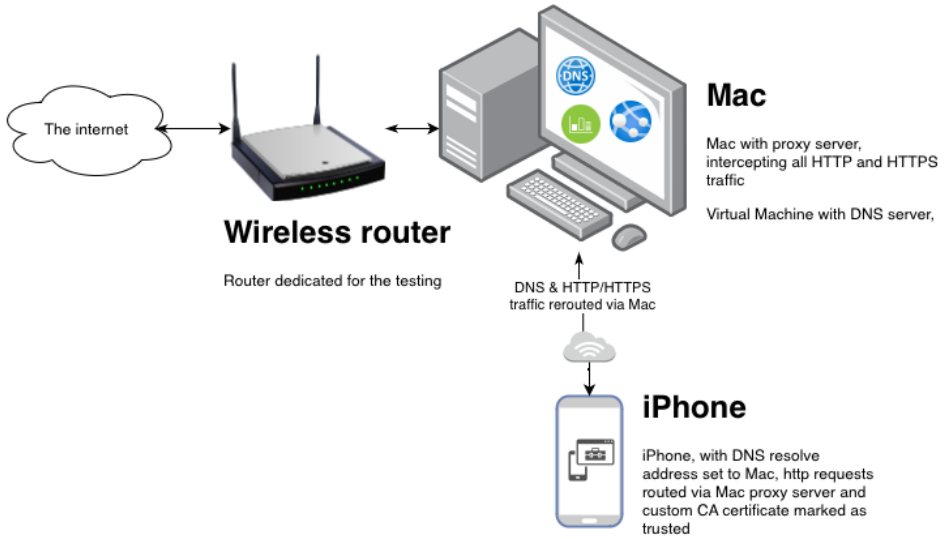

Figure 6: Overview of the test setup

It is important to note that we have full access and control to the test device and the local network. This allows us to intercept network traffic, even when encrypted connections like HTTPS are used, using a proxy that resigns connections. This requires us to install and trust the proxy CA root certificate, which is only possible when you have full admin access to the devices.

Using this setup, we will get an insight into all domains the applications are requesting. We can verify if any encryption is used or if there are plain-text requests. We can get a copy of which data is transmitted and received as long as no multiple layers of encryption

[19] https://developer.apple.com/documentation/ios_ipados_release_notes/ios_ipados_13_4_release_notes?language=objc

are used. Based on the domains, we can check if any trackers are used and which ones. We can verify which data is sent back to the application's server based on the data. This allows us to verify if any data is sent that could be used to track users, such as device IDs or data that can be used for fingerprinting. Note that a proxy server will only work for applications without enforced certificate pinning. The iOS application will reject our new certificate and connections when certificate pinning is implemented.

This setup will also allow us to check which level of HTTPS validation is used in the application. When certificate pinning is implemented, the application will reject any certificate with a different fingerprinting or hash than a predefined set, depending on how the certificate pinning is implemented. This is expected for privacy and security-critical applications like bank software and social networks. In most cases, a normal HTTPS chain verification is expected. This can be verified by checking if the application only works when the new root certificate is trusted. If the application processes any data from the server while the root certificate is untrusted on the device, the application should reject the connections. The application is vulnerable to man-in-the-middle (MITM) attacks if it still accepts the connection with an untrusted root certificate.

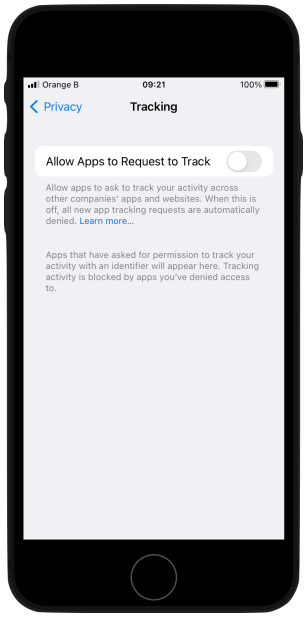

Figure 7: Overview of ATT settings on the test device.

It is the plan to test the current state regarding privacy. To achieve this, we want to run the latest version of the applications in scope. To make this possible, we use an iPhone still supported by Apple. This will be an iPhone Xs with iOS 15. As the iPhone Xs is a recent model and can run the latest available iOS version, it should be able to run the latest version of all App Store applications. If there is an update of an application available during the test, the test for that application will be redone after doing a clean install of the application to ensure we have 'first use' behaviours included in the testing.

As we want to check which applications use trackers without consent, the App Tracking Transparency (ATT) will be disabled system-wide. This setting can be found at Settings ⇒ Privacy & Security ⇒ Tracking, as shown by Figure 7.

Part of the research also investigates if the testing can be more efficient by automating the flow. Findings are discussed in the future work section.

On top of the proxy to intercept all requests, the computer will also run a DNS server and the iPhone default DNS resolver will be set to the computer. This will allow us to make a copy of all DNS requests. The list of DNS requests will reveal which domains are connected to by the iOS application. Note that most OS systems have a cache of the last DNS lookups. There is a risk that not all requests will be sent to our custom DNS server, if there is a recent request for that domain. This is not a problem for this research as we don't need to know exactly how many times a domain was called, but we need to know which domains are requested. The test iPhone will also be put on airplane mode between each test to ensure the local DNS cache is flushed. The DNS logs are an additional data source to verify our findings and requested domains detected via the proxy and to check if any domains were missed.

Used hardware and software

Router: An Asus RT-AC68U. Dedicated to this test. No other computers or devices were connected to this router to reduce noise and other traffic. This router also offers some insight into the network traffic but was too basic for this test.

Computer: A MacBook Pro, running macOS Monterey 12.5, with a licensed Charles Proxy[20] for the proxy server. We did also run PiHole[21] in a virtual machine (VM). This VM was running via VirtualBox[22]. At the end of each test run, a copy of the proxy record session, and PiHole logs were saved to be analysed.

iPhone: An iPhone Xs, with iOS 15.7.2 configured to use the PiHole as the default DNS server, proxying all traffic to the MacBook Pro, and Charles Proxy CA Root certificate marked as trusted.

6.2. Data collection

We will collect and make an overview of all requests captured by the proxy for every application in scope. We check whether the connection was encrypted or sent as plain text for every request. We will list the domain and which data were sent and received. See Table 1 for an example of sample data. Based on the domains, it is in most cases possible to detect if trackers are used. The data parameters (eg parameters in the URL, HTTP Body request for POST calls) can help to detect if any data can be used to identify a user. We can also verify which data is sent to identify the user, such as device properties, that can be used for fingerprinting.

App	Domain	Protocol	Parameters	Data received & ...
9292	api.9292.nl	HTTPS	byTram:true& ...	...
9292	www.9292.nl	HTTPS		...
LinkedIn	linkedin.com/mob/sso/you	HTTPS	x-li-track:...	...
AppFigures	api.appfigures.com/v2/reports	HTTPS	...	...

Table 1: Example sample of collected data

By setting the new root certificate as untrusted on the test device and retesting the applications, we can detect which applications are not enforcing any HTTPS validation and accepts any connections (and are vulnerable to MITM or other attacks as they will keep working with invalid certificates), which indicate the security level of the application.

We will perform a clean install of every application in scope, ensure only one application is installed each time, and start monitoring the traffic before the application is started for the first time. It is useful to get a copy of all the data the application sends on the first launch. The idea is then to go through multiple flows and screens of the application. For example, by filling in the contact form, registering an account, logging in, and using the password forget flow,... The application is also left idle in the background, reactivated and forced quit and restarted to check if any other data is being sent.

As shown by Figure 7, the "Allow Apps to Request to Track" will be disabled at the iOS system level.

[20] https://www.charlesproxy.com
[21] https://pi-hole.net
[22] https://www.virtualbox.org

6.3. Analysing data

The data collected needs to be processed before we can make conclusions. For every application in scope, we need to review the collected data and check for the following items:

Encryption: Are not encrypted network calls performed? How many calls were sent unencrypted? To which domains are insecure data sent? Which data is sent over those plain-text connections? What is the risk for the user regarding privacy?

TLS Validation: Are all HTTPS connections rejected if the proxy CA root certificate is not marked as trusted? Are any requests failing, even when the CA root certificate is marked as trusted? Is TLS certificate pinning detected? Which TLS version is used?

Domains: Which domains are requested? Can we link those domains to services like trackers? Are there domains owned by the developer? Are any unexpected domains being contacted?

Data: Which data is sent about the user? Is this data required for the application? Is any data sent that is expected to be processed locally? Any unexpected data such as names, cities, tracking data,... Can we detect fingerprinting?

To automate and improve the processing of data, custom scripts are developed. Those are available at https://github.com/jelledelaender/OUProxyAnalyzer and are part of this research. The scripts will be expecting output files of Charles Proxy and PiHole as input. See the README[23] files in the repo for more information on the use of the scripts.

It is important to note that detecting which data is being sent is not always easy or possible. Data could be encrypted (on top of the HTTPS encryption), hashed, represented in a custom binary format, or transformed (e.g. split into multiple parts). Data could be sneaked into unrelated requests or hidden in other data objects. For example, a user's location could be hidden inside an uploaded image metadata file or inside a large JSON object of a different request. This research will focus on data that is retrievable.

6.4. Validation

The validation of the findings and collected data is taken into account at multiple levels. For example, in the technical setup, as the procedure to collect data, and by searching similar research to compare findings.

Data integrity: Data is collected by running applications and intercepting data via a proxy server. The device is fully wiped to avoid noise; only the single application in scope is installed. Before testing another application, the previous application is deleted to ensure no background tasks are still running. When unexpected requests are detected, e.g. by the system, the test is repeated to verify the data.

Data completeness: To ensure all data is collected, multiple aspects were taken into account:

- It is possible to intercept data and requests on multiple methods. An HTTP proxy is one method. Data can also be rerouted by using a VPN setup, or data

[23] https://github.com/jelledelaender/OUProxyAnalyzer/blob/main/README.md

could be intercepted at the router or network gateway device. For this research, and after a proof of concept, it was decided to use the proxy setup as the proxy showed all requests of the test cases and insight into the data with resigning option.

- It was also decided to use a custom DNS server to keep track of all requested domains via a second, independent system. The DNS server has a log of all requested domains that need to be resolved. This list can be compared with the list of detected domains in the proxy data to verify if any important domain was not detected in the proxy setup.
- Part of the testing flow was also to force-quit an application and start it again. This was done to ensure we had multiple starts recorded. Most applications send data when they launch, like to fetch resources, check for updates, and simply report the user's session, which device and so on. Some applications have a different launch for the initial launch, e.g. optional permission requests are often only asked at the later start of the application to reduce the clutter and improve the user experience on the app's first launch.

Other research: Part of the related work, Chapter 4, is about other research focussing on the part of this study, but via a different research method. The results can still be interesting to compare and validate our findings. For example Pradeep et al. [2022] and Chothia et al. [2017] about TLS certificate pinning and use of encrypted connections, Han et al. [2020] about tracking and monetisation models. Also, the setup of using a proxy to collect such data was confirmed by Shah [2012].

Other methods to validate the findings are discussed in Chapter 10: Future Work.

7. Applications in scope

Our study researches if applications respect the users' privacy. This is done by testing applications that are popular in Belgium. This section explains how the set of applications that are tested was defined.

When Apple announced the first iPhone in 2007[24], users could not install any software. All applications on the iPhone were pre-installed by Apple. In 2008[25], Apple announced iPhone 2.0 software (now simply known as iOS) which added the App Store. The App Store is the most user-friendly and common way to get third-party software on an iOS device. There are alternative ways to install custom software on iOS, for example, by jailbreaking the device or using in-house App Store distribution profiles, which are meant for companies to distribute software internally, like internal tools. Apple controls iOS and the App Store strictly by deciding which software can be added and distributed via the App Store. All applications must comply with strict guidelines and rules, which Apple checks every time a developer submits an app version to the App Store.

As almost all downloads and installs of software on iOS devices happen via the App Store, Apple has insight into applications' download numbers and popularity. Apple shares the exact download numbers with the developer but not publicly. Luckily, Apple maintains and shows top lists in the App Store per country and provides three types of top lists:

[24] https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/
[25] https://www.apple.com/newsroom/2008/03/06Apple-Announces-iPhone-2-0-Software-Beta/

- Free apps: Top list of all free applications downloaded and used.
- Paid apps: Top list of all paid applications downloaded and used.
- Grossing apps: Top list based on the revenue of applications. Including in-app payments and subscriptions. This list contains free apps with in-app payments, as paid applications.

Lim and Bentley [2013] performed research on the used algorithms in mobile ecosystems and concluded the importance of the algorithm to define the top rankings to make the app store successful. Apple does not reveal its algorithm for defining the top ranking. Software developers could abuse it more easily to force their applications into the top rankings and get extra visibility and exposure. For example, if the ratings on the first day of a new application are an essential factor for the ranking, developers could heavily buy fake ratings to manipulate the rankings. Alex Walz wrote a blog article[26] about App Store Optimization (ASO) and performed research into which elements can have a factor in defining the ranking, with the conclusion that the top-ranking lists evolve and change continuously, based on the number of downloads, the retention time of applications on the device, ratings but also the region, local time, local events, and local interests.

7.1. Popular applications

There is no simple definition to define when an application is 'popular'. Apple has top lists per category and per country. As the top rankings change continuously, a random snapshot of the top ranking can result in applications that are only 'popular' for a short time or only at one moment, like an app for a festival. To solve this, we want to list popular applications over a longer period.

To decide which applications are popular over a longer period, an application was written. This application collects snapshots of the top lists at different times. We can use this dataset to get a list of popular applications over a longer period per category, giving us a more reliable overview of popular applications.

Apple shows top lists in the App Store via the mobile App Store application but does not offer the list on its website. The only list Apple provides is a list[27] of which applications exist for iOS, sorted per category and alphabetically. The iOS App Store is a web application using internal APIs. This data can be scraped. Luckily, multiple tools already scrape the Apple App Store to collect and give insight into top lists. One example is AppFigures[28]. AppFigures is a tool for developers to easily track the popularity of applications of the developer and competitors. Other tools are SensorTower[29], App Annie[30], and AppFollow[31].

The initial idea was to use a scraper to get the top list of a tool. However, an API would be preferred to improve the reliability and quality of the dataset. The different tools are reviewed to select the best API to be used, as Apple's dataset is not easily usable. All tools are web applications for developers and companies to track their and competitors' applications. Tracking the position in the top lists, as ratings, reviews and more.

[26] https://moz.com/blog/app-store-rankings-formula-deconstructed-in-5-mad-science-experiments
[27] https://apps.apple.com/us/genre/ios/id36
[28] https://appfigures.com
[29] https://sensortower.com
[30] https://www.appannie.com/en/
[31] https://appfollow.io

- **AppFigures**: AppFigures is offering an API to the users. One of the API endpoints is '/rank' with the description: "Ranks show your standings in the app store. There are two ways to get that data from the API.". This meets exactly the data needed. However, it's limited to given application IDs, tracking owned applications and known applications of competitors, but not the top list itself. The website shows the top list, updated every hour. The website uses JavaScript to load the data dynamically from a private API[32]. This API is only giving the current app store listing and, as far as I could find, it is not possible to define a custom period. It is possible to set a country and category. This internal API can be used if we call it daily and hourly and store all data. Table 6 shows which data is given per application. Note that the wanted data can be set in the query, but as this is a private API, the options are not documented, so brute force and guesses will be required to discover other data.

- **App Annie**: I found traces of an API[33] at App Annie, however, all documented, resources and endpoint were outdated and not working. I found a private API on websites like AppFigures. App Annie has an anti-bot detection, and a simple HTTP request to this endpoint did fail with an HTTP 503. The API was returning data by setting a custom User Agent. Belgium is not listed in the API on the public website, and you cannot request data of a given date. However, when you create an account, those options are available, and it is easy to check the private API usage. The API returns one list with one entry per rank. For every rank, a tuple was set with an entry for the free, paid and grossing app. Table 7 is an example of which data was returned per application.

- **SensorTower**: I did not find traces of an API, besides some GitHub code repositories using the private API used by the web application. This API[34] is returning one big list, with repeatedly a free, paid and grossing app. It is less user-friendly to filter all free applications easily. Still, the API is returning much more information such as ratings, the number of reviews, and even deducted information such as the number of downloads and estimated profit. The API allows retrieving data of a given day, up to 90 days in the past. It is also returning interesting metadata such as the privacy policy URL, EULA URL, and if the app has ads which can be an interesting factor to compare with the findings in this research. See Table 8 for an overview of which data was given exactly per application.

- **AppFollow**: AppFollowing is, of this list, the first service that offers an API[35] that can be used to retrieve the top public lists for a country and a given date. An account is required, and a free tier is available. The API returns every application's position, name, rating and rating count. See Table 9.

[32] https://appfigures.com/_u/api/ranks/snapshots?category=25204&country=BE&count=50&start=0&fields=results,id,entries,name,developer,developer_id,price,currency,storefront,vendor_identifier,category,subtype,timestamp,total_count

[33] https://www.appannie.com/mkt/i/v1/apps/top?category=Overall&country=BE&device=iphone&market=ios

[34] https://sensortower.com/api/ios/rankings/get_category_rankings?category=0&country=BE&date=2020-07-26T00%3A00%3A00.000Z&device=IPHONE&limit=100&offset=0

[35] https://appfollow.docs.apiary.io/#reference/0/api-methods/33.-public-top-charts

- **iOSAppStats**[36]: iOSAppstats is a PHP application that needs to be scraped to use the data. I did not find any underlying API, and getting AppStore Stats of a different day seems impossible. The given data is also minimal.
- **SimilarWeb**[37]: I could not find how Similar frontend code is loading the data via JavaScript.

In conclusion: It was decided to use SensorTower as the data source. This is mostly due to the extended set of available data per application that can be interesting when analysing the research findings. For example, if the app is free, the metadata contains if the app has ads, in-app purchases, ratings, links to privacy/EULA and so on. App Anny is the fallback service, followed by App Followed. AppFigures was only returning current data and was not allowed to return data of a different day and was not usable unless we ran our application for a long time to collect data. iOSAppStats and SimilarWeb would require a scraper, so they are less ideal in this case.

7.2. Excluded applications

Only applications that are accessible in the Belgian App Store are taken into account. This means that popular applications in jailbreak stores or only available via in-house distribution profiles are excluded from this research, as well as applications that only appear in App Stores of other countries. iPad-only applications are also excluded.

7.3. Ranking applications

To know which applications are popular over a longer period, an application was built to collect top lists of iOS applications. This application is available at GitHub[38] and is written in Ruby and using SQLite. The application is threefold. See the README file on the software repo for more technical info and details on the provided scripts.

Downloading data

The first part is 'rankings.rb'. When the script is executed, it will download the top lists of the requested period. The data will be stored in an SQLite file. The script uses the API of SensorTower. The free version of the SensorTower API returns data for the last three months. Any request for older data will fail with "*HTTP 404 - Upgrade to Paying Plan*". The application has no support for authenticated calls, which could be future work if a larger data set is required.

Generating top list

The second script, 'toplists.rb', is developed to analyse the database and create the list of *popular* applications. A top list can be generated per category. The category can be 'free', 'paid' or 'grossing'. There is no exact definition of when an application is called 'popular'. The top list of Apple is changing continuously. Applications can be ranked 1 for one day, or an app can be ranked 10th for a whole month. The top lists provided by Apple are ranking

[36] http://www.iosappstats.com
[37] https://www.similarweb.com/apps/top/apple/store-rank/us/all/top-free/iPhone/
[38] https://github.com/jelledelaender/OUAppStoreRankings

applications based on multiple factors. The exact conditions are not known to avoid abuse of the ranking. The application's position is a key factor in our popularity calculation. The top lists are updated frequently by Apple, resulting in changes in positions. Some applications are more popular on weekdays, while others are more popular during the weekend or on holidays. For this reason, we are making a copy of the top list over a longer period. We can calculate the lowest, highest, and average rankings for every application. The average position is a great start for the popularity algorithm, with a lower score meaning more popular.

$$popularity\_score = \frac{\sum_{n=1}^{number\_of\_occurrences\_app} app\_position[n]}{number\_of\_occurrences\_app}$$

Note that the top lists that were collected are limited. By default, only the top 50 is being collected. Applications with a higher position are excluded. The algorithm above is a great start but does not consider how often an application is included in the top list.

An application can be one time in the top list, with position 30, or multiple days in the top list with an average position of 30. The latter is a more popular application than the first example, so the number of occurrences needs to be taken into account. One option is dividing the score by the number of occurrences, so applications that appear more often will have a lower number and result in being more popular, which defines our algorithm as:

$$popularity\_score = \frac{\sum_{n=1}^{number\_of\_occurrences\_app} app\_position[n]}{number\_of\_occurrences\_app^2}$$

Evaluation: To verify this popularity score algorithm, a database of the top lists from 25 April to 24 July 2022 was generated. The following top list was generated of the free iPhone apps. The list shows the position in the long-term popularity list, the app name, the minimum position, the average position, the max position, the number of days in the top list of Apple (in the last three months) and the popularity score at the end.

	Name	min pos	avg pos	max pos	# days	score
1	GovApp	1	1.771	7	83	0.021
2	Google Maps	2	5.978	13	89	0.067
3	NMBS : Trein info & tickets	2	6.371	24	89	0.072
4	BeReal. Your friends for real.	1	6.843	26	89	0.077
5	SHEIN - Online Mode	1	9.865	29	89	0.111
6	Waze navigatie & Live verkeer	4	12.360	28	89	0.139
7	TikTok: Video's & Muziek	4	12.607	24	89	0.142
8	Google	4	13.090	23	89	0.147
9	WhatsApp Messenger	5	13.528	41	89	0.152
10	Payconiq by Bancontact	2	14.933	31	89	0.168
11	Instagram	5	16.921	43	89	0.190
12	Subway Surfers	2	17.306	47	85	0.204
13	Spotify - Muziek en podcasts	11	21.022	44	89	0.236
14	Booking.com Reisdeals	5	21.506	42	89	0.242
15	Snapchat	8	19.817	48	82	0.242
16	YouTube	9	19.377	50	77	0.252
17	Telegram Messenger	3	22.719	48	89	0.255
18	Stumble Guys	2	17.345	46	58	0.299
19	Facebook	15	24.571	50	77	0.31

20	Klarna | Shop now. Pay later.	12	28.345	48	87	0.326
21	Gmail - E-mail van Google	18	29.023	50	86	0.337
22	NewProfilePic Picture Editor	1	5.133	32	15	0.342
23	WeTransfer	9	26.936	50	78	0.345
24	My bpost	7	24.194	50	67	0.361
25	Google Chrome	19	31.952	50	84	0.380
26	Vinted: Tweedehands shoppen	9	31.880	50	83	0.384
27	Messenger	18	27.944	48	72	0.388
28	Too Good To Go	1	27.081	49	62	0.437
29	Netflix	15	30.848	50	66	0.467
30	Disney+	17	36.000	49	76	0.474

Table 2: Top free apps in Belgium for iPhone, in May, June & July 2022

In Table 2, we find that *GovApp* is the most popular application, with a popularity score of 0.021. *GovApp* was not on the top list every day, only 83 days versus 89 days for other apps. However, the position was always between 1 and 7, with an average of 1.771. *Google Maps* is ranked number 2, and never hit position 1 (in the global Belgian App Store ranking), but was present every time in the top list. Number 3 is the Belgian railway company with a slightly higher average position and has been present for 89 days.

We can see that the popularity list has a combination of applications that are for a long time in the app list and applications that are for a shorter time in the App Store top list but did have a better position, which was the main objective of the defined algorithm.

Future improvements: To improve the algorithm to define the popularity of an application could be improved further. For example, the user application ratings could be taken into account. The ratings of previous versions. The number of ratings. The estimated number of downloads and more. Those options could be evaluated as future work but left out of the scope for defining the list of popular apps for this research. The main focus is to research the privacy aspect of the applications rather than defining the best popularity algorithm.

App details

Lastly, a small script, '*appinfo.rb*', can be used to get more detailed information about a listed application, such as the rating, ranking, privacy URL (if any) and more. See Section D for an example output of the NMBS application.

7.4. Selected applications

Using the ranking application described in the previous section, a database was collected with the top lists of the Belgian App Store over May, June and July 2022. Table 2, Table 3 and Table 4 show the top lists by using the algorithm as defined above. We have 3 top rankings as the rankings are generated for free, paid, and most grossing applications.

Based on the generated list of popular applications, a selection was made with applications in scope for the research. See Table 5 for the selected applications. There are 50 applications selected. Figure 8 shows the distribution over the monetisation models. Figure 9 gives an overview of the number of apps per app category. The game category is the biggest category with 23 applications.

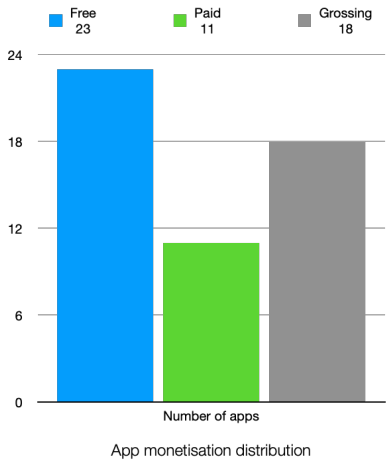

Figure 8: App monetisation models of selected apps

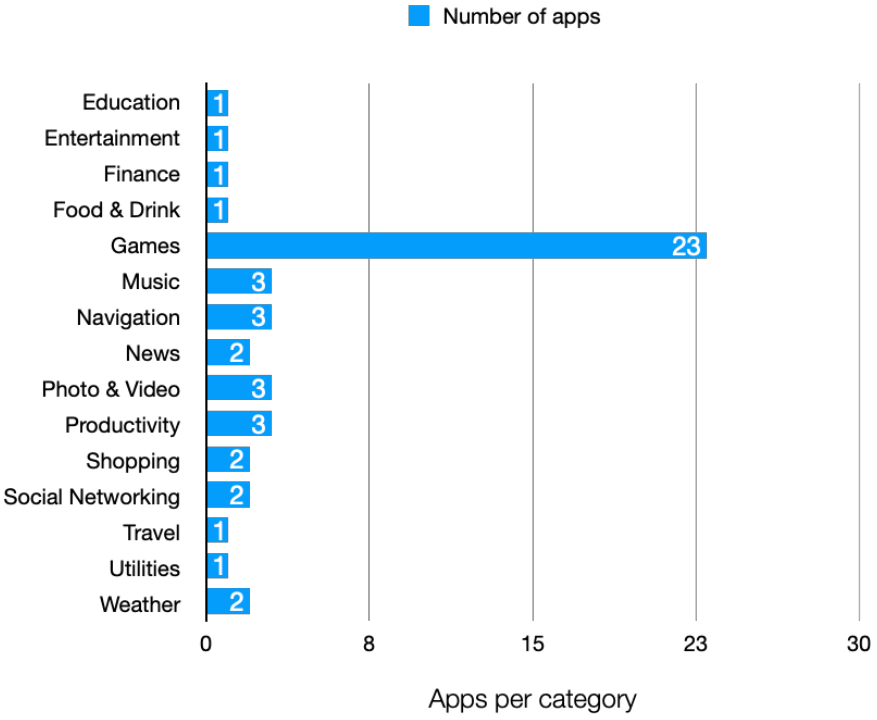

Figure 9: Apps per category of selected apps

8. Results & discussion

For this research, 50 applications, as defined in Section 7.4 and listed in Table 5, were analysed by intercepting and logging all traffic via a proxy setup, as described in Section 6.1. This data was processed manually and by custom-developed scripts[39], to support answering the research questions.

Of the 50 selected applications, one application, Geometry Dash, crashed at startup on the test device. This was unexpected, as it is considered a popular application, and the free counterpart, Geometry Dash Lite, works perfectly on the test device. For this reason, the actual number of tested applications is 49 as no analytics could be performed on this one application.

Remark: While analysing the intercepted data, it was detected that not all data was intercepted. For this research, an HTTP proxy server was used, which only intercepts the network requests over the HTTP protocol. It was detected that different protocols, such as XMPP[40], were used in at least two applications, BrainToss and Whatsapp. Our focus is limited to findings by analysing the HTTP traffic. This is discussed in more detail in Chapter 10. The missing requests are also confirmed by comparing the detected domains on the proxy and the DNS. See Figure 10, where the proxy data is shown on the left and the DNS analysis on the right. Domains such as *pps.whatsapp.net*, *chat.cdn.whatsapp.net* are missing on the proxy logs.

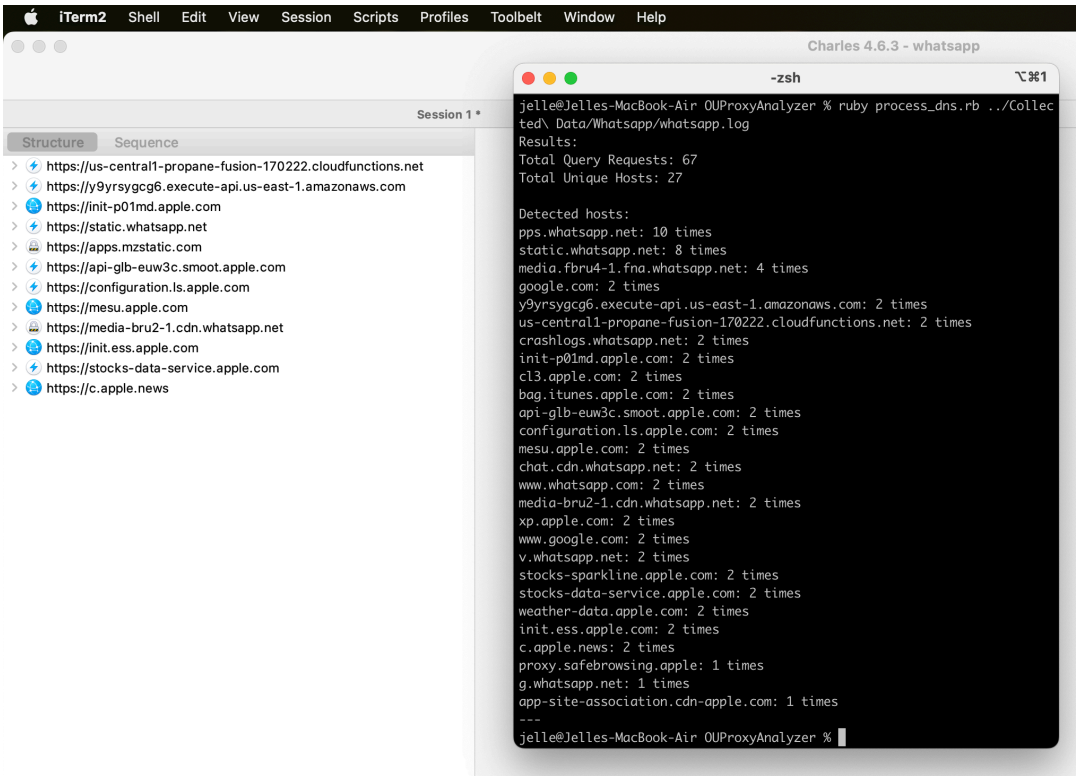

Figure 10: Overview of detected domains via the proxy on the left, and DNS logs on the right

[39] https://github.com/jelledelaender/OUProxyAnalyzer

[40] XMPP or Extensible Messaging and Presence Protocol, is designed for real-time exchange of messages like chat systems, and keeping track of presence status

8.1. Use of encrypted connections

A lot of applications exchange data with servers. Some applications require data exchange, such as social applications or applications showing the latest available data. Other applications are syncing data for a better user experience and allowing users to have the same data on all the users' devices or tracking how the application is used.

Of the selected applications, we did not find any applications that did not connect to servers. All applications had at least one network call. 8 of the 49 applications had more than 1.000 network requests. 13 applications had less than 100 network requests during the test session.

When network requests are performed over plain text HTTP, other computers and network devices could intercept and get insight into the exchanged data, possibly harming privacy. A secure protocol such as HTTPS is essential to protect the privacy and ensure no tampering.

In our testing, 29.253 requests were captured. 2.934 were not encrypted. When filtering out the OCSP requests, which are in our dataset all performed over plain text HTTP requests, there are 2.888 unencrypted requests, or 9,87% of all requests are not encrypted.

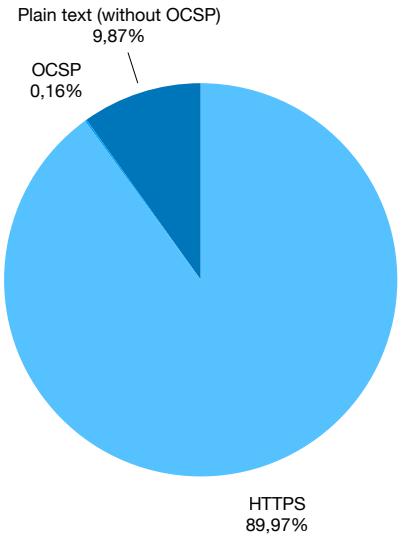

Figure 11: HTTP vs HTTPS requests distribution

Apple enforces HTTPS in the applications via Apple Transport Security (ATS)[41]. Developers can still whitelist domains that can bypass the Apple Transport Security requirement. ATS is only enabled for applications built against iOS 9 or newer, so applications not updated after 2015 will not have ATS enabled. ATS only impacts resources requested via the app itself. ATS is not limiting links that are opened by the browser. In our test, as all HTTP traffic is intercepted, those links will be detected when clicked, as all browser traffic is intercepted and processed via the proxy.

[41] https://developer.apple.com/documentation/security/preventing_insecure_network_connections

13 of the 49 applications did perform minimal one unencrypted request, excluding OCSP requests. This is around one-fourth of the applications in scope, which does sound quite alarming. The requests need to be analysed more deeply to know the impact on privacy.

- **Legal Documents**: 4 applications provide and load links to legal documents, such as the Terms Of Service, Privacy Procedure, and End User License Agreement (EULA), over the insecure protocol. This was the case for *GTA SA*, *Stumble Guys*, *Geometry Dash Lite*, and *Earn To Die 2*. No personal user data was involved in those requests.
- **CDN Files**: Applications are loading static files of CDN servers. For example, unencrypted requests to CDN servers were detected for *GTA SA*, *HLN*, *Geometry Dash Lite*, *State of Survival: Zombie War*, *Deezer* and the *Rise of Kingdoms*. No personal data was involved. However, in the case of *HLN*, JavaScript files were loaded, which could harm privacy as there are no integrity checks for data loaded over HTTP. The JavaScript could be tampered with, resulting in the app running unexpected JavaScript code.
- **Certificates**: *Payconiq* is using certificate pinning technique. 9 unencrypted requests were detected, of which three were for OCSP checks. The other six requests are part of the certificate pinning process, downloading the expected CA certificates. No personal data was involved.
- **Dev Servers**: *Street Kart Racing Game* performs insecure requests to a dev server of the developer. No personal data is being sent. The three detected requests are downloading the game config, such as rolling friction coefficients of the tires, such as the tire wear for slip, per kilometre etc.
- **API Servers**: *Plague Inc* and *State Of Survival: Zombie War* are making unencrypted requests to API servers. Mostly to report and collect device metadata, such as the device model, platform, iOS version, screen resolution, memory usage, type of CPU/GPU, the device ID, and tokens for push notifications.
- **Photos**: N*ewProfilePic Picture Editor* allows users to create a new profile picture by uploading a selfie which is then processed server side to a profile picture by applying some effects. All requests are over HTTPS, except the API request to upload the original selfie and the request to download the modified image. The server is also not supporting HTTPS. No personal data besides the image was found, but this is still unexpected and could harm the user's privacy as any user on the network could intercept such data.

The number of applications still having non-encrypted traffic, one-fourth of the applications in scope, is luckily a much better number than the research of Orikogbo et al. [2016] in 2016, where 98% of the apps still had unencrypted network calls. The difference can be explained by the adoption of TLS, and the enforcement of ATS by Apple. Table 10 summarises all captured requests during the tests.

In conclusion: One-fourth of the selected applications are making insecure network requests. This represents 9,87% of all intercepted requests. *New Profile Pict* exposes potentially personal data by exposing the user's uploaded photos. *Deezer* is exposing which songs a user is listing to by exposing the requests to the album and playlist images. This

is no personal data, but can still expose the music preferences of the user. 2 games send device metadata that can be used to identify a user, unencrypted over the internet. One app, *HLN*, is loading executable JavaScript files over an insecure connection and executing those, which could be risky as the content could be changed. Remarkably, most servers do have HTTPS set up correctly. However, the applications still use the insecure variant. This makes it relatively easy to fix, so the main question is why this is not done yet, especially as these are *popular applications*.

8.2. Certificate pinning

TLS Certificate pinning is expected on applications where data confidentiality and integrity are essential, such as applications of banks and governments. When TLS certificate pinning is used, TLS resigning via a proxy is impossible, depending on how the certificate pinning is implemented. This results in a higher security and trust level of all exchanged data on the network.

Of the 49 applications, eight applications had some level of certificate pinning implemented or 16,32%. While 49 applications are a relatively small data set, it matches the expectations and findings of the research by Pradeep et al. [2022] where they found that 11% of 2515 iOS applications do have certificate pinning enforced last year. See Table 11 for the certificate pinning status per application.

In the list of selected applications, most applications with certificate pinning enforced are social-community focused: TikTok, BeReal, Snapchat, Whatsapp and Pokemon Go, or bank-related applications such as Payconiq.

Spotify has a partial integration of certificate pinning implemented. From the testing, the login API call was failing over the proxy. However, playing music, creating an account and so on were possible. It seems only the login API was provided with extra security measures, such as TLS certificate pinning.

My Bpost, on the other hand, an application of the Belgian National postal service to track parcels, make payments, create shipping labels and more, has a strict certificate pinning enforced. This was initially unexpected. However, as you can manage payments, manage postal preferences and create shipping labels, it does make sense and ensures a high level of security and privacy protection.

GovApp is an application of the Belgian government, focusing on providing a safe alternative to SMS messages for communication and notifications of the government and authorities, as described on GovApps' Apple App Store page[42]. Certificate pinning was expected but not implemented or enforced.

In conclusion, a small subset of the applications use certificate pinning to some extent. In most of those applications, this was expected due to the nature of the application, such as social applications. One application, GovApp, did not have certificate pinning where I would have expected this due to the focus on government communication. While certificate pinning makes it hard to intercept which data is exchanged, detecting which servers and endpoints the application calls is still possible.

[42] https://apps.apple.com/be/app/govapp/id1620323239

8.3. TLS validation

In the previous Section, we checked which applications had TLS certificate pinning enforced. This was done by checking which applications reject the proxy server's certificates while the proxy CA root certificate is trusted on the device. In that test, we found eight applications that have TLS certificate pinning implemented. All other applications did accept the new resigned TLS certificate of the proxy.

By repeating the experiment without marking the proxy CA root certificate as trusted on the device, it is possible to detect which applications are accepting *any* TLS connection without verifying the chain or validity of the certificate. Those applications use HTTPS but still can harm privacy due to the lack of validations. Those applications are also vulnerable to Man-In-The-Middle (MITM) attacks.

It's expected that no applications are vulnerable to those attacks and have the baseline of TLS validation enabled. This is offered out of the box by the iOS SDK and iOS frameworks[43].

See Table 12 for the results of the TLS validation tests.

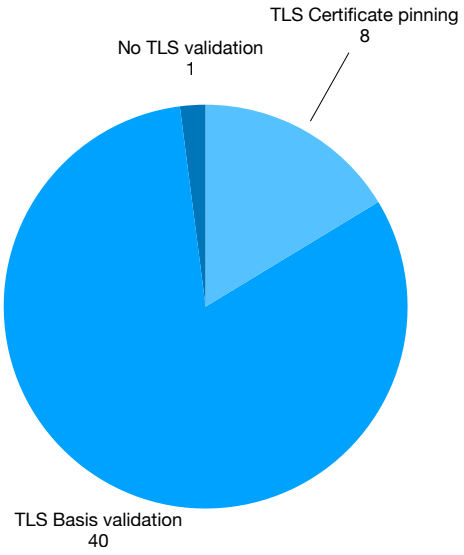

Figure 12: Overview of TLS validations and levels

One application, *Rise Of Kingdoms*, is vulnerable to the MITM attack as it did accept all HTTPS connections even with an untrusted CA root certificate on the test device.

A second application, *Coyote*, did detect that the connection did not had a proper TLS certificate and did not allow us to login. However, once we were logged in, the application was downloading map data from CDN servers without validating the TLS certificates and

[43] https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/SecureNetworking/SecureNetworking.html

did accept data from this server even when an invalid TLS certificate was used. As a result, tampered maps could be downloaded, or map requests could be intercepted which can reveal the location of the user.

Via this setup, it was also found that *BrainToss* was still functional. However, no network requests were captured on the proxy. Identical for *Whatsapp* where chat messages still worked. Photo uploads and other API calls over HTTP failed on Whatsapp. Part of those data exchanges is not using the HTTP protocol and is therefore not intercepted and resigned by the proxy. As a result, we cannot safely state if those requests are appropriately encrypted and to what extent. See Section 10 for more info.

In conclusion, we can state that almost all applications in scope have the baseline of TLS validation enabled and enforced, except for two applications. This could harm privacy, as when a MITM is performed, the application will not detect this, and different data could be used and processed for the user. This also means the application will not detect when data is sent to an untrusted server.

8.4. Tracking techniques

Previous sections focus on the security measurements taken to protect data exchanges over the internet, preventing other parties from intercepting or tampering with data. Even when only encrypted connections are used, and all connections are verified by a strict TLS validation or TLS certificate pinning, user privacy could be harmed by tracking the users without consent. This section and the following section will focus on the tracking aspect. Are applications tracking every click and action? This section lists techniques that can be used to track users. The list is created by online research and experience. The list is not complete but covers the most used techniques. Most techniques apply to websites and mobile applications. However, some techniques require access to the device or network, like tracking via a proxy or DNS server.

Tracking can give insight into the application's use and under which conditions. For example, which network connection users have, the type of device, the behaviours of the users and flows through the application. This information can help improve applications and user experiences. Tracking users can also get more information about the current user to optimise and personalise advertisements to improve the monetisation model.

Developers and company owners are tracking application users for multiple reasons.

Multiple techniques can be used to track people and users. An initial step is identifying users to increase the dataset's value and tracking.

Apple introduced in iOS 2 a method 'uniqueIdentifier'[44] of the 'UIDevice' class that could be used to get a unique identifier of the current device. Developers could use this identifier to recognise a device. This identifier was static for the device and never changed. Any application on the device got the same identifier, making it easy for developers with multiple applications or frameworks installed on multiple devices to map which applications were installed by one user.

Seriot [2010] gave in 2010 a presentation[45] at the Black Hat DC, a well-known computer

[44] https://web.archive.org/web/20140703160701/https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIDevice_Class/DeprecationAppendix/AppendixADeprecatedAPI.html

[45] https://www.blackhat.com/html/bh-dc-10/bh-dc-10-briefings.html#Seriot

security conference, about the privacy issues at iOS. Seriot's paper describes how the unique device identifier can be used to spy on users and gives Apple recommendations, for example, that iOS should ask explicit permission before the application can access data from the GPS sensor as the device identifier.

Apple deprecated this method in iOS 5 and fully removed it in iOS 7 as the method was *abused* by tracking frameworks to get more insight and spy and track on users. Apple did replace the method with three new methods that are more privacy-friendly, which was announced at WWDC in 2014. Stites and Skinner [2014] gave a presentation about the new API, see Image 13. The new identifiers are unique per developer and only shared between vendor applications and will only be, potentially, reset when the user uninstalls all applications of one vendor. There is also an advertising ID which can be disabled and reset at any time by the user, giving the user control of tracking by an identifier.

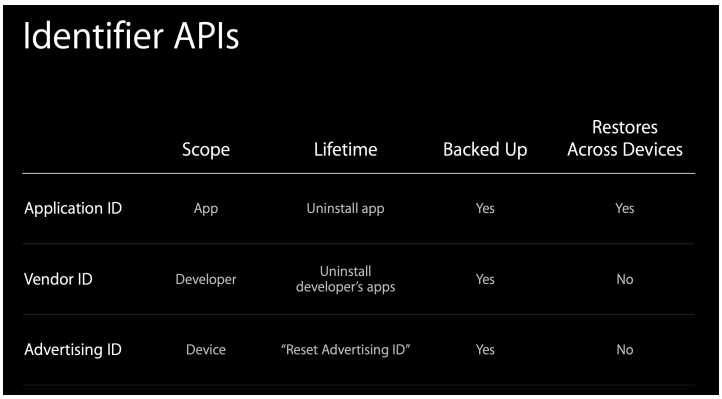

Figure 13: Replacement method of uniqueIdentifier in iOS 5, with scopes of the tokens.
© Apple Inc - Source: Apple Developer Documentation

When no user ID or device ID is available, it is still possible to identify and track users via different techniques. One of them is fingerprinting.

Device fingerprinting: The fingerprinting technique can be defined as collecting a bigger dataset of configuration settings of the user, such as hardware, software and sensor information, which can be converted to an ID by an algorithm and can be used as an ID to track the user. Think about the device model, current version of the OS, screen resolution, list of installed fonts, device language, browser settings and capabilities. Apple is reducing the amount of metadata available by applications to make fingerprinting harder.

Zhang [2021] recently performed detailed research about this technique on mobile devices, which can also be (ab)used by browsers via JavaScript. One of the conclusions of Zhang was that they could easily identify a device, even after a factory reset, with a dataset of fewer than 100 samples of sensor data.

As fingerprinting uses data that is globally accessible on the device, it can be used to track users and devices in multiple applications and websites.

Cookies & Session storage: The well known way of tracking users on the web is via cookies. There are two different types of cookies, server-side cookies and client-side cookies. Cookies are small text files, small databases that can store data that will be available at every request and are stored during the session or for a longer period. Cookies can be used for functional purposes, which are needed to ensure the website is working. Think about storing information such as the current state. Cookies can also store identifiers to track and identify

users.

While cookies are scoped to the current domain of the website, they can reveal and leak more information and details of the current users when common third-party resources are included. If multiple websites use a shared tool like Google Analytics, there is a chance that this tool can link the visits of multiple websites to the same person. Englehardt et al. [2015] performed deep research about cookies and how those can be used to track users.

Luckily, browsers that care about privacy offer features to prevent cross-site tracking via those cookies. Apple added more technical limitations over those cookies in Safari 13.4 on iOS[46] which is enabled by default and limits the possibility for cross-site tracking.

With HTML 5, browsers have support for local storage. Cookies are limited to 4096 bytes, whereas HTML5 local storage has a limit of typically 5MB. A user can allow a higher limit when requested by the website or web application. HTML5 local storage can be used like cookies and used to read and store values that will be persisted between sessions.

The ePrivacy Directive[47], also known as the European Union cookie law and later extended via the GDPR, requires explicit user consent before data can be stored that can link and track users and behaviours.

Web Beacons: A web beacon is a technique to know if a user accesses a resource. This is often done via an invisible pixel on a page or email. When the page is loaded, the email is loaded, and the resources, including the invisible pixel, are loaded. The beacon has parameters based on the session, user, and email, allowing the server serving the resource to track this event and save metadata.

This is also implemented in applications to track if events are triggered and when users are opening views.

By using beacons. Application owners and developers can track if a user is opening emails and messages and which views the user is opening, even when the user is not authenticated.

Heatmap: Developers can track all clicks and actions on the application and websites. This will generate a heatmap that will make it visually clear where users click. See Figure 14 for an example, where the red zones are the most clicked areas and the blue zones less. This feedback is helpful for developers to verify if the user interface is not too complex. The heatmap will also allow the developer to verify expectations. Are users clicking on headers and expecting it to be a link or button? Which button is clicked in case multiple buttons have the same action? Is the call-to-action (CTA) button the most used button? With this behaviour information, additional metadata of the user, such as the language and type of device, can be collected and stored.

Session Replay: Related to the heatmap, another tracking technique is recording a full user session. This can be seen as recording the screen while the user uses the application or website. This will include the scrolling behaviour, the location of the mouse, which is often the attention point, the clicks, the time on every section and the complete flow of how a user went from one place to another. It is called the *session replay* as the developer can replay the complete session of the user and see anything he did.

This technique can be considered privacy intrusive as those sessions contain user data potentially considered private or internal data such as Personal Identifiable Data (PII). Some tracker services anonymise data to be GDPR 'compliant', like detecting and blurring or hiding email addresses and names.

[46] https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/
[47] https://gdpr.eu/cookies/

Figure 14: Example heatmap - © Heatmap JS - https://www.patrick-wied.at/static/heatmapjs/

Chairani et al. [2019] performed research and worked on a method to calculate a privacy scoring system related to tracking techniques like session replays.

Proxy: In this research, described in Section 6.2, will use a proxy to intercept all data and get insight into network traffic. This can be used on any network to track the users on the network. This works for unencrypted, but also on encrypted traffic, as long as no TLS certificate pinning is enforced and there is admin access to the device in scope.

Routing traffic via proxies can be privacy-sensitive, as all user data could be intercepted and copied to untrusted sources. This is often the case in corporate networks, to have strict network monitoring and controls.

Apple announced they would add a private VPN solution called "Private Relay" to prevent proxies from intercepting and sniffing on personal data[48].

DNS: By default, any computer is using the DNS resolver of the network gateway. This will allow the network manager or internet service provider (ISP) insight into which applications or websites someone uses.

This is not always a problem as only the domain will be known and not any data such as form data, but even this level of information can be considered as harming the user's privacy.

A user could use a different DNS service to resolve this issue, such as CloudFlare 1.1.1.1 DNS resolve, which aims to be privacy-first. However, as the DNS protocol is over plain text and not using any encryption, any server on the route could intercept and get still an insight into which domains are requested.

A better solution to this tracking is using DNS-Over-HTTPS (DoH), where the DNS request is fully encrypted and only known by the DNS server and the user. This is listed in Section 6.4 as we will use a custom DNS server to validate the findings via the proxy in this research. Apple did add support for DoH in iOS 14 and macOS 11.

Internet Service Provider: Unless data is fully encrypted, the certificates are verified, any ISP can intercept data by performing man-in-the-middle attacks, see all user traffic, and gain insights and track users. The ISP can also detect and map the Media Access Control (MAC) address of the user's modem to the user to track the user. Using a secured VPN is a solution to secure data connections.

[48] https://developer.apple.com/videos/play/wwdc2021/10096/
https://www.apple.com/newsroom/2021/06/apple-advances-its-privacy-leadership-with-ios-15-ipados-15-macos-monterey-and-watchos-8/

Inline Browser Trackers: Some applications have links to external websites. Those are often opened in an inline browser and not in the native browser on the device, so the user will stay longer in the initial application and go back to the app when the inline browser is closed. Felix Krause, a security & privacy researcher, discovered in August 2022[49] that some applications are modifying the loaded websites in such inline browsers by adding extra tracking code to tracking services of the application owner. Related to the finding of Felix Krause, it was detected that Tik Tok injected code that could monitor and log keystrokes[50]. The official answer is that this code was not activated, and only part of a larger debugging project that was only used for troubleshooting and debugging issues.

8.5. Tracking services

By tracking users, companies can have valuable insight into who is using the application, how the applications are used, which features are used, how often, if any bugs or exceptions happen and more. This helps in, e.g. optimising the applications, increasing the user experience, and optimising ad revenue. The types of tracking can be categorised into multiple categories, each with a different purpose:

- **Ads**: Monetising applications by showing ads. Tracking users can optimise ads and increase profit, as relevant ads can be shown. Users can also be retargeted by showing ads of websites the users visited before if the user can be linked to other websites and application sessions.
- **App performance and bug trackers**: Tracking the type of devices used and technical details. Often also collecting exceptions and crashes of the application so that the bugs can be fixed, the app is optimised for the used devices, resulting in a more stable application.
- **App usage**: Tracking which features and how the application is used. Insight into which features are used, tracking which buttons are pressed, and type of devices used such as CPU, GPU, available memory, network connection and more.
- **Game-focused trackers**: Same as app usage trackers, but optimised for games, like tracking game levels.

The list above is not complete, as other types of tracking exist. For example, some trackers are used for profiling users, for anti-fraud systems and anti-spam like Google Captcha. Most trackers fit into one or more categories as they offer many features. For example, most trackers do offer a baseline of app usage trackers. It is also important to state that not all tracked data is always shared with the app companies. For example, ad trackers track data to optimise ads by showing relevant ads to the current user but don't always share all tracker data with the company behind the app.

Below is a list of available tracker services per category, compiled by research. The list is not complete, but it should list the well-known services.

[49] https://krausefx.com/blog/announcing-inappbrowsercom-see-what-javascript-commands-get-executed-in-an-in-app-browser

[50] https://www.forbes.com/sites/richardnieva/2022/08/18/tiktok-in-app-browser-research/

Ads Trackers:

- **Facebook**: Using the Facebook SDK, users can be tracked to improve ads and marketing efforts, like tracking sales, user retentions and more.
- **Flurry**: Flurry is an analytic and advertisement platform acquired by Yahoo.
- **Countly**: Mobile app analytics as marketing analytics.
- **Vungle**: Vungle is an in-app video platform for performance marketers.
- **SupersonicAds**: Supersonic is a mobile game publisher to help scale games and profitability.
- **UnityAds**: The Unity Ads SDK provides a monetisation framework for games.
- **AdColony**: A mobile monetisation and advertising platform.

App Usage Trackers:

- **Google Analytics**: Google Analytics is a tracking product of Google. It is mostly known for tracking websites. As some applications are web applications, it can also be used to track applications. Google Analytics allows to trigger events and follow users in the apps like on web pages.
- **App-Measurement**: Firebase was a company Google acquired in 2014. Firebase is a set of frameworks that allows developers to implement stuff more easily. One part is analytics via Google Analytics for Firebase. "Analytics reports help you understand clearly how your users behave, which enables you to make informed decisions regarding app marketing and performance optimisations"[51]. This tracking is linked to Google Analytics and collects data via the 'app-measurement' domain.
- **App Analytics by Apple**: Apple is also tracking users and crashes. When a user configures his device for the first time, the user can decide to share data with the developer. Crash information but also metadata and usage data. This data is available to the developer via the developer portal.
- **UXCam**: Deep analytics, user flows and KPI monitoring. Including video recording solutions where developers can see what the users did exactly. Support for heatmaps. Error and crash collections.
- **Segment**: Detailed event tracking and customer tracking software. Including methods to enrich the profiles of the users to get more business insights. Funnels, real-time data.
- **AppLovin**: Tracking of mobile applications. AppLoving provides multiple tracking services. Adjust is focused on tracking app usage.
- **Mixpanel**: MixPanel defines itself as "product analytics for product people", by tracking funnels, engagement and application usage.

[51] https://firebase.google.com/docs/analytics

- **Facebook**: While already under Ads Trackers, the Facebook SDK also provides insight into application usage.
- **Hotjar**: A product experience insights tool providing behaviour analytics by heatmaps, session recordings and more.

App performance and bug trackers Trackers:

- **CrashLytics**: CrashLytics was originally a tool to automatically collect crash logs of users, so developers are faster aware of issues, with all required technical information to fix the issue. CrashLytics was acquired by Twitter and then Google and is now also offered via Firebase.
- **Sentry**: Sentry is an open-source "Events & Issues Monitoring". It is originally designed to keep track of JavaScript exceptions but is also used in native applications. It also has support for events and tracking users' metadata.
- **AppMetrica**: AppMetric offers deep product analytics, such as crash and error reporting
- **DataDog**: Data analytic tool focused on analysing server and application performance.
- **BugSnag**: BugSnag is an error monitoring and reporting software.
- **NewRelic**: Cloud based platform to track the performance of applications and servers.

Game focused trackers:

- **GameAnalytics**: GameAnalytics collects user data and events and is focused and optimised for games and monetising games.
- **DeltaDNA**: Game analytics platform.
- **LuneLabs**: App tracking, focused on games and app usage & ads optimalisation.
- **Unity3D Tracking**: Game tracking module of Unity3D framework.

In this research, trackers were identified by intercepting the data exchanged by the applications. Table 17 shows which trackers were detected. This was performed by collecting and analysing which domains were called by every application in scope.

Figure 15 shows an overview of which category of trackers are detected. App usage trackers are the most commonly detected tracker, followed by ads trackers.

All detected trackers are classified and listed per category:

Ads: Table 13 lists all detected trackers that fall under the ads category. 2 trackers, Facebook and Double Click, are used in half of the applications. In total, 13 different trackers were found in the 49 applications. 68,62% of the applications had at least one ads tracker.

App usage: Table 14 shows that *app-measurement* is used in more than 33 applications, which Google owns. The second most used app usage tracker is Facebook. It is also remarkable that, except for five applications, an app-usage tracker is used in every application,

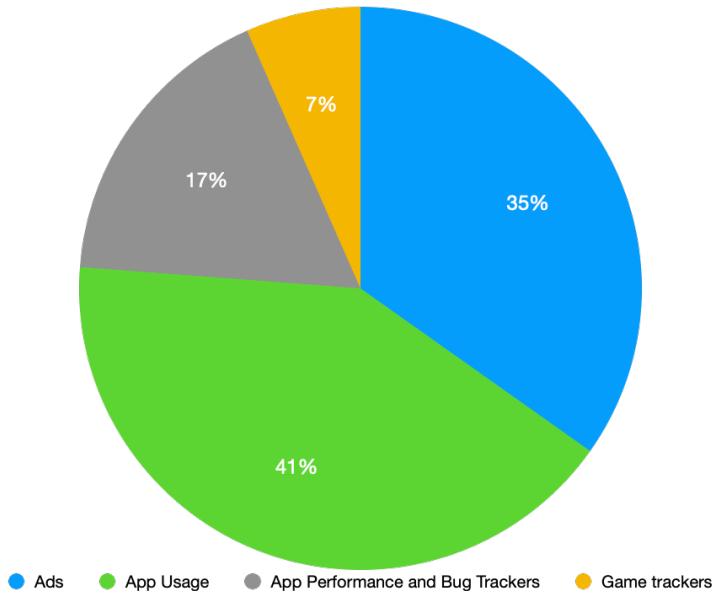

Figure 15: Overview of the types of trackers

or 90.20% of the applications have an app-usage tracker. The only apps without a detected app usage tracker are *GovApp*, *SnapChat*, *Whatsapp*, *Spotify* and *60Seconds*.

Game trackers: Similar as the app usage trackers, Table 16 shows the detected game trackers. Those are only expected in games. Unity3D is the most used game tracker in the given application scope. A game-focused tracker was found in most games, except for Clash of Cans, HomeScapes, Pokemon Go, CoinMaster, Poppy Playtime, EarnToDie, and GTA.

App performance & bug trackers: Table 15 shows that CrashLytics is the most used bug tracker in our dataset and is owned by Google. In 16 applications, no bug tracker was detected. Apple is also offering crash reports and some insight into the applications[52]. This data is directly collected via the iOS platform and is harder to detect. 55.93% of the applications has at least one app performance tracker installed.

Another finding is that there is only one application without any detected trackers in our selection of 49 applications, which is *Whatsapp*. Knowing that Whatsapp did implement a strict TLS certificate pinning, which makes it more complex to inspect all traffic, and the fact we noticed that we are not intercepting all data as some data is also going over XMPP, does not entirely exclude the fact that Whatsapp is still using one or more trackers, and is not excluding no server-side tracking is performed. We could state that every application has at least one tracker installed, which *could* track the user. This research found 227 trackers installed, or on average, five trackers per application.

Some applications also made calls to services such as OneTrust, CookieLaw and CookiePro. Those are services to help to comply with the GDPR law, ensuring consent is requested when required and stored. This was detected in 20 applications. However, in almost all cases, an (inline) browser was opened to show the EULA, TOS or Privacy-Policy and not part of the application itself. There was also no user tracking besides functionally showing the cookie banner and storing the consent.

[52] https://developer.apple.com/documentation/xcode/acquiring-crash-reports-and-diagnostic-logs

The findings of Kollnig et al. [2022a] in the random dataset of 12.000 iPhone apps, is that the median number of tracking services, per application, is 3. 3.13% of the iPhone apps have more than ten trackers embedded, and 79.35% has at least one tracker installed. Binns et al. [2018], who performed a research on 959.000 Android apps, stated that the median number of trackers per app is 10 and 90.4% of the Android apps have at least one tracker.

Kollnig et all also found that Google Firebase is present in 53.9% of the iOS apps. Facebook (or now called Meta) 28.0%. Google CrashLytics 25% of the apps. This represents the same top list as our findings, as in Table 17, as app-measurement is part of Firebase. We do have a higher rate of Firebase usage (64% vs 53.9%), Facebook (49.02% vs 28.0%) and CrashLytics (47.06% vs 25%), which can be due to the selection of applications, and how old the applications are and how actively they are being developed. Our selection is based on popular applications, while Kollnig et all had a random set of all available apps, so potentially older applications and possibly less maintained.

Remark: While checking the collected data, it was detected that some trackers are tracking some unexpected metrics: SuperSonicAds is sending quite some data back to the server. See Table 19. For example, this tracker did keep track of my current city. As GPS access was not allowed, a different method must be used to estimate the location, such as Wifi names[53] or IP Geo database (IP2Geo). The IP2Geo database would not be precise enough. The provider name of my mobile carrier and the name of the fixed internet ISP were retrieved and shared. SuperSonicAds is not the only tracker collecting such data. UnityAds also collects my mobile carrier name and the mobile operator ID, device properties such as the battery state, if the iPhone is jailbroken, if the phone was charging, the number of free memory and more. Tabola was reading the linked country of my SIM card and sending this to the Tabola server and other device properties.

In conclusion, we can state that almost all applications in scope have a minimum of one tracker installed. We can detect four big groups of trackers: *app performance & bug trackers* to ensure the quality and stability of the applications, such as CrashLytics, *app usage trackers* such as app-measurement and Google Analytics, including *game-focused app usage trackers*, and lastly, *ad trackers* to monetise the applications. Knowing that half of the applications have the Facebook tracking SDK installed, and knowing that 70% of the applications use a tracker owned by Google, such as App-Measurement, Google Analytics or CrashLytics installed. We can state that those two companies are getting a lot of data on how applications are used and by who. This is also concluded by Kollnig et al. [2022b], where they state that Apple is making tracking of individual users more difficult but that this also increases the powers of the top trackers such as Google and Facebook, as they have a lot of user data that still can be linked to users due the massive dataset collected by them and fingerprinting techniques. We can also state that some trackers collect unexpected metadata such as the ISP and carrier names, battery states, if you are charging, and how much RAM memory is still available without any consent.

8.6. Differences based on monetisation model

The selection of applications is mixed with free, paid, and top-grossing applications, as shown in Figure 8. One of the research questions is if there is a difference in how privacy is respected based on the monetisation model of the application.

[53] Projects such as `https://wigle.net` maps, a combination of, Wifi names to locations

Use of encrypted connections

As Apple is enforcing HTTPS, and developers can add a TLS certificate for free by using LetsEncrypt or other parties, all applications are expected only to send data via encrypted channels. However, as concluded in Section 8.1, one-fourth of the applications still perform requests over unencrypted channels.

I expect those to be mostly by free applications, as those applications are often less maintained than paid or well-grossing due to less revenue and being less important or side projects.

	Number of apps	
Free	5	21,74%
Paid	4	36.36%
Grossing	5	27,78%

Looking at Table 10 and the table above, 21,74% of the free applications do have unencrypted data exchanges, 36% of the paid applications, and 27,78% of the grossing apps. Surprisingly, I found relatively more unencrypted connections in paid applications. One explanation could be that our data set is too small, as we only have 11 paid applications in scope. One of them, GTA SA, has not been updated for a long time. If we exclude this app, the number would be 27% and more equal to the other categories. *In conclusion*, we can state that the occurrences of unencrypted traffic, excluding OCSP requests, are almost identical for all monetisation models.

TLS certificate pinning

In the applications in scope, all applications with TLS certificate pinning enforced are free applications. 2 applications are also in the grossing category but are initially free. This can be explained as TLS Certificate pinning is expected in bank and social applications where integrity and authentication are essential and less related to the monetisation model of the application.

TLS validations

Luckily, almost all applications in scope have enforced a baseline of TLS validation. Only two applications did accept traffic from untrusted sources. Both applications are in the grossing category. *Rise of Kingdoms* did accept all data from all locations. *Coyote* has TLS validation on most API calls but did soften validation for data from the map data server of *here.com*.

Tracking

Table 18 shows how many applications, per tracker category and grouped per monetisation model, have at least one tracker implemented. This is visualised in Figure 16. A first impression is that most grossing applications have tracking services implemented.

Ad trackers, 60% of all free apps, 50% of paid apps and almost 90% of the grossing applications have at least one ad tracker. This makes sense for grossing apps, as there is often an InApp purchase option to disable ads. I expected the number of ad trackers in paid apps to be lower. However, this insight helps to optimise ads when the paid app has a free version and helps to increase the global application's monetisation.

All grossing applications are tracked via **app usage trackers**, followed by 90% for paid apps and 82% for free applications. Globally we can state that almost all applications use app usage trackers, especially as Apple also offer an app usage tracker at the system level, which is not included in our results.

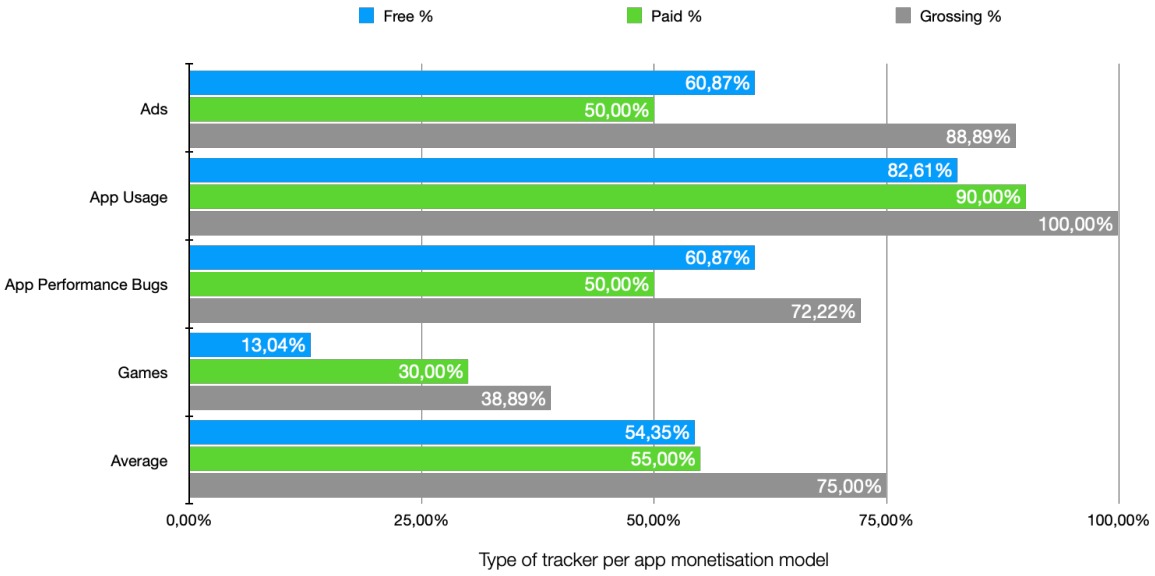

Figure 16: Overview of the types of trackers per monetisation model

Tracking bugs and app performance is mostly done in grossing apps (72%), free apps (60%) and least in paid apps (50%). I would have expected a higher rate on all categories, as such reports help improve and increase the app quality. Luckily Apple already keeps track of crash reports when the user consents. Those reports of Apple are limited to application crashes and do not include warnings and errors that occur without an app crash, often covered by other tracking services.

Conclusion

Based on this data set, we can conclude that grossing apps have most tracking services implemented. Most apps in the top-grossing list are initially free to attract a bigger audience and get revenue via ads or InApp payments. Serving ads, tracking app usage, and fixing performance and bug issues are important to this monetisation strategy. Ads and performance trackers' second biggest group of apps is the free app group which is expected as the monetisation model is often ads based. I did expect fewer ads trackers in paid applications.

9. Conclusions

With the results and analyses of all collected data in the previous section, the subquestions of Chapter 5 can now be answered. 49 applications were manually tested to check which trackers are used, how many trackers are used, to verify the TLS validation level and encryption status, and if TLS certificate pinning is used, and which data is being exchanged.

RQ1: Which tracking techniques exist to track users on iOS?

Apple initially made it easy to track users by providing a unique device ID (UDID) accessible by developers in all applications. This UDID could be used to track users across multiple applications. Apple has replaced this UDID with other IDs such as the *Application ID*, *Vendor ID* and *Advertising ID*, see Figure 13. Each with a different scope and lifespan. The Advertising ID is the only ID that is shared between applications. With the introduction of ATT in iOS 15, access to this Advertising ID is controlled and not accessible when no

explicit consent is given. However, tracking and identifying users via different methods are still possible.

Section 8.4 lists in detail different tracking techniques that can be used to track users on mobile platforms like iOS.

RQ2: Which tracker services exist to track users on iOS?

Multiple tracking services such as Google Analytics and Facebook exist to track users and provide a platform that is often easy to use and install for developers and gives a great insight into app usage, app users and more. Such trackers can be grouped into four categories: *ad trackers*, *app usage trackers*, *game usage trackers* and *app performance & bug trackers*. Section 8.5 lists a set of popular tracking services per category.

RQ3: Which tracking services can we detect in our sample set?

Tables 13, 14, 15 and 16 shows which tracker services are detected for every application in scope. It was concluded that every application has at least one tracker and, on average, five trackers per application.

Table 17 lists all tracking services and the number of times the tracker service was used. Half of the applications use the Facebook tracker, 70% of which use a tracker owned by Google, such as Google Analytics, Firebase or CrashLytics.

The most used tracker type is the app usage tracker. 90.20% of the apps in scope had an app usage tracker installed. The top 3 app usage trackers are Google Analytics via Firebase (64.71%), Facebook (49.02%) and AppsFlyer (25.49%).

The second top category is ads trackers which were found in 68,63% of the applications. Facebook (49.02%) is the most used ads tracker, followed by DoubleClick of Google (39.22%) and UnityAds (17.76%).

App performance and bug trackers were detected in 55.93% of the applications. The most popular app performance tracker is CrashLytics (47.06%), followed by Sentry (11.76%) and DataDog (7.84%).

See Section 8.5 for a deeper analysis of the detected tracker services.

RQ4: To what extent do iOS applications use secured connections?

It was found that one-fourth of the applications in scope are making insecure network requests. This represents 9,87% of all intercepted requests. See Section 8.1.

New Profile Pict is uploading images in plain text to a server which are usually selfies and considered personal data, as the user is identifiable. *Deezer* exposes metadata of the songs and albums played by the users by loading album and playlist images over plain text CDN requests. Two games send device metadata containing data that could identify a user over HTTP. Lastly, it was found that *HLN* is loading JavaScript files executed in the app over insecure connections, allowing tampering and injection of bad code.

RQ5: Are TLS validation and TLS certificate pinning used?

See Figure 12. Besides two applications, all 49 applications have fully enforced a baseline of TLS validation. One game did not have TLS validation and allowed data from spoofed servers. The other app, *Coyote*, has TLS validation on the API calls but is softening security when loading map data. Those apps can easily be victims of MITM attacks.

It was also found that eight applications have a more strict TLS validation by enforcing some level of TLS certificate pinning. This was the case for social applications and *MyBpost*,

which handles some payment and personal data. One application, *GovApp*, did not offer TLS certificate pinning, even though this security level was expected, as the main objective of GovApp is to be a trusted platform for all notifications of the Belgian government.

RQ6: To what extent are differences observable between the monetisation models regarding user privacy?

Section 8.6 researches if there is any notable difference based on the monetisation model related to tracking services, usage of encrypted connections and level of TLS encryption. The total count of applications in scope is 50 . There are three monetisation models: free, paid and grossing, with 23, 11 and 18 apps in our dataset, visualised by Figure 8. There are only around 11 and 23 applications in each monetisation type, which makes the data scope relatively small to make statements with high accuracy.

However, taking into account the size of the dataset, it was found that:

- Number of apps with unencrypted traffic is relatively almost equal for each monetisation model.
- Five applications exchange data over unencrypted connections where privacy or security is at risk:
  - The upload of personal photos was a free application.
  - Three grossing apps (HLN, State Of Survival and Deezer) are loading metadata or executing JavaScript over insecure connections.
  - One paid game, Plague Inc, sends metadata over insecure connections.
- Grossing apps and paid apps have most trackers implemented. See Figure 16.
- All apps have a baseline of TLS enforced, besides two grossing apps.
- Eight applications have TLS certificate pinning implemented. Mostly free applications and one grossing application.

In conclusion, we can state that there is no notable difference between the monetisation models for the applications in scope, besides that grossing applications tend to use more trackers than free and paid applications.

Main question: To what extent do iOS applications, that are popular in Belgium, respect the privacy of the users?

It is essential to state that one-fourth of the selected applications have some network requests over unencrypted connections, which is unexpected and concerning. 10% of all requests were unencrypted. At the same time, most of the contacted servers support secure connections, making it hard to understand why those apps still use the unencrypted method.

Five applications, or 10% of the applications in scope, have a potential security and privacy risk with this setup, as they execute data from insecure locations or exchange possible personal data like selfies or metadata unencrypted.

Regarding tracking, almost all tested applications use trackers. On average, five trackers were detected per application. There is only one application where we could not find any tracker. This is also the same application where it was detected that not all traffic was properly intercepted.

No consent was given to be tracked, but many third-party trackers were active and personal data was sent to those trackers. Some trackers are even sending the name of the city where this research was performed, including the carrier name of the SIM card of the test iPhone, as well as the name of the ISP of the fixed internet connection. Even paid apps do have ad trackers embedded.

One application is vulnerable to MITM attacks, and one is partly vulnerable. Luckily all other applications do have a baseline of TLS validations enforced, and eight apps do have some level of TLS certificate pinning enforced. One application did not use TLS certificate pinning, which was expected as the app's focus is to centralise government notifications and communication.

In conclusion: We can conclude that a baseline of security and privacy is present. However, even in this small subset of 50 applications, we found multiple improvement points and concerns regarding security and privacy that look easy to resolve. Knowing that Apple offers a baseline of app usage and crash reporting with a proper consent system[54], it is concerning that, so commonly, an app is still using different tracking services for additional insights. The ATT consent is also not fully respected, as the loaded trackers send personal data or data that allows identification of the user by fingerprinting techniques. Especially with ATS in place, I expect all data to be encrypted. Especially when uploading personal data like selfies or executing JavaScript files.

Responsible disclosure: The main objective of this research is to verify to which extent the users' privacy is respected. Findings of this report, like the missing TLS certificate pinning and security issues, were reported to the developers of those applications to get those issues addressed and resolved. As of March 20, 2023, no replies were received besides one reply that the email was forwarded to the security team.

10. Future Work

This research was performed by manually analysing the traffic of 50 applications, which is a rather small data set. Future work could be to research how data collection and automatic testing could be automated. This is more complex on iOS due to the sandbox and restrictions. Possibilities could be using the accessibility frameworks such as VoiceOver to 'remotely' control the application by clicking buttons and analysing the current views. Another possibility could be using AssistiveTouch[55]. It is also possible to connect a mouse to AssistiveTouch, via USB or BlueTooth, which could be spoofed software to control the device in an automated way. Once automated, the research could be repeated with a bigger data set. This will also allow us to repeat the test and track any evolutions and changes. This could also be valuable to the research of Kollnig et al. [2022a], where iOS applications were dynamically tested by opening the applications and leaving them open for 30 seconds without any interactions. Kollign et all. have a public project to find iOS applications and download them, but not yet for running fully all applications flows dynamically.

Section 6.3 describes how the collected data was analysed and processed. Custom scripts were developed to process the output of the proxy server. This allowed us to check if any unencrypted traffic was detected quickly, how many requests were intercepted and more. The

[54] https://developer.apple.com/app-store-connect/analytics/
[55] https://support.apple.com/en-gb/HT210546

scripts also detect which tracking services are used. Future work could be to extend those scripts by extending the list of trackers and processing multiple applications at the same time. The scripts could summarise the detected trackers per application and applications per tracker. Another useful addition would be to verify the proxy data directly with the DNS logs to detect if the proxy data set missed any data or trackers.

This research collected data via a proxy setup and a custom DNS server to log all queries. As stated in Chapter 8, it was detected that not all traffic was detected, for instance, XMPP traffic was unnoticed. One alternative research method could be to use a VPN to reroute all traffic and intercept the data on the VPN server side and compare it with the findings of our research. If the device can be jailbroken, data can be intercepted by logging all packets of the network interfaces. Other options are intercepting data at the network gateway or rerouting traffic by hijacking DNS requests.

Section 7.3 describes the algorithm used to define popular applications. This algorithm can be further improved by using more metadata such as reviews, ratings and more.

One of the findings of our research is that grossing applications have the most trackers. Most grossing applications are initially free and have in-app payments to remove ads or enable extra features. It could be interesting to verify if there is a notable difference in the tracking if this application is used for free, like our research, and when the application is upgraded by purchasing the in-app payment.

In late 2020, Apple added privacy labels and App Tracking Transparency in the App Store. It would be valuable to verify if the claimed privacy labels in the App Store match the actual tracking behaviour of the applications and check for any misleading information. Especially as a recent study[56], February 2023, states that similar labels of the iOS privacy labels in the Android Google Play Store are misleading and incorrect. Comparing this with the iOS App Store in more detail would be useful. See also the new research of Kollnig et al. [2022b], but applied to data gathered via dynamic testing.

Lastly, this research focused on the encryption level of exchanged data, and which trackers we can detect of popular applications by third-party developers. It could also be interesting to analyse in more detail the default tracked data and events by iOS, and trackers in the applications owned by Apple. Especially after a report of Mysk Co at November 2022[57] that shows that every action and click in the App Store on iOS is triggering a call to an Apple server with tracking and user information.

[56] https://foundation.mozilla.org/en/privacynotincluded/articles/mozilla-study-data-privacy-labels-for-most-top-apps-in-google-play-store-are-false-or-misleading/

[57] https://www.macworld.com/article/1373780/app-store-activity-data-class-action-lawsuit.html

A. Appendix: Top lists

Table 2 shows the top free apps, in the Belgian App Store of Apple over the period of May, June and July 2022. Table 3 shows the top list for paid apps and Table 4 free apps, which are top-grossing via in-app purchases. The list is filtered on iPhone apps.

Used commands to generate the tables:

```
ruby toplist.rb free iphone 30 Example_May-July_2022.sqlite
ruby toplist.rb paid iphone 30 Example_May-July_2022.sqlite
ruby toplist.rb grossing iphone 30 Example_May-July_2022.sqlite
```

	Name	min pos	avg pos	max pos	# days	score
1	Minecraft	1	1.921	12	89	0.022
2	Oei, ik groei!	1	3.843	13	89	0.043
3	Babyfoon 3G	1	7.136	38	88	0.081
4	Poppy Playtime Chapter 1	1	7.264	38	87	0.083
5	WeatherPro	1	10.169	43	89	0.114
6	Forest - Your Focus Motivation	1	11.129	46	85	0.131
7	Rovio Classics: AB	1	11.977	37	87	0.138
8	Monopoly - Classic Board Game	3	12.455	34	88	0.142
9	Geometry Dash	3	12.605	43	86	0.147
10	Plague Inc.	3	13.523	39	88	0.154
11	Football Manager 2022 Mobile	2	13.697	44	89	0.154
12	Procreate Pocket	4	14.348	36	89	0.161
13	AutoSleep slaaptracker	7	14.750	37	88	0.168
14	Earn to Die 2	1	15.254	47	71	0.215
15	Grand Theft Auto: San Andreas	6	20.135	50	89	0.226
16	Threema. De veilige messenger	3	17.382	49	76	0.229
17	60 Seconds! Atomic Adventure	1	15.373	47	67	0.229
18	Bloons TD 6	6	19.425	47	80	0.243
19	True Skate	10	19.519	46	79	0.247
20	TouchRetouch	8	22.500	48	74	0.304
21	Monash University FODMAP diet	4	23.013	48	75	0.307
22	FILCA - SLR Film Camera	2	18.585	43	53	0.351
23	Pou	9	28.462	50	65	0.438
24	FiLMiC Pro Video Camera	8	28.450	47	60	0.474
25	Stardew Valley	4	27.407	50	54	0.508
26	TjilpOMatic - Vogelgezang ID	6	27.889	49	54	0.516
27	Pocket Build	4	22.744	50	43	0.529
28	TripRoad Pro	2	25.070	50	43	0.583
29	Street Kart Racing Game - GT	4	24.390	49	41	0.595
30	Braintoss	9	30.490	50	49	0.622

Table 3: Top paid apps in Belgium for iPhone, in May, June & July 2022

	Name	min pos	avg pos	max pos	# days	score
1	TikTok: Video's & Muziek	1	1.494	3	89	0.017
2	Tinder	1	1.573	3	89	0.018
3	Disney+	3	3.562	6	89	0.040
4	YouTube	3	4.191	6	89	0.047
5	Candy Crush Saga	4	5.865	10	89	0.066
6	LinkedIn: Een baan zoeken	5	7.798	14	89	0.088
7	Pokémon GO	1	8.079	20	89	0.091
8	Coin Master	5	10.303	26	89	0.116
9	Strava: hardlopen en fietsen	3	10.697	28	89	0.120
10	Netflix	8	11.888	29	89	0.134
11	Coyote: gps-navigatie & radars	6	14.247	26	89	0.160
12	Clash of Clans	3	14.719	33	89	0.165
13	PUBG MOBILE	3	16.663	32	89	0.187
14	YouTube Music	12	17.135	25	89	0.193
15	Homescapes	11	17.607	25	89	0.198
16	Duolingo	9	17.798	32	89	0.200
17	Gardenscapes	10	18.472	30	89	0.208
18	Clash Royale	3	18.716	47	88	0.213
19	ROBLOX	9	19.652	40	89	0.221
20	Royal Match	10	19.685	30	89	0.221
21	State of Survival: Zombie War	7	19.843	37	89	0.223
22	HLN	12	20.157	30	89	0.226
23	Rise of Kingdoms	6	22.270	40	89	0.250
24	Deezer - Muziek en podcasts	16	24.663	34	89	0.277
25	Fishdom	14	24.843	34	89	0.279
26	Bumble - Date & Vrienden & Chat	17	25.247	36	89	0.284
27	Evony - The King's Return	10	28.976	47	85	0.341
28	Dropbox: Cloudopslag en Sync	22	30.730	42	89	0.345
29	Empires & Puzzles: Match-3 RPG	8	30.800	50	85	0.362
30	Badoo - Dating. Chats. Friends	25	33.764	44	89	0.379

Table 4: Top grossing apps in Belgium for iPhone, in May, June & July 2022

B. Appendix: Selected applications

Table 5 shows the applications selected for this research.

	App	App Category	Monetisation Category
1	GovApp	News	Free
2	Google Maps	Navigation	Free
3	Waze	Navigation	Free
4	NMBS	Travel	Free
5	TikTok	Entertainment	Free & Grossing
6	BeReal	Social Networking	Free
7	WeTransfer	Productivity	Free
8	Vinted	Shopping	Free
9	Booking.com	Travel	Free

10	Stumble Guys	Games	Free
11	WeatherPro	Weather	Paid
12	WeatherPro Lite	Weather	Free
13	GTA San Andreas	Games	Paid
14	Pocket Build	Games	Paid
15	60 Seconds!	Games	Paid
16	HLN	News	Grossing
17	Empires & Puzzles	Games	Grossing
18	Duolingo	Education	Grossing
19	Coin Master	Games	Grossing
20	Candy Crush Saga	Games	Grossing
21	Pokemon Go	Games	Grossing
22	Coyote: gps-navigatie & radars	Navigation	Grossing
23	Royal Match	Games	Grossing
24	Plague Inc.	Games	Paid
25	Homescapes	Games	Grossing
26	Geometry Dash Lite	Games	Free
27	Geometry Dash	Games	Paid
28	Braintoss	Productivity	Paid
29	Forest: Focus for Productivity	Productivity	Paid
30	YouTube	Photo & Video	Free & Grossing
31	State of Survival: Zombie War	Games	Grossing
32	Too good to go	Food & Drink	Free
33	Subway Surfers	Games	Free
34	Snapchat	Photo & Video	Free
35	NewProfilePic Picture Editor	Photo & Video	Free
36	SHEIN - Online mode	Shopping	Free
37	Whatsapp Messenger	Social Networking	Free
38	My bpost	Utilities	Free
39	Clash of Clans	Games	Grossing
40	Earn To Die 2	Games	Paid
41	Earn To Die 2 Lite	Games	Free
42	Fishdom	Games	Grossing
43	Payconiq by Bancontact	Finance	Free
44	Street Kart Racing Game - GT	Games	Paid
45	YouTube Music	Music	Grossing
46	Deezer: Music & Podcast player	Music	Grossing
47	Poppy Playtime Chapter 1	Games	Paid
48	Evony - The King's return	Games	Grossing
49	Rise of Kingdoms	Games	Grossing
50	Spotify	Music	Free

Table 5: Selected apps for research. Combination of free apps, paid apps and grossing apps

C. Appendix: App Annie example data set

The following tables shows the data returned by app store trackers.

Name	Example value
id	281855823121
name	ING Banking
developer	ING Banking
developer_id	488551
storefront	apple:ios
vendor_identifier	1364967778
price	0.00 EUR

Table 6: Example app info provided by AppFigures

Name	Example value
sort_metric	changeInRank
name	ING Banking
company_url	/company/1000200000036315/
headerquarters_id	Poland
id	1364967778
company_name	ING Bank
country_code	pl
app_icon_css	ios
iap	false
change	0
icon	https://static-s.aa-cdn.net/img/ios/1364967778/b1d5890ca594f2e51a75f0688d4167f6_w80

Table 7: Data structures of external App Store trackers

Name	Example value
app_id	1364967778
canonical_country	BE
name	ING Banking
humanized_name	ING Banking
icon_url	https://is1-ssl.mzstatic.com/...150x150bb.png
os	ios
id	1364967778
appId	1364967778
publisher_name	ING BELGIUM
publisher_id	416838657
icon	https://is1-ssl.mzstatic.com/image/.../150x150bb.png
iconUrl	https://is1-ssl.mzstatic.com/image/...150x150bb.png
url	https://apps.apple.com/BE/app/id1364967778?l=nl
categories	[6015, 6000]
valid_countries	[US,AU,CA,...]
app_view_url	/ios/be/ing-belgium/app/ing-banking/1364967778/
publisher_profile_url	/ios/publisher/ing-belgium/416838657
release_date	2019-12-09T08:00:00Z
updated_date	2020-07-22T00:00:00Z
in_app_purchases	
shows_ads	null
buys_ads	null
rating	1.92577
price	0.0
global_rating_count	542
rating_count	485
rating_count_for_current_version	485
rating_for_current_version	1.92577
version	1.14
apple_watch_enabled	null
apple_watch_icon	null
imessage_enabled	null
imessage_icon	null
humanized_worldwide_last_month_downloads	{"downloads": 30000, ...}
humanized_worldwide_last_month_revenue	{"prefix": "< $", "revenue": 1000...}
bundle_id	be.ING.OneApp
support_url	http://www.ing.be/iosphoneapp
website_url	null
privacy_policy_url	https://www.ing.be/.../PrivacyStatementNL.pdf
eula_url	null
feature_graphic	null
content_rating	
rank	1
delta	0
downloads_revenue_date	2020-07-23T00:00:00Z

Table 8: Example app info provided by SensorTower

Name	Example value
country	be
rating_cnt	486
pos	1
icon	https://is1-ssl.mzstatic.com/image/.../53x53bb.png
url	https://apps.apple.com/be/app/ing-banking/id1364967778
genre_id	0
ext_id	1364967778
pos_curr	1
device	iphone
artist_name	ING BELGIUM
feed_type	free
pos_diff	0
title	ING Banking
rating_avg	1.9

Table 9: Example app info provided by AppFollow

D. Appendix: Application metadata by top list generator

The following output is generated by the *top list generator* script. In this case, a dump of the NMBS app is printed:

```
name: NMBS : Trein info & tickets
times_seen_in_database: 89
min_rating: 2
avg_rating: 6.370786516853933
max_rating: 24
privacy_g_url: https://www.belgiantrain.be/en/privacy
url_g: https://apps.apple.com/BE/app/id1504870215?l=nl
support_url_g: https://www.belgiantrain.be/nl/support/forms
website_url_g: https://www.belgiantrain.be
eula_url_g: https://itunes.apple.com/{...}/wa/viewEula?id=1504870215
date: 2022-06-05
app_id: 1504870215
category: free
device: iphone
rank: 24
version: 3.5.3
rating: 1.79929
rating_for_current_version: 1.79929
rating_count: 1519
rating_count_for_current_version: 1410
canonical_country: BE
categories: [6003, 6010]
url: https://apps.apple.com/BE/app/id1504870215?l=nl
support_url: https://www.belgiantrain.be/nl/support/forms
website_url: https://www.belgiantrain.be
privacy_policy_url: https://www.belgiantrain.be/en/privacy
eula_url: https://itunes.apple.com/{...}/wa/viewEula?id=1504870215
release_date: 2020-09-08T07:00:00Z
updated_date: 2022-07-07T00:00:00Z
humanized_worldwide_last_month_downloads: 70000
humanized_worldwide_last_month_revenue: 1000
price: 0.0
in_app_purchases: 1
shows_ads: 0
buys_ads: 0
apple_watch_enabled: 0
imessage_enabled: 0
icon_url: https://is1-ssl.mzstatic.com/image/{...}24bb.png
```

E. Appendix: Results

E.1. HTTP versus HTTPS

App	Category	Total	Encrypted	Plain Text	Excl. OCSP	%
GovApp	free	84	82	2	0	0.00%
Google Maps	free	850	849	1	0	0.00%
Waze	free	1447	1447	0	0	0.00%
NMBS	free	429	429	0	0	0.00%
TikTok	free&grossing	400	398	2	0	0.00%
BeReal	free	236	229	7	0	0.00%
WeTransfer	free	349	348	1	0	0.00%
Vinted	free	1109	1109	0	0	0.00%
Booking.com	free	708	708	0	0	0.00%
Stumble Guys	free	424	420	4	**4**	**0,94%**
WeatherPro Lite	free	467	467	0	0	0.00%
WeatherPro	paid	292	292	0	0	0.00%
GTA San Andreas	paid	13	9	4	**4**	**30,77%**
Pocket Build	paid	70	70	0	0	0.00%
60 Seconds!	paid	36	36	0	0	0.00%
HLN	grossing	1396	1392	4	**4**	**0,29%**
Empires & Puzzles	grossing	195	195	0	0	0.00%
Duolingo	grossing	4432	4432	0	0	0.00%
Coin Master	grossing	3536	3429	7	0	0.00%
Candy Crush Saga	grossing	61	61	0	0	0.00%
Pokemon Go	grossing	319	305	14	0	0.00%
Coyote	grossing	196	196	0	0	0.00%
Royal Match	grossing	103	103	0	0	0.00%
Plague Inc.	paid	27	10	17	**17**	**62,96%**
Homescapes	grossing	343	343	0	0	0.00%
Geometry Dash Lite	free	757	729	28	**28**	**3,70%**
Braintoss	paid	48	47	1	0	0.00%
Forest: Focus Productivity	paid	132	132	0	0	0.00%
YouTube	free&grossing	841	839	2	0	0.00%
State of Survival:ZombieWar	grossing	2570	289	2281	**2281**	**88,75%**
Too good to go	free	274	274	0	0	0.00%
Subway Surfers	free	440	440	0	0	0.00%
Snapchat	free	254	254	0	0	0.00%
NewProfilePic Picture Editor	free	171	161	10	**10**	**5,85%**
SHEIN - Online mode	free	900	900	0	0	0.00%
Whatsapp Messenger	free	24	24	0	0	0.00%
My bpost	free	55	52	3	0	0.00%
Clash of Clans	grossing	1302	1302	0	0	0.00%
Earn To Die 2	paid	37	31	6	**6**	**16,22%**
Earn To Die 2 Lite	free	6	6	1	**1**	**16,67%**
Fishdom	grossing	316	316	0	0	0.00%
Payconiq by Bancontact	free	79	70	9	**6**	**7,59%**

Street Kart Racing Game	paid	152	149	3	**3**	**1,97%**
YouTube Music	grossing	421	420	1	0	0.00%
Deezer	grossing	732	500	232	**232**	**31,69%**
Poppy Playtime Chapter 1	paid	28	28	0	0	0.00%
Evony - The King's return	grossing	198	198	0	0	0.00%
Rise of Kingdoms	grossing	561	269	292	**292**	**52,05%**
Spotify	free	1433	1431	2	0	0.00%
Total		29253	26220	2934	**2888**	**9,87%**

Table 10: Requests filtered on encryption

E.2. Certificate pinning

App	Category	Pinning
GovApp	free	No
Google Maps	free	No
Waze	free	No
NMBS	free	No
TikTok	free & grossing	**Yes**
BeReal	free	**Yes**
WeTransfer	free	No
Vinted	free	No
Booking.com	free	No
Stumble Guys	free	No
WeatherPro	paid	No
WeatherPro Lite	free	No
GTA San Andreas	paid	No
Pocket Build	paid	No
60 Seconds!	paid	No
HLN	grossing	No
Empires & Puzzles	grossing	No
Duolingo	grossing	No
Coin Master	grossing	No
Candy Crush Saga	grossing	No
Pokemon Go	grossing	**Yes**
Coyote: GPS-navigatie & radars	grossing	No
Royal Match	grossing	No
Plague Inc.	paid	No
Homescapes	grossing	No
Geometry Dash Lite	free	No
Geometry Dash	paid	No
Braintoss	paid	No
Forest: Focus for Productivity	paid	No
YouTube	free & grossing	No
State of Survival: Zombie War	grossing	No
Too good to go	free	No
Subway Surfers	free	No

Snapchat	free	**Yes**
NewProfilePic Picture Editor	free	No
SHEIN - Online mode	free	No
Whatsapp Messenger	free	Yes
My bpost	free	Yes
Clash of Clans	grossing	No
Earn To Die 2	paid	No
Earn To Die 2 Lite	free	No
Fishdom	grossing	No
Payconiq by Bancontact	free	**Yes**
Street Kart Racing Game - GT	paid	No
YouTube Music	grossing	No
Deezer: Music & Podcast player	grossing	No
Poppy Playtime Chapter 1	paid	No
Evony - The King's return	grossing	No
Rise of Kingdoms	grossing	No
Spotify	free	**Partial**

Table 11: Overview of TLS Certificate pinning

E.3. TLS certificate validations

App	Category	TLS Certificate Validation
GovApp	free	Yes
Google Maps	free	Yes
Waze	free	Yes
NMBS	free	Yes
TikTok	free & grossing	Yes
BeReal	free	Yes
WeTransfer	free	Yes
Vinted	free	Yes
Booking.com	free	Yes
Stumble Guys	free	Yes
WeatherPro	paid	Yes
WeatherPro Lite	free	Yes
GTA San Andreas	paid	Yes
Pocket Build	paid	Yes
60 Seconds!	paid	Yes
HLN	grossing	Yes
Empires & Puzzles	grossing	Yes
Duolingo	grossing	Yes
Coin Master	grossing	Yes
Candy Crush Saga	grossing	Yes
Pokemon Go	grossing	Yes
Coyote: GPS-navigatie & radars	grossing	**Partial**
Royal Match	grossing	Yes
Plague Inc.	paid	Yes

Homescapes	grossing	Yes
Geometry Dash Lite	free	N/A
Geometry Dash	paid	N/A
Braintoss	paid	N/A
Forest: Focus for Productivity	paid	Yes
YouTube	free & grossing	Yes
State of Survival: Zombie War	grossing	Yes
Too good to go	free	Yes
Subway Surfers	free	Yes
Snapchat	free	Yes
NewProfilePic Picture Editor	free	Yes
SHEIN - Online mode	free	Yes
Whatsapp Messenger	free	Yes
My bpost	free	Yes
Clash of Clans	grossing	Yes
Earn To Die 2	paid	Yes
Earn To Die 2 Lite	free	Yes
Fishdom	grossing	Yes
Payconiq by Bancontact	free	Yes
Street Kart Racing Game - GT	paid	Yes
YouTube Music	grossing	Yes
Deezer: Music & Podcast player	grossing	Yes
Poppy Playtime Chapter 1	paid	N/A
Evony - The King's return	grossing	Yes
Rise of Kingdoms	grossing	**No**
Spotify	free	Yes

Table 12: Overview of TLS certificate validations

E.4. Tracker Services per application

Ad Trackers

App	Type	Facebook	Double Click	Quant	OutBrain	AddAppTr	PubNative	Amazon	Vungle	UnityAds	SupersonicAds	Criteo	Tead	AdColony
GovApp	free													
Google Maps	free		X											
Waze	free	X	X											
NMBS	free	X	X		X								X	
TikTok	free	X												
BeReal	free													
WeTransfer	free		X											
Vinted	free	X	X			X	X	X						
Booking.com	free	X												
Stumble Guys	free	X	X							X				
WeatherPro Lite	free		X						X			X		X
GeometryDash Lite	free	X	X						X	X	X			
YouTube	free		X											
Too good to go	free													
Subway Surfers	free	X	X						X	X	X			X
Snapchat	free													
NewProfilePic	free										X			
SHEIN	free	X												
Whatsapp	free													
My bpost	free													
EarnToDie2 Lite	free													
Payconiq	free													
Spotify	free													
WeatherPro	paid		X	X				X				X		
GTA	paid													
Pocket Build	paid													
60 Seconds!	paid													
Plague Inc.	paid	X												
Braintoss	paid													
Forest: Focus	paid		X											
Earn To Die 2	paid													
Street Kart	paid	X	X											
Poppy Playtime	paid	X												
TikTok	grossing	X												
HLN	grossing		X		X							X	X	
Empires& Puzz.	grossing	X	X							X				
Duolingo	grossing	X	X							X				
Coin Master	grossing	X												

Candy Crush	grossing	X												
Pokemon Go	grossing	X	X											
Coyote	grossing													
Royal Match	grossing	X								X	X			
Homescapes	grossing	X							X	X	X			
YouTube	grossing		X											
State Survival	grossing	X												
Clash Clans	grossing													
Fishdom	grossing	X							X	X	X			
YouTubeMusic	grossing		X											
Deezer	grossing	X	X											
King return	grossing	X								X				
Rise Kingdoms	grossing	X												
Sum free		9	10	0	1	1	1	1	3	3	3	1	1	2
Sum paid		3	3	1	0	0	0	1	0	0	0	1	0	0
Sum grossing		13	7	0	1	0	0	0	2	6	3	1	1	0
Total sum		25	20	1	2	1	1	2	5	9	6	3	2	2

Table 13: Overview of Ads trackers per application

App Usage Trackers

App	Type	app-measurement	Google Analytics	AppsFlyer	Flurry Tracking	Facebook	Hotjar	Amplitude	AppLovin	ironSource
GovApp	free									
Google Maps	free		X							
Waze	free	X	X			X				
NMBS	free	X	X			X	X			
TikTok	free	X		X		X				
BeReal	free	X								
WeTransfer	free	X	X					X		
Vinted	free	X				X				
Booking.com	free					X				
Stumble Guys	free	X				X			X	
WeatherPro Lite	free	X							X	
GeometryDash Lite	free	X				X			X	
YouTube	free	X								
Too good to go	free	X		X						
Subway Surfers	free	X				X			X	
Snapchat	free									
NewProfilePic	free	X								X
SHEIN	free	X	X	X		X				

Whatsapp	free									
My bpost	free	X								
EarnToDie2 Lite	free				X					
Payconiq	free	X								
Spotify	free									
WeatherPro	paid	X	X							
GTA	paid		X							
Pocket Build	paid			X						
60 Seconds!	paid									
Plague Inc.	paid					X				
Braintoss	paid	X								
Forest: Focus	paid	X	X	X						
Earn To Die 2	paid				X					
Street Kart	paid	X		X		X				
Poppy Playtime	paid					X				
TikTok	grossing	X				X				
HLN	grossing	X	X							
Empires& Puzz.	grossing				X	X				
Duolingo	grossing	X				X				
Coin Master	grossing	X		X		X				
Candy Crush	grossing					X				
Pokemon Go	grossing			X		X				
Coyote	grossing	X	X	X						
Royal Match	grossing			X		X				
Homescapes	grossing	X		X		X				
YouTube	grossing	X								
State Survival	grossing	X		X		X				
Clash Clans	grossing		X							
Fishdom	grossing	X		X		X				
YouTubeMusic	grossing	X								
Deezer	grossing	X	X			X				
King return	grossing	X				X				
Rise Kingdoms	grossing	X			X	X				
Sum free		16	5	3	1	9	1	1	4	1
Sum paid		4	3	3	1	3	0	0	0	0
Sum grossing		13	4	7	2	13	0	0	0	0
Total sum		33	12	13	4	25	1	1	4	1

Table 14: Overview of App Usage trackers per application

App Performance & Bugs Trackers

App	Type	CrashLytics	Sentry	DataDog	AppCenter	BugSnag	NewRelic
GovApp	free	X					
Google Maps	free						
Waze	free	X					
NMBS	free						
TikTok	free						
BeReal	free	X		X			
WeTransfer	free			X			
Vinted	free	X					
Booking.com	free	X					
Stumble Guys	free						
WeatherPro Lite	free	X					
GeometryDash Lite	free						
YouTube	free						
Too good to go	free	X	X				
Subway Surfers	free	X					
Snapchat	free		X				
NewProfilePic	free						
SHEIN	free	X					
Whatsapp	free						
My bpost	free	X					
EarnToDie2 Lite	free						
Payconiq	free	X					
Spotify	free	X					
WeatherPro	paid	X		X			
GTA	paid						
Pocket Build	paid						
60 Seconds!	paid						
Plague Inc.	paid						
Braintoss	paid	X					
Forest: Focus	paid						
Earn To Die 2	paid						
Street Kart	paid	X					
Poppy Playtime	paid			X			
TikTok	grossing						
HLN	grossing	X					
Empires& Puzz.	grossing					X	
Duolingo	grossing	X					
Coin Master	grossing	X					
Candy Crush	grossing						
Pokemon Go	grossing	X					X

Coyote	grossing	X					
Royal Match	grossing	X					
Homescapes	grossing		X		X		
YouTube	grossing						
State Survival	grossing	X					
Clash Clans	grossing		X				
Fishdom	grossing		X		X		
YouTubeMusic	grossing						
Deezer	grossing	X	X				X
King return	grossing	X					
Rise Kingdoms	grossing						
Sum free		12	2	2	0	0	0
Sum paid		3	0	2	0	0	0
Sum grossing		9	4	0	2	1	2
Total sum		24	6	4	2	1	2

Table 15: Overview of app performance and bug trackers per application

Games

App	Type	Unity3D Tracking	DeltaDNA	GameAnalytics	LuneLabs
GovApp	free				
Google Maps	free				
Waze	free				
NMBS	free				
TikTok	free				
BeReal	free				
WeTransfer	free				
Vinted	free				
Booking.com	free				
Stumble Guys	free	X		X	
WeatherPro Lite	free				
GeometryDash Lite	free	X			
YouTube	free				
Too good to go	free				
Subway Surfers	free				X
Snapchat	free				
NewProfilePic	free				
SHEIN	free				
Whatsapp	free				
My bpost	free				
EarnToDie2 Lite	free				

Payconiq	free				
Spotify	free				
WeatherPro	paid				
GTA	paid				
Pocket Build	paid	X			
60 Seconds!	paid	X			
Plague Inc.	paid				
Braintoss	paid				
Forest: Focus	paid				
Earn To Die 2	paid				
Street Kart	paid	X	X		
Poppy Playtime	paid				
TikTok	grossing				
HLN	grossing				
Empires& Puzz.	grossing	X			
Duolingo	grossing	X			
Coin Master	grossing				
Candy Crush	grossing				
Pokemon Go	grossing				
Coyote	grossing				
Royal Match	grossing	X			
Homescapes	grossing				
YouTube	grossing				
State Survival	grossing		X		
Clash Clans	grossing				
Fishdom	grossing	X			
YouTubeMusic	grossing				
Deezer	grossing				
King return	grossing	X			
Rise Kingdoms	grossing		X		
Sum free		2	0	1	1
Sum paid		3	1	0	0
Sum grossing		5	2	0	0
Total sum		10	3	1	1

Table 16: Overview of game trackers per application

Overview of trackers

Tracker	Type	Number of apps	%
app-measurement	App Usage	33	64,71%
Facebook	Ads & App Usage	25	49,02%
CrashLytics	App Performance/Bugs	24	47,06%
Double Click	Ads	20	39,22%
AppsFlyer	App Usage	13	25,49%
Google Analytics	App Usage	12	23,53%
Unity3D Tracking	Games	10	19,61%
UnityAds	Ads	9	17,65%
SupersonicAds	Ads	6	11,76%
Sentry	App Performance/Bugs	6	11,76%
Vungle	Ads	5	9,80%
DataDog	App Performance/Bugs	4	7,84%
Flurry Tracking	App Usage	4	7,84%
AppLovin	App Usage	4	7,84%
Criteo	Ads	3	5,88%
DeltaDNA	Games	3	5,88%
OutBrain	Ads	2	3,92%
Amazon	Ads	2	3,92%
Tead	Ads	2	3,92%
AdColony	Ads	2	3,92%
AppCenter	App Performance/Bugs	2	3,92%
NewRelic	App Performance/Bugs	2	3,92%
Quant	Ads	1	1,96%
AddAppTr	Ads	1	1,96%
PubNative	Ads	1	1,96%
BugSnag	App Performance/Bugs	1	1,96%
GameAnalytics	Games	1	1,96%
LuneLabs	Games	1	1,96%
Hotjar	App Usage	1	1,96%
Amplitude	App Usage	1	1,96%
ironSource	App Usage	1	1,96%

Table 17: Applications per tracker

Overview of trackers per monetisation model

	Free Apps			Paid Apps			Grossing Apps		
Type	Apps	Total	%	Apps	Total	%	Apps	Total	%
Ads	14	23	60,87%	5	10	50,00%	16	18	88,89%
App Usage	19	23	82,61%	9	10	90,00%	18	18	100,00%
App Performance Bugs	14	23	60,87%	5	10	50,00%	13	18	72,22%
Games	3	23	13,04%	3	10	30,00%	7	18	38,89%
Average	12,5	23	54,35%	5,5	10	55,00%	13,5	18	75,00%

Table 18: Applications with a minimum of one tracker per monetisation model

Overview of data collected by SuperSonicAds

xCodeVersion	1320
bdo	4
ismsd	1
fnv	3f5effc2...8652
supplyTrackID	512...461
city	**Wilrijk**
iabbp	baad5f...c033fea072
mobileCarrier	**Orange B**
dynamicParameterFlags	2815...528
iabai	9fb24710-...a98d8_1245070955
rati	2
sessionDepth	1
userApp	adca378c-...1e5_318627
sdt	b
dsOppPlumbus	0CgUIOBIBMgoHCA...MzIKBQgeEgEw
adType	2
auff	32
deviceGroup	16
cld	a
deviceModel	**iPhone8,1**
demandBundleEntityID	450891
bds	32a7fb928...1c268455f0a
dpi	12
ppi	2.0
isp	**Proximus**
ifer	e1e0cab...d6ea60a9
ispl	true
supplyBundleId	com.kiloo.subwaysurfers
deviceLanguage	en
rt	16683...7579
iabbt	hard
instanceID	2562945
country	**BE**
rewardsHex	1

userAgent	Mozilla5.0 (iPhone; CPU **iPhone OS 15_6** like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile15E148
cr	KLUv_SA...-kSmS0JoLY
simp	On
browserName	Mobile Safari UI/WKWebView
ipData	0
uPlumbus	0CgkIBRIFNDk0...MDAwMA
displayWidth	187.5
auid	2e8bfb8d-...-b824f2d8a83e
deviceOs	ios
dsicp	0.0173...15873
appOrientation	Portrait
deviceOEM	Apple
dsReqPlumbus	0CgsIN...Dc3Mg
insType	2
advertiserID	99933
displayHeight	333.5
diskFreeSize	49430
deliveryType	8
creativeID	71...66
suppressStatistics	false
deviceOSVersionID	20525460
segments	segments
att	2
caf	0
ifb	32a7fb928d8...68455f0a
demandBundleID	com.fluffyfairygames.idleminertycoon
idfv	DA06BECE-...-2FC39E2B354A
encodedAppUserId	adca378c-9a01-...-80a374ec71e5
vuid	DA06BECE-...-2FC39E2B354A
rip	h8IOwOBc...bhy5ZA
ifprb	0.0018
te	false
decisionStrategyID	1120
sdp	0
scid	35
cf	true
bannerID	8535509
gsID	258233
petel	2
skVer	3.0
campaignID	8495565
udbf	0
ubc	4PivuIUya_LEUL3...FaHb8IeYhw
deviceOSVersion	15.6
cp	false

platform	2
urc	YQjdMSL...ignqPhuw
accLang	en-GB, en;q=0.9
endcardType	0
isLimitAdTrackingEnabled	true
sdkAbName	0
UII	9c711c1a-8b8c-...-6dd65f2e326a
landingID	12275
SDKVersion	5.104
appID	318627
connectionType	2
mdc	false
etype	xButton
esignature	0288a8a0-...-3123bb83189a
sp	1 277...-2-0

Table 19: Data collected and sent by SuperSonicAds

F. Appendix: Reflection

I finished the *software engineer graduation assignment preparation* in April 2020, three years ago. Due to personal reasons, I had to pause the graduation assignment multiple times, which was not optimal as the technology is evolving quickly, especially regarding security and privacy. Some initial ideas were useless due to new features and guards added to iOS. For example, in iOS 14, Apple announced that a warning would be shown when an application reads the device's clipboard for awareness. Since iOS 16, you need to give explicit consent before an application can read your clipboard. In my original plan, I wanted to store a unique value in the clipboard to verify if any application was reading and sending this data to a server. Mid 2020, shortly after I completed my graduation assignment preparation, a report was published that multiple applications were reading the clipboard and potentially sharing this data[58].

Related to the timing, other studies were published while I was researching. For example, the research of Kollnig et al. [2022a] where a set of tools is made public to download iOS applications and to analyse the binaries for detecting trackers. Luckily the dynamic testing, by intercepting traffic, was done by opening every application for 30 seconds without any interactions, which makes my research still valuable to analyse those apps in more detail.

Apple also added privacy labels in the App Store last year, indicating the types of data collected and processed by applications. The original idea of my research was to get a better insight into which data is being collected, which is what Apple tries to achieve via the labels. However, the developer, not Apple, sets the privacy labels and does not always represent the collected and processed data accurately. I could not take this into account and compare my findings with the App Store labels, which would have been a great addition.

Another reflection point is that I spend a long time on the definition of *popular applications*. I tried to find an ideal algorithm such as multi-criteria and multi-weights algorithms to get a more well-founded, science-based definition of popular applications. I lost quite some time

[58] https://www.engadget.com/53-more-apps-grabbing-ios-clipboard-data-175912135.html

on this part and had to shout out earlier to discuss this problem.

I created a proof of concept of the technical setup to verify the setup and ensure I could intercept and decrypt all data and list the requested domains. This was done by testing a few applications. I also read other studies on best practices regarding this. However, it was only late that I discovered I was missing data from some applications, like Whatsapp. Especially while testing TLS validations, by marking the proxy CA certificate as untrusted, all data transfers should fail unless no TLS validation was performed. I noticed some applications were still working while no data was logged in the proxy server. If I had discovered this earlier, I could have used a different method, like a VPN, to intercept all data. I verified all my data sets, and as only data sent via protocols other than HTTP bypassed the proxy, I could confirm I did not miss much data, luckily.

While I'm quite used to LaTeX, I lost quite some time maintaining the tables in the appendix and adding new columns during the analyse phase. I ended up using a spreadsheet app and regex to convert the CSV back to LaTeX after losing quite some hours.

As for the last point: I'm happy that my own graduation subject was accepted, as I was very interested and curious to know to which extent applications respect our privacy and which trackers are used. I expected fewer applications with unencrypted network requests and hoped to have more applications without additional trackers, or that the ATT setting would be at least more respected.

Bibliography

Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. Third party tracking in the mobile ecosystem. In *Proceedings of the 10th ACM Conference on Web Science*, pages 23–31, 2018. 13, 41

Matius Chairani, Mathieu Chevalley, Abderrahmane Lazraq, and Sruti Bhagavatula. By the user, for the user: A user-centric approach to quantifying the privacy of websites. *arXiv preprint arXiv:1911.05798*, 2019. 35

Tom Chothia, Flavio D Garcia, Chris Heppel, and Chris McMahon Stone. Why banker bob (still) can't get tls right: A security analysis of tls in leading uk banking apps. In *International Conference on Financial Cryptography and Data Security*, pages 579–597. Springer, 2017. 13, 21

Christian J. D'Orazio and Kim-Kwang Raymond Choo. A technique to circumvent ssl/tls validations on ios devices. *Future Generation Computer Systems*, 74:366–374, 2017. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2016.08.019. URL https://www.sciencedirect.com/science/article/pii/S0167739X16302801. 12

Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. Pios: Detecting privacy leaks in ios applications. In *NDSS*, pages 177–183, 2011. 10

Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 289–299, 2015. 35

Catherine Han, Irwin Reyes, Álvaro Feal, Joel Reardon, Primal Wijesekera, Narseo Vallina-Rodriguez, Amit Elazari Bar On, Kenneth Bamberger, Serge Egelman, et al. The price is (not) right: Comparing privacy in free and paid apps. In *Proceedings on Privacy Enhancing Technologies (PoPETS)*, 2020. 6, 14, 21

Konrad Kollnig, Anastasia Shuba, Reuben Binns, Max Van Kleek, and Nigel Shadbolt. Are iphones really better for privacy? a comparative study of ios and android apps. *Proceedings on Privacy Enhancing Technologies*, 2022(2):6–24, 2022a. 13, 41, 46, 68

Konrad Kollnig, Anastasia Shuba, Max Van Kleek, Reuben Binns, and Nigel Shadbolt. Goodbye tracking? impact of ios app tracking transparency and privacy labels. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 508–520, 2022b. 13, 41, 47

Andreas Kurtz, Andreas Weinlein, Christoph Settgast, and Felix Freiling. Dios: Dynamic privacy analysis of ios applications. *Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Technical Reports, CS-2014-03*, 2014. 11

Soo Ling Lim and Peter J Bentley. Investigating app store ranking algorithms using a simulation of mobile app ecosystems. In *2013 IEEE Congress on Evolutionary Computation*, pages 2672–2679. IEEE, 2013. 22

Zulfiqar N Momin and Ahmed Moledina. Methods and systems for tracking users, September 1 2016. US Patent App. 14/625,668. 5

Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X. 509 internet public key infrastructure online certificate status protocol-ocsp. Technical report, 1999. 8

Damilola Orikogbo, Matthias Büchler, and Manuel Egele. Crios: Toward large-scale ios application analysis. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 33–42, 2016. 13, 30

Panagiotis Papadopoulos, Antonios A Chariton, Elias Athanasopoulos, and Evangelos P Markatos. Where's wally? how to privately discover your friends on the internet. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 425–430, 2018. 5

Amogh Pradeep, Muhammad Talha Paracha, Protick Bhowmick, Ali Davanian, Razaghpanah Abbas, Taejoong Chung, Martina Lindorfer, Narseo Vallina-Rodriguez, Dave Levin, David Choffnes, et al. A comparative analysis of certificate pinning in android & ios. In *Internet Measurement Conference*, 2022. 10, 12, 21, 31

Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Johanna Amann, and Phillipa Gill. Studying tls usage in android apps. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 350–362, 2017. 12

Nicolas Seriot. iphone privacy. *Black Hat DC*, 2010:30, 2010. 33

Kunjan Shah. Penetration testing for iphone/ipad applications. *Security consultant foundstone professional services.*, 2012. 12, 21

David Stites and Katie Skinner. User privacy on ios and os x, 2014. 34

Jiexin Zhang. *Hardware and software fingerprinting of mobile devices*. PhD thesis, University of Cambridge, 2021. 34