

Dominik Ziegler, BSc

Private Information Leakage in Mobile Applications

Automated Network Analysis

Master's Thesis

to achieve the university degree of
Master of Science

submitted to
Graz University of Technology

Tutor

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Second Tutor

Dr.techn. Peter Teufl

Institute for Applied Information Processing and Communications (IAIK)

Graz, March 2016

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____
Date Signature

Eidesstattliche Erklärung¹

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____
Datum Unterschrift

¹Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Acknowledgments

At that point I would like to express my gratitude to all those who joined me on this journey.

My heartfelt thanks go to my parents who have always shown great patience and support for my work. It was them who showed me what is really important in life.

Additionally, I would like to express my sincerest thanks to my grandparents and family for their loving nature and their listening and of course cooking skills.

Not to forget my sister for her ambitious (and sometimes rough) nature who always managed to get me back on the straight and narrow.

Last but not least, I would like to thank all my friends for their understanding and the countless hours of laughing and for their spontaneity. Without you, life would be monotonous and boring.

A final thank goes to all my teachers, for their exceptional education not only in technical but also in social skills. In particular, I would like to thank my advisor, Peter, for the countless hours of support, motivation and helpful pieces of advice.

*For my grandfather, Alois,
who could not live to celebrate this moment.*

Abstract

Smartphones should make our lives easier and more comfortable. As a result, new features (e.g. geolocation, camera) and applications taking use of these mechanisms are introduced each year. At the same, smartphones provide location-independent access to the Internet. This combination of easy access to data as well as the permanent access to the Internet, can, however, have an enormous impact on the security of those devices.

For this reason, several techniques have been developed to analyze applications in terms of security or errors. One way to achieve this, is by inspecting the network traffic caused by applications. Compared to other mechanisms this allows to analyze software independently from the operating system. However, due to the vast amount of applications, an automated process is necessary to analyze multiple applications at once.

On that account, this thesis proposes an automatic approach for Internet traffic inspection. More precisely, the purpose of the application was to automatically inspect already captured network-dumps and visualize the results in an easy to use way. Furthermore, the capabilities of the application should easily be extendable to adapt to ongoing changes. In this context we have developed extensions to analyze the data flow of mobile applications. Specifically, we have concentrated on automated detection of private data in captured-network dumps.

Based on this work we have analyzed the top applications of two mobile operating systems in terms of data leakage. Furthermore, we have used these results to investigate whether the two operating systems have different impact on the developed applications.

The results not only show that the transferred data differ on these operating systems, but also that due to different approaches in the permission system, applications on one platform tend to excessively use functions they do not necessarily need. Additionally, we have found that identical applications from the same vendor might set apart from the implementation on the opposing operating system.

Kurzfassung

Smartphones sollen unser Leben einfacher und bequemer gestalten. Infolgedessen werden laufend neue Funktionen (z.B. Ortung, Kamera) bzw. Anwendungen vorgestellt. Gleichzeitig bieten Smartphones einen ortsunabhängigen Zugang zum Internet. Die Kombination aus dem einfachen Zugriff auf Daten mit einem permanenten Internetzugang kann allerdings gravierende Auswirkungen auf die Sicherheit der Geräte haben.

Aus diesem Grund existieren verschiedene Techniken, die die Sicherheit bzw. Fehler der Anwendungen untersuchen. Eine Möglichkeit besteht darin, den Netzwerkverkehr der Applikationen zu erforschen, was im Vergleich zu anderen Mechanismen eine plattformunabhängige Analyse ermöglicht. Um mehrere Applikationen gleichzeitig zu analysieren, benötigt man aufgrund der zahlreichen Anwendungen einen automatisierten Prozess.

Die vorliegende Arbeit stellt einen automatisierten Ansatz für die Analyse von Internetverkehr vor. Ziel der Anwendung war es, Netzwerkverkehr automatisch zu analysieren und die Resultate benutzerfreundlich zu visualisieren. Außerdem sollte die Anwendung möglichst einfach erweiterbar sein, um auf Änderungen reagieren zu können. Deshalb wurden Erweiterungen entwickelt, die den Datenfluss von Applikationen analysieren. Der Schwerpunkt lag dabei auf der Erkennung von privaten Daten im Netzwerkverkehr.

Abschließend wurden die Top-Applikationen zweier mobiler Betriebssysteme auf private Daten untersucht. Die Ergebnisse benutzten wir, um herauszufinden, ob Betriebssysteme die entwickelten Anwendungen unterschiedlich beeinflussen.

Die Resultate zeigen nicht nur, dass sich die übermittelten Daten voneinander unterscheiden. Aufgrund der divergierenden Auffassung hinsichtlich der Rechteverwaltung greifen Anwendungen auf der einen Plattform auch häufiger auf Funktionen zu, welche sie nicht zwingend benötigen. Außerdem haben wir festgestellt, dass sich das Verhalten von Anwendungen ein und desselben Herstellers auf unterschiedlichen Plattformen unterscheidet.

Contents

Abstract	v
1 Introduction	1
1.1 Smartphone Statistics	3
1.1.1 Smartphone Adoption and Usage	4
1.1.2 Internet Usage	5
1.2 Privacy-Incidents	6
1.2.1 Operating Systems	8
1.3 Goals	11
2 Related-Work	14
3 Tools and Frameworks	18
3.1 Burp-Suite	19
3.1.1 Capture-Network Traffic	19
3.2 Play-Framework	21
3.3 Elasticsearch	23
4 Core Architecture	24
4.1 Base-Framework	26
4.1.1 Structure	26
4.1.2 Controllers	28
4.1.3 Elasticsearch	28
4.1.4 Retrieval	29
4.2 Plugins	29
4.2.1 Base Plugin	29
4.2.2 Plugin Registration	30
4.2.3 Result Caching	31
4.3 Network-Dump	32

Contents

4.4	User-Interface	33
4.4.1	Overview screen	34
4.4.2	Analysis screen	35
4.4.3	Comparison screen	36
5	Privacy Leakage Plugins	37
5.1	Prerequisites	38
5.1.1	Private Data Plugin	39
5.1.2	Pre-Processing	39
5.1.3	Filtering	40
5.2	Name Plugin	41
5.2.1	Detection	41
5.2.2	Challenges	42
5.3	Email Plugin	42
5.3.1	Detection	43
5.3.2	Challenges	43
5.4	Username & Password Plugin	44
5.4.1	Detection: Usernames	44
5.4.2	Detection: Passwords	44
5.4.3	Challenges	45
5.5	Credit Card Plugin	46
5.5.1	Detection	46
5.5.2	Challenges	48
5.6	Geo Data/Position Plugin	48
5.6.1	Detection	49
5.6.2	Challenges	49
5.7	Phone Number Plugin	50
5.7.1	Detection	51
5.7.2	Challenges	52
5.8	Session/Token Plugin	52
5.8.1	Detection	53
5.8.2	Challenges	54
5.9	Device Information Plugin	54
5.9.1	Detection	54
5.9.2	Challenges	55
5.10	Sample Data	56

Contents

6	Evaluation	58
6.1	Goals	59
6.2	Limitations and Assumptions	60
6.3	Setup	62
6.4	Top 50 iOS Applications	64
6.5	Top 50 Android Applications	67
6.6	Approach	69
6.7	Analysis	72
6.8	General Comparison: Android and iOS	75
	6.8.1 Incidents	75
	6.8.2 Categories	76
	6.8.3 Results	78
6.9	Comparison: Common Application per OS	81
	6.9.1 Results	84
6.10	Challenges	84
6.11	Notable examples	85
	6.11.1 Location Map	86
	6.11.2 Credentials in Network Traffic	86
	6.11.3 SQL Queries	87
	6.11.4 Passwords	88
7	Future Work	89
8	Conclusion	91
	Bibliography	94

List of Figures

1.1	Smartphone users worldwide in billion. Based on statistics from Statista Inc. [Sta15b]	4
1.2	PC sales, compared to tablet sales worldwide in million. Based on statistics from Statista Inc. [Sta15a]	5
1.3	Malware by Platform 2000-2013 [For13]	9
1.4	Malware by Platform 2013 [For13]	10
3.1	Main user-interface of Burp-Suite 1.6 showing the Toolbar	20
4.1	Framework Architecture	25
4.2	Overview screen, showing all available network-dumps	34
4.3	Analysis screen, showing a detailed analysis result for one or more network-dump(s)	35
4.4	Comparison screen, showing differences between network-dumps	36
5.1	Custom content format detected in the traffic of some applications	57
6.1	Top 50 iOS Application Categories (22. June 2015)	66
6.2	Top 50 Android Application Categories (22. June 2015)	68
6.3	Permission Systems on Android and iOS	71
6.4	Top 50 iOS Applications (22. June 2015) Analysis Report	73
6.5	Top 50 Android Applications (22. June 2015) Analysis Report	74
6.6	Number of detected private data incidents per category on iOS	79
6.7	Number of detected private data incidents per category on Android	80
6.8	Common Applications of iOS and Android (22. June 2015)	83
6.9	Locations of nearby users extracted from the network traffic	87

1 Introduction

With the rise of smartphones and thus mobile applications, access to private data such as geo-location, credit-cards, phone numbers or the like has never been easier. Likewise, various sensors, for instance GPS and camera as well as the permanent access to the Internet, in conjunction with the missing expertise of some users, contribute to the predominant situation of abuse and private-data leakage. In 2014 alone, 2000 new Android malware samples were discovered everyday [Sva14]. Detecting or even preventing malicious applications or careless usage of sensitive data is therefore becoming an important topic when talking about smartphones.

At the same time, mobile platforms such as Android, iOS or Windows Phone, or more precisely, their dedicated app marketplaces speed up the distribution of mobile applications by providing a central collecting point [TKN12]. By the end of 2014, already more than 3 billion apps have been available on all major platforms. Hence, developers nowadays are more likely to build apps instead of providing complete web services. As a consequence, users tend to use apps with network access more quickly [Xu+11]. This trend in combination with the amount of private data available on mobile platforms leads to new attack vectors. Researchers or antivirus companies are therefore investing a lot of time and money in detecting malicious or poorly implemented applications.

Because of this concerning development in malware and the growing access to sensitive information on mobile operating systems, the present thesis deals with the topic of private data-leakage of modern smartphones. In particular, great attention has been paid to not only detect private information in mobile applications (e.g. phone-numbers, user names, passwords, etc.) but also to figure out implications on the transmitted data due to operating system peculiarities like permission systems.

1 Introduction

One way to achieve this is by analyzing the application and therefore understanding the nature and intentions of software. Traditionally, techniques such as static code analysis, dynamic code analysis or network monitoring are used to gain insight into the data flow of an application. However, in general code analysis is too vigorous to perform in large operations. This is aggravated by the fact that most platforms limit the capabilities of applications and therefore do not permit reverse engineering or measurements in the background on the device itself. Additional steps to achieve reverse engineering on mobile platforms are therefore almost always necessary.

Another approach to get an overview of the data flow is to analyze the network traffic caused by applications via software like Wireshark¹ or Burp-Suite². The drawback of these tools however is, that they require a certain kind of expertise and most certainly experience in this field. Additionally, analyzing and interpreting the so captured outgoing and incoming traffic is a very time consuming and challenging task to undertake. Evidently, this process does not scale due to the significant number of available applications.

Using a more automated approach in network analysis, one could facilitate this procedure and therefore help specialists or security researchers gain insight into the application's nature. However, in spite of the increasing number of apps and users and the fact that a majority of these smartphone applications rely on HTTP or HTTPS protocol, which is a well defined standard, it is still a sophisticated task to automatically analyze the traffic and therefore the nature of these applications. That is, because most smartphone apps generally do not rely on a common terminology or way of data representation. More precisely: mobile services may use existing technology like JSON, multipart/form-data or simply plain-text to encode their content and prepare it for transmission, but there is hardly any generic approach on how to interpret and understand all the transmitted information. For example: if a developer decides to omit any describing parameters for the transmitted data, it is undoubtedly rather complex to automatically gain complete insight into the submitted information. In fact, we nowadays have less understanding of the underlying technology and protocols used in

¹<https://www.wireshark.org>

²<http://portswigger.net/burp/>

1 Introduction

smartphone applications as compared to web services [Cal+09].

On that account this thesis proposes a way to facilitate this process. Hence, the presented architecture takes the output of different network analyzer tools such as Burp-Suite (see Section 3.1) and is capable of automatically processing and presenting it in a more structured way. For example, it is possible to extract certain predefined information like geo-location or search in the provided dump for leaked data. Furthermore, the framework is able to automatically detect data breaches or just simply list used technologies or protocols. To achieve this goal, a modular architecture is inevitable. Henceforth, the functionality of this framework can be further extended by so-called plugins, each serving exactly one purpose (e.g. username-detection, email-address detection etc.).

However, to combine various detecting mechanisms, a great deal of effort has to be put into each module of the framework, which would most certainly go beyond the scope of this thesis. For this very reason and to provide a complete experience, a common core providing basic features, such as validating and processing the input, has been developed first. This core has been created by Mattias Rauter, Christof Stromberger and Dominik Ziegler. As mentioned above, the modular nature is a major aspect of the framework. As such it is possible to divide the development process into clearly structured tasks. By focusing on distinct fields of interest, e.g. tracking-frameworks, authentication mechanisms or private-data leakage, different extensions for the core framework can be developed without interfering the work of others.

1.1 Smartphone Statistics

The proposed framework automatically analyzes and processes captured Internet traffic. Although this technique is not limited to mobile devices and can equally be used with desktop applications, this thesis focuses primarily on smartphone- respectively mobile-traffic. The following gives an overview of the development and why it is also important to also concentrate on mobile-traffic in the upcoming years.

1.1.1 Smartphone Adoption and Usage

As of this writing, 1.91 billion people worldwide are using smartphones, with an estimated number of 2.56 in 2018 [Sta15b]. Figure 1.1 illustrates this development. Compared to the year 2007, which marks the beginning of the smartphone-boom with the release of the Apple iPhone and soon afterwards of Android, with roughly 122.32 million devices, nowadays more than 15 times as many people are using smartphones.

If we look at the global smartphone sales since 2007 this fact becomes even more obvious. By the end of 2014 roughly 1.2 billion devices have been sold to end users in that year alone. According to estimates this figure will increase to approximately 1.95 billion p.a. by the end of the year 2019.

Comparing this number to the personal computer sales (including desktops as well as notebooks) with an estimate of 123,1 million in 2017 (157 million in 2010), we can clearly see a decrease in sales [Sta15a]. Figure 1.2 gives an overview of this development as well as a future outlook. For the stated reasons we can thus safely assume that the number of smartphones will rise even more rapidly over the next couple of years. Directing the focus on mobile internet traffic is therefore becoming an important factor.

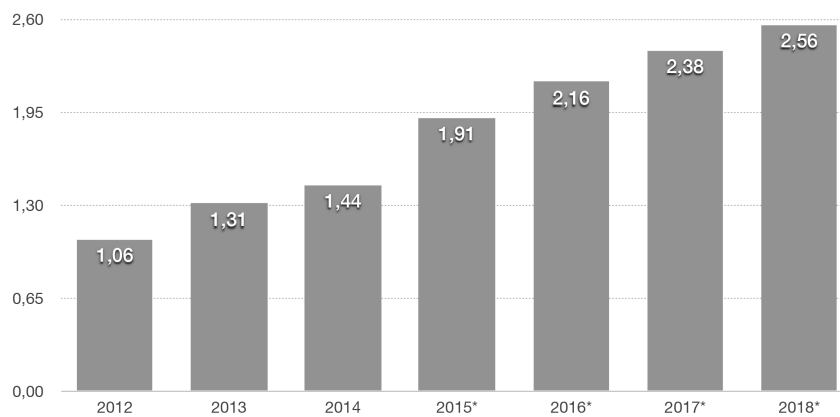


Figure 1.1: Smartphone users worldwide in billion. Based on statistics from Statista Inc. [Sta15b]

1 Introduction

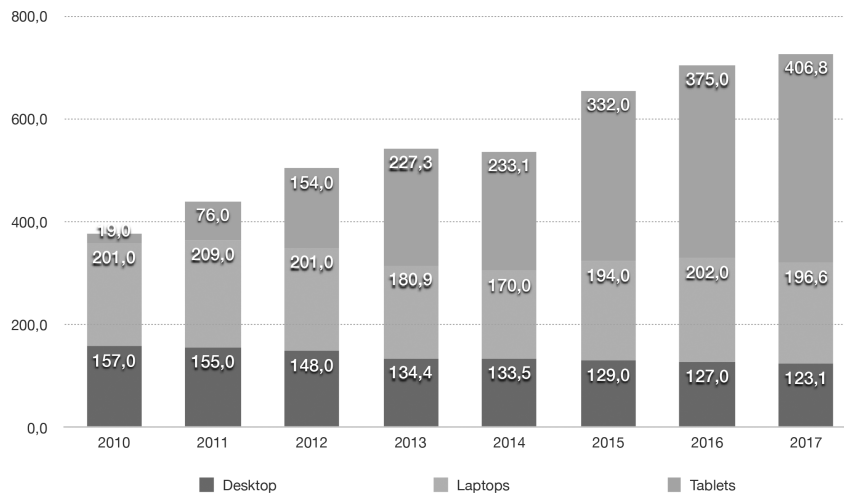


Figure 1.2: PC sales, compared to tablet sales worldwide in million. Based on statistics from Statista Inc. [Sta15a]

1.1.2 Internet Usage

Estimates predict that by the year 2020 traffic caused by smartphones will have increased exponentially to circa 17 exabytes, that is 17 000 000 000 gigabytes per month [Sta14b]. In 2014, this number was still at about 2.1 exabytes per month. If we look at the smartphone adoption statistics (see Section: 1.1.1) this development can easily be explained with the predicted number of smartphone sales on the one hand and with the increase in data consumption per user.

Looking at these figures in detail, this results in a total of approximately 3.5 gigabytes of internet traffic caused by smartphones per user per month.

Certainly, these numbers are based on predictions and may vary from country to country and over the time. Still, as they are based on the current developments they are a good indicator on what to expect.

1.2 Privacy-Incidents

It should be noted at this point that despite the fact that there already exist considerable approaches to reduce or detect malicious or faulty software still a noticeable amount of malware-related incidents are reported each year. Additionally, not all threats can be detected or even prevented by application certification or malware detection frameworks. In some cases, deception (e.g. Phishing) and lack of knowledge or awareness can be the reason for private data theft [Ira+08] [Bod13]. Moreover, sending sensitive information over the Internet can be a required process in certain situations like for example authentication mechanisms.

It is therefore all the more important to know in which kind of data attackers are most interested in. Consequently, it may be of great help to get an overview of different types of malware in order to gain a better understanding of malicious software and attack vectors. Additionally, it is important to note that (mobile operating) systems constantly change and evolve. As do their security mechanisms. It is therefore equally important to look at recent privacy incidents to see how current malware collects and transmits (private) information.

However, when talking about malware, it is first of all crucial to know which kind of malicious software even exists. In general, harmful software can be split into three categories [Loo14]:

Malware: Malware is software designed to steal (personal) data from devices and profit from them. Many times, it is the main cause for financial fraud or unresponsive devices. It is the hypernym for different kinds of software such as viruses, trojans, worms or ransomware. When dealing with malware, network analysis can be of great help to identify such software, because most of the time, it transmits data over the Internet in order to communicate with or send acquired information to a server operated by attackers. Theoretically, it is possible to detect or at least contribute in a meaningful way to help identify most of the current malware implementations via network analysis.

Chargeware: The aim of chargeware is to generate profit for its operator and therefore usually entails high costs for the user. In the majority of cases,

1 Introduction

the user is not informed about these billings, which usually happen without proper notice or the user's approval. Depending on the implementation details it can be fairly challenging to detect these threats via network analysis. That is, because all too often, chargeware uses conventional techniques like premium SMS or premium rate numbers to generate profit.

Adware: In the majority of cases, adware displays importunate content on the user's screen and reduces the overall user experience. Additionally, adware most commonly collects personal data in large amounts and sends it to a backend. Even though adware might not pose a risk to the user from a financial point of view or in terms of device damage, stealing personal data can be equally dangerous as they intrude into private lives.

By knowing different types of malicious software it is equally important to understand how it is utilized on smartphones in order to know the capabilities and limitations of network analysis. In brief, when looking at recent threats, we can detect a trend towards malware, in particular ransomware which is a new kind of attack, which emerged in the last years. Ransomware is designed to prevent the use of a device by locking it down, or encrypting its contents. The victim often needs to make payment in order to regain access to their computer or phone. Most of the time, ransomware is installed as drive by download, meaning it disguises as versions of legitimate programs in order to delude the victim. While at first glance network analysis might not be of great help when dealing with such software, eventually ransomware will have to communicate with a server in order to transmit the encryption key or to send device or user related information. This traffic can then be captured and analyzed.

Another concerning observation has been made by researchers who found that some malware came preinstalled on certain devices. This implies that attackers must have infiltrated the supply chain and have gained full control over the installed software. By posing as ringtone apps or the like, this kind of malware then tries to capture login data by displaying forged text messages. However, this behavior may most certainly be detected with network analysis.

Likewise a trend towards forged app store software has been observed. After installation, this software runs in the background to intercept phone calls or text messages. Obviously, without network access an attacker would

1 Introduction

draw no benefit from this kind of attack. The collected data therefore have to leave the device at a certain point. As a result is possible to identify those kind of attacks.

Last but not least, security analysts have detected software which uses the processing power of smartphones for crypto currency such as Bitcoins or Litecoins. Although the computation power of modern smartphones is still not fast enough, calculating challenges for crypto currency can be possible by accumulating a large amount of smartphones.

Interestingly, most of the identified threats have been observed and been predominant on the Android platform in last year. However, while clearly not all threats can be detected or even prevented by network traffic inspection, most malware can be identified by inspecting the sent data of infected devices. This is based on the fact, that attackers expect some sort of benefit from their software. As a result, most of the collected data has to be sent over the Internet to an endpoint. This shows that analyzing network traffic can be a valuable tool and of additional help to detect those kinds of attacks. Furthermore, we can see that attackers are most likely to be interested in user-credentials or content of text-messages (e.g. for mobile banking).

1.2.1 Operating Systems

As anticipated, all major incidents recorded in the last three years emerged on the Android Platform [For13]. Figure 1.4 gives an approximate overview of the prevailing situation. According to Svajcer [Sva12], there are two reasons for this. On the one hand, the Android operating system has gained large popularity over the last years, with more than 1 million activations per day [Sva12]. On the other hand, attackers try to distribute the applications very quickly and widely. So-called application marketplaces³ provide a perfect environment for malicious applications. Typically, these app stores entail a particularly large user-basis, which in return implies a high possibility of installation. Moreover, most of the time malware uses several techniques to avoid detection by the application marketplaces' security mechanisms. And this is the crux of the matter.

³Centralized solution, offered by vendors in order to distribute applications

1 Introduction

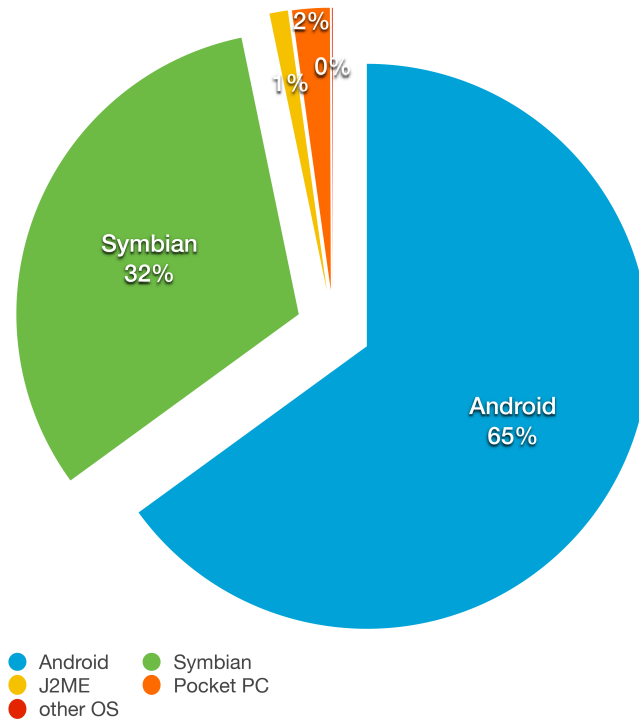


Figure 1.3: Malware by Platform 2000-2013 [For13]

In general, two different approaches on how to analyze and certify applications suitable for app store distribution are used in practice, both evidently having advantages but also disadvantages.

On the one hand, there exists a manual approach, meaning that applications respectively the code are partly or completely analyzed by hand. Still is possible under certain conditions to hide functionality in applications which might be missed. Additionally, just recently there have been reports where manual app certification has failed on a large scale [Ben15]. Certainly a manual certification approach is not 100% secure, but it at least reduces the risk and aggravates the situation [Sva12].

On the other hand a more automated technique is used. As a result, special software designed to find flaws or misuse of certain functionality in applications is applied. Obviously, companies are not releasing detailed

1 Introduction

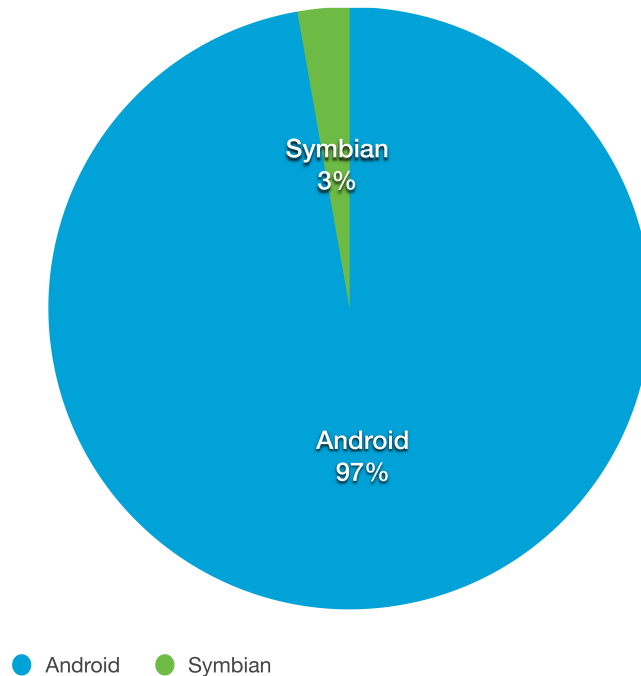


Figure 1.4: Malware by Platform 2013 [For13]

information on this process, in order to not reveal security related parts and to keep up with attackers. However, this process has also proven its weakness, as just recently attackers managed to circumvent automated security mechanisms and implant malware into an application store [Fox15].

Now talking about the current mobile operating systems, it is assumed that Apple and Microsoft rely on manual app certification processes to a certain extent [App15][Pla13]. In contrast, Google claims to use a more automated technique for the Android Platform and its Google Play Store⁴, namely Google Bouncer [Loc12]. Indeed, based on experience, publishing and updating an application for the Google Play Store can be achieved within a couple of minutes, which is a good indicator for automated analysis.

Additionally, besides the official Google Play Store, there exist third party solutions to install applications, on the Android operating system. Big

⁴<https://play.google.com/>

1 Introduction

companies and vendors like Amazon⁵ benefit from this opportunity and distribute applications via their own marketplaces, a fact not hidden from attackers. Especially in China where access to official stores is often not possible, third party solutions become the only way of discovering new applications [Ng+14]. Approximately more than 70% of devices in China receive content from unofficial app stores like Blackmarket⁶. Typically, attackers use these platforms to create free versions of otherwise paid applications and implant malicious code this way.

Another reason, why Android is as popular among attackers is its operating system update policy. At the moment, a device running Android can only receive official core updates through its manufacturer. As a consequence, all too often cheap devices do not receive any security updates after a certain period, even though a newer version of Android has been developed, leaving this device vulnerable to possible attacks [Ama14].

Summarizing, we see that when dealing with malware, special care has to be taken when analyzing applications on Android. Naturally, this does not imply that there is no malware on Apple or Windows smartphones at all. The prevailing situations and above mentioned circumstances, however, show that at the moment attackers are heavily concentrating on Google's operating system. However, analyzing network traffic is more or less operating system independent. We can therefore safely assume that there is no big difference when inspecting data created by any mobile operating system.

1.3 Goals

Over the last years, an increase in new malware types like ransomware has been noticed [Cis14]. This can be attributed to the fact that carriers as well as vendors try to put a stop to conventional malware approaches [Blu14]. Mechanisms such as the elimination of premium SMS forces attackers to come up with new ideas. Especially the trend to ransomware is concerning, as this kind of software usually means high costs to the users and leaves

⁵<https://www.amazon.de/gp/mas/get/android/>

⁶<http://www.blackmart.us>

1 Introduction

the devices unusable [Fse14]. Likewise important is the fact that attackers were able to infiltrate the mobile supply chains and could thus manage to install malware prior. There is hardly any protection possible against these kinds of attacks. Consequently, it is all the more important to be on one's guard and to abstain from installing applications from untrusted sources. Antivirus software will therefore become even more important over the next years [Sec13].

However, not all incidents concerning private data theft can be prevented this way. Sometimes, lack of knowledge of software developers can also cause unwanted leakage of sensitive information.

The goal of this investigation became to implement an application to cope with the current development of malware, as well as to provide means of automatically finding private data such as usernames, passwords or the like in captured network traffic. As a result, extensions for the developed framework to simply display or detect these data have to be developed. Another important point was to design the software in such a way that it is able to analyze network traffic independently from the operating system of the smartphone.

However, when comparing the network traffic of different operating systems, or more precisely, the applications running on it, it is equally important to talk about permission systems. In general, there exist two different approaches on how to control the execution of applications, limit device capabilities or prevent unauthorized usage of sensors and private data.

On the one hand there are permission systems which perform on access level, which means that with the first attempt to access private information (e.g. GPS, photos, etc.) the user is notified and may or may not grant access. On the other hand an "all or nothing" approach is utilized. This permission system asks the user upon installation to allow access to needed sensors or functionality. Once installed, the application may make use of these mechanisms as often and whenever necessary.

Generally speaking, both systems have certain benefits. However, the goal of the study is to see, whether these different approaches have an impact on the developed applications and on uncontrolled, that is without proper notice, usage of data.

1 Introduction

This leads to another important aspect. Not only are we interested to see whether private information has been sent by an application, but also if the amount and type of data varies significantly between different operating systems.

It can be said, the Android operating system has become the main target for attacks these days. However, not every device is infected with malware or has access to third party application stores. For this reason, one aspect of this thesis, was to find out how permission systems impact the overall data of conventional, in other words non-malware applications.

Summarizing, we can say that even though network analysis cannot prevent or detect malware, it is not only an essential process in identifying unwanted software but should also help identify poorly implemented applications which do not provide enough security mechanisms to guarantee data safety. As such, the intention of the presented application and the developed plugins was not to serve as a malware detection platform, but rather as a set of tools to detect exactly which data is sent over the Internet. Especially on the Android platform, where permissions (e.g. Internet access, user data, contacts) are often needed for certain functionality, it is sometimes not clear whether these permissions are really necessary or not. For example: Internet access in combination with access to contact data does not necessarily mean that this contact information is sent over the Internet. Then again, having no Internet access does not automatically mean that private information cannot be leaked. It could be possible that this data is shared with another application, which in return does have the permission to send information over the Internet. However, the corresponding application in return does not require access to private data.

2 Related-Work

Privacy is becoming an important topic when it comes to analyzing smartphone applications. However, as of this writing there are different approaches on how to dissect and inspect software, each focusing on a specific use case.

In general, we can distinguish between two types of analysis methods. First of all, there is code analysis, an easy way to inspect data flow either at runtime, called dynamic analysis, or on existing source or binary code, which is called static analysis [RS10]. One of the main advantages of dynamic code analysis is, that it is possible to detect and track data flow throughout an application. This way, one can detect unwanted or uncontrolled access to data or certain functionality. The drawback of this method however, is that a program needs to be executed on a device itself. As a result, a mechanism to virtually execute an application in a sandbox or similar has to be developed first. Additionally, dynamic analysis usually does not provide a complete report of an application, as only the parts which are executed can be inspected. This way, it is possible to hide certain functionality from dynamic analysis.

On the other hand, static analysis can be performed on existing code alone, making it extremely powerful and versatile to use. However, by using only static analysis we usually can not trace every movement and sometimes might miss coherences of data.

Typically, by combining both static and dynamic analysis we can achieve much more accurate reports than using each technique alone. Then again, the effort necessary to accomplish this is usually fairly high.

Naturally, different approaches for both dynamic and static analysis have been presented over the last couple of years. However, due to the specific

2 Related-Work

nature of code analysis, most of the time additional steps like unlocking a device have to be made in order to achieve results. Additionally, code analysis heavily depends on the operating system itself and therefore the programming language used. That said, not all languages allow a complete reverse engineering of already compiled code.

In contrast to this, we can also analyze applications via monitoring the network traffic. Admittedly, this reduces the accuracy of an analysis report to only the parts which were sent over the Internet. However when talking about private data we are most likely to be interested in only this information which is transmitted to another party. In other words, if smartphone applications do not send any data over the Internet, we can almost be certain, that no private data is sent either.

Another point, which supports the usage of network analysis, is that it does not depend on the underlying platform. As a result, it can be used on any existing and possibly future device, making it very stable and reliable to ongoing changes. Using network analysis to detect applications which sent private information is therefore a reliable and fast way.

Hence, Enck et al. [Enc+10] presented an approach to completely monitor the network traffic of Android applications in real time. They propose an application called "TaintDroid" which virtually executes the programs to be analyzed and is therefore able to detect outgoing private data. TaintDroid uses a combination of various analysis techniques to track private information. By "tainting" this data, it is possible to find information once it leaves the device and trace it back to its origin. However, TaintDroid requires a rooted¹ Android device and can therefore not be used on every smartphone.

In contrast, Falaki et al. [Fal+10] presented an approach in 2010 to analyze the network traffic caused by smartphones. They showed that roughly 50% of traffic is caused by browsing, while applications like mail or messaging contribute to the other half. In their findings, they state that packet loss is the primary cause for slow connections and current server configurations are not optimized for smartphone traffic. Additionally, they propose mechanisms to improve network overhead and power consumption. However, this approach

¹Elevated User Rights

2 Related-Work

is not generic and does not provide easy methods to extend the applications functionality to detect certain kind of data.

Stepping further into the topic, Callado et al. [Cal+09] explain in their survey the main problems of network traffic analysis. In their work, they first of all split the traffic into two categories namely 'packet-based' and 'flow-based', and compare the results. Moreover, they give an overview of traffic analysis techniques like signature-matching, sampling and inference as well as their own proposal. Last but not least, Callado et al. [Cal+09] give an outlook of future trends. Nevertheless, this process focuses heavily on inspecting the kind of application and does not analyze the data itself. As such, it is not possible to detect certain kind of information in the analyzed network traffic.

On the other hand, several approaches have been presented to overcome the previously mentioned issue of malware infected devices or malicious applications. Desmet et al. [Des+08] identify the permanent access to the Internet and the capability of modern smartphones to execute untrusted code as the fundamental dilemma. They propose a new technology called SxC (Security-by-Contract) which enforces the permissions given to untrusted code. As a consequence, third-party code can not escape from its sandbox or elevate its rights. At the same time however, it is still possible to develop applications which collect private data within its given boundaries. This solution therefore obviously cannot solve the problem of private data leakage alone.

While a lot of research is conducted in the field of protecting the user and their data, Ongtang et al. [Ong+12] point out that protecting the application itself is equally important. This implies that access to interfaces offered by applications must be protected from improper use and permissions should only be given to those applications which can be trusted. To improve the situation on the Android Platform, they propose the Saint (Secure Application INteraction) framework, which, according to them, enhances the operating system by these missing features. Again, this solution is very appealing, but is very specifically designed for one platform. Additionally, it does not prevent from the possibility of sending private information over the Internet, be it by accident or not.

2 Related-Work

Enck, Ongtang, and McDaniel [EOM09] add that malware on mobile devices is becoming a more serious problem each year. As the devices become more powerful and systems become more complex, attacks are becoming more spacious. Application certification, which means precisely analyzing the code, could restrict these problems. However, this process is missing on the Android platform. Hence, they propose Kirin, a security service, to limit malware applications during install time.

All in all, we have seen a lot of outstanding work being conducted in both directions, analyzing network traffic and attempts to restrict and control applications and prevent malware. However, most of the above presented work is very platform specific or concentrates on one field alone. Moreover, means to easily extend the functionality of the proposed design to either support more operating systems or to react to ongoing changes are mostly missing.

For this reason, the proposed design tries to close a gap between platform specific (private) data detection and existing tools widely used for network analysis. One of the main advantages is certainly the possibility to respond to current developments by extending its functionality.

Furthermore, as we are analyzing the network traffic alone, the tool can be used independently from the underlying platform. As a result, hardly any work is necessary to adapt to operating system changes. Another important point is that devices do not need to have elevated user rights.

Summarizing we can state, that due to the nature of network analysis some of the presented work indeed will yield better reports in terms of accuracy or detection rate. However, the proposed design is perfectly suited to provide a very detailed overview of used technologies and especially data flow. At the moment, there is no such generic approach, which is also easily extendable at the same time.

3 Tools and Frameworks

To guarantee a platform independent user experience, the proposed application is developed as a standalone web-application. As such, it is possible to run this tool platform independently regardless of the operating system or the browser. Additionally, this software can be deployed on a central server and can thus be further developed as a collective tool to analyze applications and create a data pool of interesting network dumps. Generally speaking the modular architecture allows further development without much effort, as will be described in Chapter 4.

However, the proposed design consists of several factors which we need to take into account first. One of them being interoperability. For this purpose, the presented work takes advantage of existing tools or at least provides possibilities to include them in the application, in order to facilitate the development process as well as to reduce the need to get accustomed with new software or mechanisms to a minimum.

With this intention, relying on existing network analysis tools was one decision. Especially for Internet traffic there exist several tools like Wireshark, or Burp-Suite, which both have been widely used by many researchers. Additionally, this software has been tested in real life scenarios and proven its functionality. However, because of the special use case of smartphone application analysis, we have decided to rely on Burp-Suite in the first case. This is, because Burp-Suite automatically provides functionality to capture encrypted Internet traffic. A detailed description of Burp-Suite will therefore be given in the following.

At the same time, it is just as important to have a good core for web based applications. Keeping the goal of platform independency as well as easy scalability in mind, we have decided to rely on Play-Framework, a Java based toolset designed for large web-applications.

By providing interfaces for these tools and connecting them in a meaningful way, it is therefore possible to not only easily analyze newly captured smartphone traffic, but to also use existing network dumps and have them analyzed.

3.1 Burp-Suite

The proposed framework provides, as of this writing, no possibility to capture network traffic or create network dumps of (mobile) applications from within the tool itself. As such, it depends on input from existing solutions. One of them being Burp-Suite¹. The benefit of this approach is that established products can still be used and existing workflows do not necessarily have to be changed. Settling-in-periods to new products can therefore be omitted.

In the proper sense, Burp-Suite is not only a tool to create network-dumps but a complete toolchain for security-testing. Thus, it provides mechanisms to capture encrypted traffic, which becomes heavily important when trying to analyze applications which rely on the HTTPS-protocol. In order for this to work, a Root-Certificate first has to be imported into the device's certificate chain. Burp-Suite can then create on-the-fly certificates for the captured packets and intercept or even modify secured network-requests or -responses. Still, devices used to capture network-traffic are not vulnerable to attacks from another party, as each instance of Burp-Suite creates its unique Root-Certificate. Additionally, the requests have to be rerouted to the machine running Burp-Suite. For a conventional analysis process this is realized via a proxy setting on the device itself, as described in Section 3.1.1.

3.1.1 Capture-Network Traffic

When analyzing network traffic, the capture settings are fairly important. This section deals with the basic settings of Burp-Suite, used to analyze the

¹<http://portswigger.net/burp/>

3 Tools and Frameworks

applications mentioned in Chapter 6.

As of version 1.6 the main user-interface offers twelve different tabs for the user to select from (see also figure 3.1).

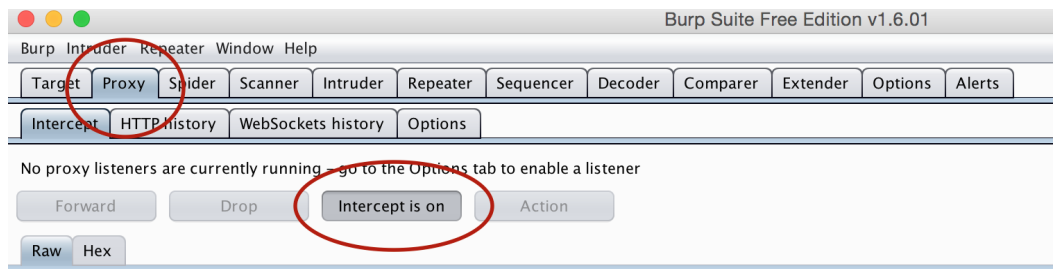


Figure 3.1: Main user-interface of Burp-Suite 1.6 showing the Toolbar

To actually capture traffic a few adjustments have to be made:

- **Proxy Listener:** Typically a proxy-server redirects requests from one machine to another. While capturing network-traffic, Burp-Suite acts as a proxy listener to capture the requests sent from the device. An important factor when trying to analyze smartphone traffic is to set the address to "specific" in order for the smartphone to detect the proxy.
- **Intercept Client Requests/Server Responses:** In order to capture all traffic, the filters have to be adapted. Creating two filters for both HTTP and HTTPS protocols has proven effective.
- **HTTP-History:** Once the dump has been created, it has to be exported in a suitable format. This can be achieved through the HTTP-History tab. Important for this matter is to set the filter to the desired value. The network-exchange list can then be exported to a suitable format for the developed framework via the context menu.
- **Certificate-Pinning:** Although certificate pinning cannot be circumvented using Burp-Suite, it is an important aspect when analyzing traffic, secured via HTTPS. In general, certificate pinning describes a technique used in applications to prevent so-called man-in-the-middle

attacks, where attackers try to intercept a secured connection by introducing their own certificate. During and before each request the trust status of the submitted X.509 certificate is checked. If this is not the case e.g. an intermediary certificate generated by Burp-Suite has been submitted, the communication is aborted. As a result, traffic from applications which rely on certificate pinning cannot be captured.

3.2 Play-Framework

The Play-Framework² is an open-source framework to create powerful web-applications. It is written in Scala but provides programmatic interfaces for Java. Development can therefore be done in either of those language or even a mixture of both. It is thus even possible to develop web-applications completely written in Java. In addition, the Play-Framework builds upon the Model-View-Controller Pattern, which has the ultimate goal to simplify unforeseen changes or additions during the development process. The user-interface is more or less independent from the underlying data structure or data logic. The architecture of the proposed application builds upon and uses this pattern and will be described in Chapter 4.

What led to the decision of using the Play-Framework and what makes it ideal for the proposed application is described in the following:

- **SBT:** Scala SBT³ is an open-source build toolchain to automate compile tasks. Similar to ANT or Maven, it supports tasks like library dependency management or build instructions written in a more abstract language. SBT builds upon the Scala language, but supports natively both Scala and Java code. The Play-Framework uses SBT as its preferred build tool.
- **Code Refresh:** Unlike most web-application frameworks, the Play-Framework supports automatic code refresh. In other words, stopping, redeploying and restarting the server during the development process

²<http://playframework.com>

³<http://www.scala-sbt.org>

3 Tools and Frameworks

is not necessary. Compile errors are displayed directly in the browser window. As a consequence, an increase in productivity during the development process has been observed as the deployment effort as well as the time need to compile could be reduced to a minimum.

- **Database:** The Play-Framework comes with built-in relational database support. As a consequence, writing database queries becomes virtually obsolete, and it is possible to concentrate on more important tasks. In order to create or modify the database structure, conventional Java Classes with the corresponding annotations are sufficient. In addition, special methods are provided to update, insert or delete existing data. Moreover, manual database evolutions can be created in addition to the automatically generated structure to support database changes in a productive environment.
- **Integrated Webserver / Deployment:** What makes the Play-Framework ideal is the integrated JBoss Netty⁴ web-server. In this way, it is not absolutely necessary to install or setup an Apache⁵ or Tomcat⁶ server during development process or during production phase. Still, if preferred the source can be packaged in a conventional WAR-File, an archive file with all necessary files, and deployed on above mentioned servers.
- **Distribution:** The adaption of the Play-Framework shows that the architecture is highly flexible and scales even for big projects. Major companies like LinkedIn⁷ or The Guardian⁸ already rely on the architecture.

⁴<http://netty.io>

⁵<http://httpd.apache.org>

⁶<http://tomcat.apache.org>

⁷<https://www.linkedin.com>

⁸<http://theguardian.com>

3.3 Elasticsearch

When dealing with network traffic, the captured dumps can quickly grow rather big in size. As a result an effective way to store, retrieve and search through this immense amount of information is necessary, in order to perform any kind of analysis procedure.

However, even though the Play-Framework comes with inbuilt relational database management (see 3.2) the proposed design uses Elasticsearch⁹ to store, search through or retrieve data. In short, Elasticsearch is a search engine which has been optimized to work with large datasets and provide full text search. A feature most relational database systems have not been optimized for. Additionally, Elasticsearch comes with an inbuilt web-server and a RESTful API to perform (real-time) search-queries.

As a result, the uploaded network dumps can not only be analyzed from within the framework itself, but can be made publicly accessible without any effort at all, hence, providing sustainability and the demanded interoperability between existing and future applications.

⁹<https://www.elastic.co>

4 Core Architecture

In general, the whole architecture consists of four substantial parts (Base-Framework, Plugins, User Interface, Network-Dump), each being a closed system working on its own. As a result, modifying, adding or even exchanging certain components does not affect the overall system at all. Figure 4.1 illustrates these elements and gives an overview about their relation.

The core component of the system consists of the Base-Framework, which connects all relevant parts of the application. To cope with the goal of a modular structure the basis serves as a central collection point and provides means of communication between the plugins (see also Chapter 5) and the data logic. Thus, it is possible to change or extend the functionality of the Base-Framework without the need to modify the plugin structure or vice versa. Besides, it is possible to make changes to the user interface or change the visualization of plugins without interfering in another scope.

Speaking of plugins - which function it is to analyze the data set - we can clearly see that they take up the main task. Ideally, each plugin serves only one purpose. However, by combining multiple of those extensions we can easily achieve detailed and deep levels of analysis.

Furthermore, by separating the data representation from the analysis results, we can create an additional layer of abstraction. This way, we can interpret and visualize analysis reports in any desired form. For instance, it is possible to display one and the same result in multiple ways, like graphs, overall statistics or means without the need of recalculating the result again.

Last but not least, it is important to note that in general, an arbitrary program can be used to capture network traffic. This allows the system to be versatile in respect of interoperability between existing solutions and the provided tools.

4 Core Architecture

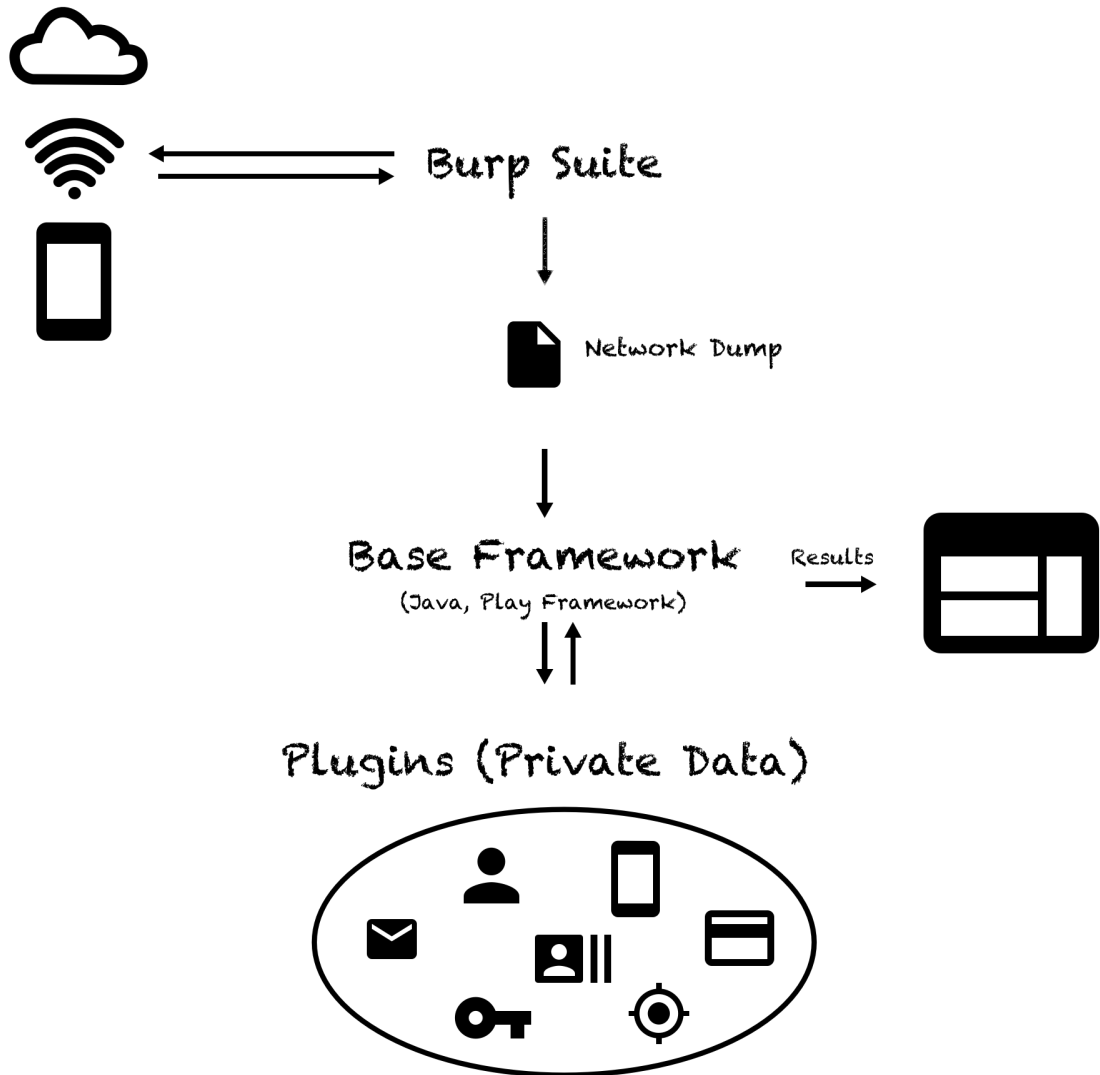


Figure 4.1: Framework Architecture

4.1 Base-Framework

The Base-Framework builds the core of the system and as such, is responsible for connecting and processing the remaining components. First of all, one of its major tasks is parsing and converting the provided network dumps into a system-wide defined format. In this way, changing the application responsible for capturing the network traffic will not affect the remaining system. Additionally, it is possible to support multiple different network-dump input formats without great effort.

In the next step, the Base-Framework is responsible from registering all detected plugins which is achieved by scanning for all possible extensions in the project folder that follow a given protocol. This is to make sure, that the plugins work to a certain extent. Additionally, in this way, the Base-Framework does not have to be changed if a new extension is developed or an existing is removed. Simultaneously, it is possible to deactivate or activate plugins via the core if not needed to save unnecessary computation power.

Another task of the Base-Framework is to pass on the results of the analysis plugins to the user interface respectively to inform the user interface of their availability. By decoupling the visualization from the Base-Framework we could achieve a much higher versatility of all parts while still providing the same functionality. This again shows the modular architecture of the overall system as well as the interchangeability of the components.

Summarizing, we see that the Base-Framework essentially plays a vital role while still leaving all important work to the according modules. Nevertheless, the core has some further details, which will be dealt with in the following.

4.1.1 Structure

Remember that the developed design works as a standalone web-application and builds upon the Play-Framework, a Java based toolset to quickly

4 Core Architecture

build large scale web-based software. Generally speaking, a typical Play-Framework project structure looks as follows:

```
app/  
  controllers/  
  models/  
  views/  
conf/  
  application.conf  
  ...  
build.sbt
```

Although the above mentioned structure represents only a portion of all project files, it simplifies the view of the project and will therefore be explained in the following:

- **Controllers:** In the Play-Framework controllers serve as the logic layer. For a good structure each controller implements or combines methods within the same context. It is therefore easy to maintain the code or find certain functionality.
- **Models:** In general, each model in a Play-Framework application represents an entry in an object-relational-database (ORM). As mentioned in Section 3.2, the Play-Framework comes with built-in database support. Still, we decided to rely on Elasticsearch to store and retrieve data. Thus, models in the proposed design do not implement the common functionality as one might be used to (see also 4.3).
- **Views:** To follow the Model-View-Controller pattern, each view is independent from the data-logic. As such, a screen represents a single landing page to display. This way, the user-interface can easily be changed without modifying the underlying data structure.

4.1.2 Controllers

Controllers play a big part in any Play-Framework application, as they build the base logic of the underlying software. As a consequence, three controllers, `Application`, `FileController`, `PluginController` have been developed. Generally speaking, each controller is intended to combine data logic which belongs together in one single place.

Application

The `Application-Controller` handles the routing of the visible user-interface URLs, meaning it renders all parts of the application which are visible to the user. This includes the depiction of the uploaded network dumps or their parse results as well as error pages like "not found" or the like. In general, the `Application-Controller` does not include much logic but is vital for the application's functionality.

File Controller

The `FileController`'s sole purpose is to handle requests related with files. This includes handling and parsing network-dumps or retrieving them from Elasticsearch. However, the `FileController` itself does not provide any functionality to analyze the uploaded dumps.

Plugin Controller

The `PluginController` is a very lightweight controller and is utilized to handle plugin registration and common tasks used within the context of plugins.

4.1.3 Elasticsearch

Elasticsearch plays a vital role in the developed application. Not only is it used to store and retrieve the uploaded Network-Dumps from the database, but it also serves as a pre-filter engine to arrange them. Especially in applications with large database-objects, elastic-search can contribute to a

huge performance boost, particularly when performing real time search operations [Seb14].

4.1.4 Retrieval

An important factor during the development process was the interoperability and openness between third party applications. As a consequence, it is possible to retrieve or search through the uploaded network-dumps via the Elasticsearch query API.

4.2 Plugins

Plugins play a huge role in the developed framework and build the core of the analysis process. We have shown that the controllers in the proposed design do not figure prominently. Instead, all the logic in analyzing the uploaded dumps is implemented within the plugins itself. Thus, it is possible to extend the framework to a virtually infinite grade. As this thesis focuses on private information, or more precisely leakage and theft of these data, several plugins dealing with this topic have been implemented. A detailed description of all developed plugins will follow in Chapter 5. Great value has been set upon simplicity, maintainability and intercommunication. In the following, the process of creating plugins will be explained in detail.

4.2.1 Base Plugin

To simplify the development process, a common basis for all plugins had to be established. For this reason, a universal Plugin base has been created. This base serves as a template, and should optimize the effort needed to create a new plugin.

4.2.2 Plugin Registration

Creating and registering a new plugin so that it is available throughout the whole framework is as easy as subclassing the above mentioned "Plugin" class. A special "PluginManager" automatically scans for each subclass of Plugin and registers it so that it is available throughout the whole system. This means that manually adding a plugin, a process which might easily be forgotten during the development process, is not necessary.

To conform to the underlying structure, each plugin needs to conform to a defined protocol.

In this context, the developed framework provides the possibility to pre-filter the network dumps before retrieving them from Elasticsearch (see also 4.1.3). Even though this process is not essential for the analysis process, it can, in some circumstances, improve the overall performance of the application. For example: Imagine the user trying to retrieve a list of all dumps with a certain keyword. Even with automatic analysis this process can be a very time consuming task, especially if the database has grown to a certain extent. Using pre-filtering we can benefit from the performance of Elasticsearch and retrieve only those Network-Dumps which matter without the need of analyzing the whole dump itself.

Furthermore, each plugin is responsible for the visual processing of its analysis results. For this reason, in each analysis step, a list of Network-Dumps is passed in order to search through. The plugin then decides how the data should be visualized by returning the appropriate HTML code.

A protocol was necessary to enable communication between plugins (see 4.2.2). However, due to the dynamic nature of the framework, there is hardly any sufficient way to find a common denominator between all developed plugins so far and those which might be introduced in the future. For this reason, each external plugin, which depends on the results of another plugins, needs to know how to interpret the results.

Intercommunication

When designing a spacious application like the proposed one, special care has to be taken in order to avoid multiple calculation of the same results. If this fact would be ignored, it would not only influence the application's performance but would also result in multiple implementations of the same code. Obviously the so developed code would hardly be maintainable. For example, a plugin which detects private data within a dump might be interested in the connection type. This means, detecting if this data is sent over a secure connection using HTTPS or whether this information might be seen by anyone in the same network. Thus, this plugin would have to implement an algorithm to detect HTTPS traffic as well as logic to detect private data in the dump. Likewise, another plugin could be keen on the percentage of secure versus insecure traffic. It would therefore also have to implement the same logic as before.

Exactly for this reason, the proposed framework provides means to share detection results with all plugins interested in this information.

4.2.3 Result Caching

When performing large bulk operations and analysis procedures, it might often be the case that the result of a calculation or an analysis report might be required multiple times; especially due to the fact that plugins can exchange information between themselves. Particularly CPU- and time-intensive calculations could therefore prolong the overall runtime and have great impact on the performance.

As will be described in Section 4.4.2, the proposed design features an analyze and a compare mode, which allows to further inspect multiple network-dumps at the same time. The benefit of this mechanism, however, would get lost if for each Network-Dump to be analyzed the report would have to be recalculated.

Once uploaded, network-dumps won't change and the analysis report will most certainly be the same for each dump. For this reason, a `ResultHandler` has been implemented in order to temporarily or if necessary permanently

store calculated results. Tests have shown that by using this technique a huge performance-boost can be achieved.

4.3 Network-Dump

As shown above, keeping settling-in-periods as short as possible by relying on and providing interoperability to existing tools, the proposed work does not provide network capture capabilities on its own.

In principle, a network dump may therefore be created by any arbitrary program and imported into the provided solution. However, for this to work a uniform format to be used internally has to be defined first. This guarantees a coherent and especially continuous analysis process, without the need of modifying existing plugins.

Because this work heavily concentrates on smartphone traffic, the developed application only supports network dumps created from Burp-Suite as of this writing. One of the main reasons for this decision is the already implemented support for capturing and deciphering encrypted traffic. Due to the nature of the internal format the existing work can, however, easily be extended to support multiple other file formats.

Altogether, as may be guessed from the context, a Network-Dump represents a collection of server request and response pairs, recorded for a single (smartphone) application to be analyzed. In the developed application the internal format is defined as follows:

- **Program:** The program parameter specifies which external program is used to capture the traffic. This can come in handy when different programs are used to analyze one and the same application.
- **Application Version:** Keeping track of the version number is a fairly important aspect when comparing dumps captured on a different date. This way, it is possible to see traffic changes, if, for example, an application has been updated.

4 Core Architecture

- Platform: Equally important as the application version is the platform on which a network dump was captured. As a result, it is possible to see differences in the network traffic between platforms.
- App Name: The application name parameter is used to differentiate the uploaded dumps and to identify them.
- Exchanges: Exchanges build the heart of a network-dump and contain a series of server-requests as well as their appropriate responses. Each request and response pair itself contains an array of header-properties as well as a body, if applicable.

4.4 User-Interface

Being the last component in the analysis procedure, the user-interface is responsible for displaying the analysis results to the user. This implies, that in general each plugin is only designed to calculate the desired results and may not determine how to display or interpret them.

In fact, there may be an arbitrary number of presentation methods, or in other words templates for one and the same result. For example, when detecting GPS-coordinates within a network dump, it is possible to list them in a table, but at the same display each coordinate on a world map, creating a more versatile method to explore data.

This is achieved by defining a unique identifier for both the plugins and the associated templates. As a result, the Base-Framework can then automatically deduce the dependency, and choose the appropriate display methods for each plugin.

Hence, these mechanisms allow adding or removing new templates for an already existing plugin without making changes to the extension itself.

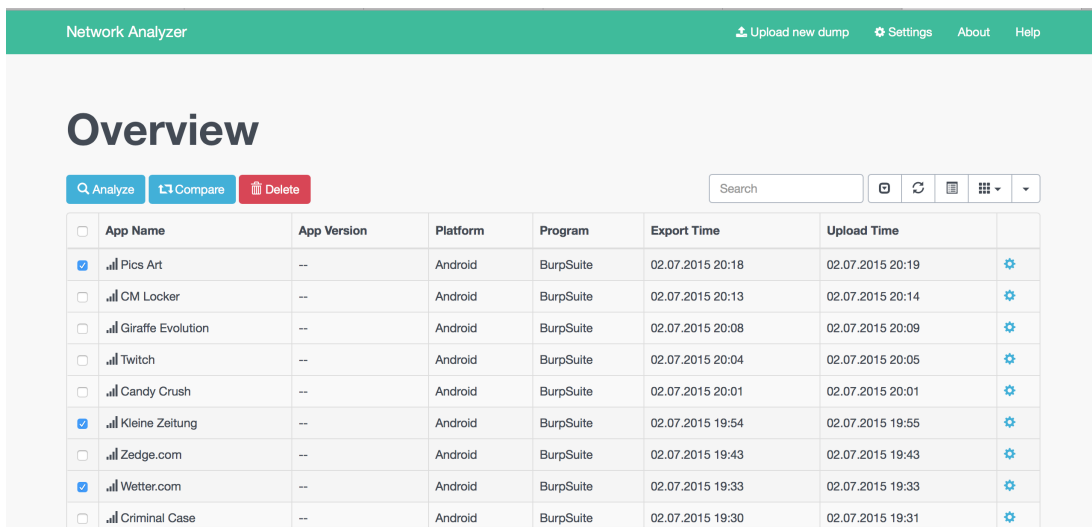
However, additional mechanisms are necessary to display these templates in a meaningful way. For this reason, the user-interface was split into two components: on the one hand, the mentioned templates and on the other so called screens, providing functionality needed for the analysis process itself.

4 Core Architecture

At the same time, they serve as a container for the templates. As a result, by decoupling templates from the remaining user interface essentially no limits are defined for visualizing the analysis results.

4.4.1 Overview screen

The overview screen (see Figure 4.2) gives an outlook on all available applications and their associated network traffic. At the same time it represents the landing page of the developed application. The overview screen offers the ability to upload new network-dumps through a dialog which also allows adding additional fields like application-name, version or category as well as the operating system. These parameters later help to further narrow down the captured traffic. Using the features of elastic search it is also possible to search in real time for data within all uploaded network traffic on the overview screen, providing an easy way to narrow down applications to be analyzed. Naturally, it also possible to delete uploaded network traffic on this screen. By default the latest network dump will be listed on top.



<input type="checkbox"/>	App Name	App Version	Platform	Program	Export Time	Upload Time	
<input checked="" type="checkbox"/>	Pics Art	--	Android	BurpSuite	02.07.2015 20:18	02.07.2015 20:19	⚙️
<input type="checkbox"/>	CM Locker	--	Android	BurpSuite	02.07.2015 20:13	02.07.2015 20:14	⚙️
<input type="checkbox"/>	Giraffe Evolution	--	Android	BurpSuite	02.07.2015 20:08	02.07.2015 20:09	⚙️
<input type="checkbox"/>	Twitch	--	Android	BurpSuite	02.07.2015 20:04	02.07.2015 20:05	⚙️
<input type="checkbox"/>	Candy Crush	--	Android	BurpSuite	02.07.2015 20:01	02.07.2015 20:01	⚙️
<input checked="" type="checkbox"/>	Kleine Zeitung	--	Android	BurpSuite	02.07.2015 19:54	02.07.2015 19:55	⚙️
<input type="checkbox"/>	Zedge.com	--	Android	BurpSuite	02.07.2015 19:43	02.07.2015 19:43	⚙️
<input checked="" type="checkbox"/>	Wetter.com	--	Android	BurpSuite	02.07.2015 19:33	02.07.2015 19:33	⚙️
<input type="checkbox"/>	Criminal Case	--	Android	BurpSuite	02.07.2015 19:30	02.07.2015 19:31	⚙️

Figure 4.2: Overview screen, showing all available network-dumps

4.4.2 Analysis screen

The analysis screen (see Figure 4.3) is responsible for displaying the analysis result of an application. On that account, each (enabled) plugin is automatically loaded and the result is visualized using the detected templates. For easier navigation the left sidebar gives an overview of the current section and helps identify the desired output.

Moreover, one key aspect of the developed application is the possibility of analyzing multiple applications in one attempt. Using the analyze-screen and special designed templates it is also possible to aggregate the data of multiple network-dumps into one single report. For instance, the architecture allows to show statistics about the percentage of detected credit cards per dump or simply show all locations detected in one single view.

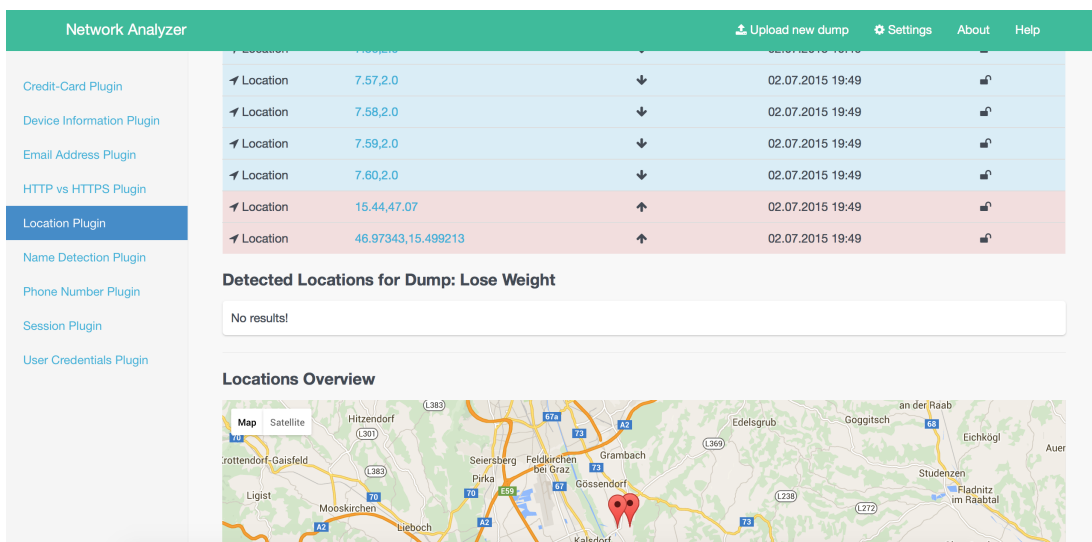


Figure 4.3: Analysis screen, showing a detailed analysis result for one or more network-dump(s)

4 Core Architecture

4.4.3 Comparison screen

In some cases, it might be necessary to see the differences between two network-dumps like for instance a comparison between an iOS and an Android application by the same vendor.

For this reason, the application offers a comparison mode (see Figure 4.4) which displays all the distinctions between two network dumps side by side. In contrast to the analysis screen which serves the purpose of giving a report over multiple network dumps at the same time, the comparison screen allows to easily scroll through the different applications, and find similarities or differences, without the need of changing context.

Additionally, the comparison screen can be used to find divergences in network traffic after application updates have been released. This allows to easily detect features which have been added in a later version of a software.

The screenshot shows the 'Network Analyzer' interface with a green header bar containing 'Upload new dump', 'Settings', 'About', and 'Help'. The main content is divided into three vertical panels, each displaying detected device information and email addresses for a specific application.

Panel 1: Detected Device Information for Dump: Picasso Art

Type	Value	Data Direction	Timestamp	Protocol Type
D	andy-115e5d2e-bf8e-4dfa-b68-ab777f04e167	↑	02.07.2015 20:15	🔒
D	andy-115e5d2e-bf8e-4dfa-b68-ab777f04e167	↑	02.07.2015 20:15	🔒
D	andy-115e5d2e-bf8e-4dfa-b68-ab777f04e167	↑	02.07.2015 20:15	🔒
D	2015-07-02	↑	02.07.2015 20:15	🔒
D	andy-115e5d2e-bf8e-4dfa-b68-ab777f04e167	↑	02.07.2015 20:15	🔒
D	andy-115e5d2e-bf8e-4dfa-b68-ab777f04e167	↑	02.07.2015 20:15	🔒

Panel 2: Detected Device Information for Dump: CM Locker

Type	Value	Data Direction	Timestamp	Protocol Type
Battery	200013.0%	↓	02.07.2015 20:11	🔒

Panel 3: Detected Device Information for Dump: Giraffe Evolution

Type	Value	Data Direction	Timestamp	Protocol Type
ID	e45946228395e4348968673f352286fb	↑	02.07.2015 20:07	🔒
ID	1435667916347-4533488794390473271	↑	02.07.2015 20:07	🔒
Name	4032D	↑	02.07.2015 20:07	🔒
Name	4032D	↑	02.07.2015 20:07	🔒
ID	1039b0498292ea1ae3	↓	02.07.2015 20:07	🔒

Figure 4.4: Comparison screen, showing differences between network-dumps

5 Privacy Leakage Plugins

When dealing with private-information, each user has a different perception of what kind of data is considered private and which one is not [PA02]. As a result, it is not a straightforward task to define "private-data leakage". Additionally, [Pri98] argues that even though some information "ought to be private", this does not necessarily imply that it must not be shared. Or in other words: even though for example a password is considered private information, it is still necessary to transmit it over the Internet, in order to authenticate a user. However, this information must not be revealed to another party.

This brings up another key point: when processing private-data, disclosure of this information is a delicate topic. This "issue of control" [Pri98] implies that as soon as private data is passed from one party to another, who did not have access to this information a priori, it is expected that this information is being dealt with in appropriate ways. The bottom line is that each party involved should ensure that this data is kept safe and all relevant measurements (e.g. encryption) have been taken to prevent undesired exposure and that data is dealt with in respect to the owners intentions.

Drawing parallels to private-data detection, we can clearly see that defining private-data is a topic on its own. For reasons of simplicity we thus define private-data as information which fully or partly could help identify or impersonate an individual and thus draw conclusions on the user's behavior or worse, their whereabouts or its complete natural entity.

Likewise we can already see the limitations of automated network-analysis. While it is possible to correctly detect and identify most outgoing traffic, we still cannot solve the "issue of control".

5 Privacy Leakage Plugins

Nevertheless, automated network-traffic analysis plays a vital role in inspecting applications. As mentioned above, one of the issues when dealing with private data is that the user needs to give his consent to when data is being transmitted. Most of the time, however, it is not clear which kind of information is being sent.

In the context of automated network-traffic analysis, we therefore define private data as data which can help identify an individual. This can be a name, location, password or similar. As a result, the developed plugins for the architecture concentrate on the most widely transmitted information, which could help identify or cause great financial damage to a person. Paired with the information gained from the current malware situation, we have therefore defined the following plugins, being discussed in the following:

- Name Plugin
- Email Plugin
- Username & Password Plugin
- Credit Card Plugin
- Geo Data/Position Plugin
- Phone Number Plugin
- Session/Token Plugin
- Device Information Plugin

5.1 Prerequisites

When trying to find private information within network traffic certain conditions have to be met in the beginning, in order to be able to detect these data.

First of all, it is necessary to consider how this information can be retrieved in an automated way. In general, there exist several solutions for this. However, keeping in mind that essentially all submitted data will eventually have to be processed on a server, the captured traffic almost always follows a certain pattern or a protocol, which can be automatically interpreted. Most

5 Privacy Leakage Plugins

of the time, applications therefore rely on common definitions like JSON¹, Query strings² or the like.

As a result, we can use this circumstance to increase the accuracy and the detection rate of the plugins by using regular expressions³ in conjunction with certain optimizations, discussed in the following.

5.1.1 Private Data Plugin

We have previously defined a base class which serves as a parent to all plugins. When talking about private data this class, however, is too generic and does not provide all functionality which is important for private data detection. For this reason, a more suitable class (`PrivateDataPlugin`) has been defined in order to handle tasks, which could be necessary for every plugin which tries to detect private information. Additionally, this class serves as a common interface to define standards for all remaining plugins.

However, it should be noted at this point, that the Private Data Plugin does not provide analysis functionality on its own. It should rather serve as a tool to reduce redundant or frequently repeating tasks and to keep the implementation effort as low as possible.

5.1.2 Pre-Processing

To keep the false positive rate as low as possible a pre-processing filter is applied on the captured data. This filter is responsible for removing irrelevant parts of the network dump. This includes, but is not limited to files like stylesheets, javascript or binary-data.

Additionally, the filter prepares the text for better analysis. This involves removing quotes around JSON-content or URL-decoding resources.

¹JavaScript Object Notation

²Parameters which are part of the URL. Usually separated by "&"

³Describing sequence of characters using a pattern

5 Privacy Leakage Plugins

Another big problem during development was noticed when analyzing Base64 encoded strings. All too often, they were part of conventional traffic, which resulted in a mixture of Base64 encoded data and alphanumeric content, meaning human-readable text. Unfortunately, there exists no convenient way to detect Base64 encoded strings in a sequence of characters. Rather, with the help of a regular expressions, it is possible to estimate if a string matches the definition. However, this does not necessarily mean that the resulting text really was Base64 encoded. For this reason, the pre-filter applies two functions: first, it detects all possible Base64 parts with a minimal length of 10 characters. This improves the false positive rate. Secondly, the percentage of binary characters is calculated. If the resulting value exceeds the threshold (5%), the detected token probably was not converted using Base64 encoding.

5.1.3 Filtering

Despite the effort, regular expressions are in most cases prone to errors and can often result in too many or too few matches. Additional mechanisms will therefore have to be applied. Generally speaking, there exist two different approaches when deciding to limit the amount of false positives: white- and blacklisting. Still, opinions are sharply divided as soon as the key question is asked: which approach is more convenient? [Tec11]

Usually, a rough differentiation can be made when answering this question. Does an application allow almost everything and only limits a few, or are there only a few cases which are allowed? If the first statement applies, then blacklisting is probably the more convenient approach, if however the latter is more accurate then a whitelisting approach probably is a much better option.

5.2 Name Plugin

The Oxford English Dictionary⁴ defines the term "name" as "a word or set of words by which a person or thing is known, addressed, or referred to" [Oxf]. Although, this statement defines the term to a pretty good extent from a technical point of view, this definition, however, is too generic.

When analyzing applications, we have to find a more restrictive definition in order to find a possible regular expression.

5.2.1 Detection

Generally speaking, finding a name in a network-dump is not an easy task to undertake. Filtering by above mentioned definition causes a high number of false positive matches. Likewise, one could use a dictionary with pre-defined names to match. This, however, would cause a huge increase in the plugin's-size, given the number of different names. Additionally, matching nicknames or typos in names would certainly fail with this approach.

However, relying on the describing nature of the network traffic we can narrow down this process to a certain extent. We can therefore define the following regular expression to mach all names within a given range of characters:

```
((first|last)?|(?<_|[a-z]))name[=:]+[;,\n]
```

This expression matches all parts of strings, which contain the term "name" and optionally start with "first" or "last". To match all characters (including spaces and umlauts) the process is not stopped until one of the defined delimiters is found.

⁴<http://www.oxforddictionaries.com>

5.2.2 Challenges

Without further context, the above mentioned regular expression seems too complex. Defining a more simplified regular expression might deliver the same results:

```
name [=:] [a-z] +
```

The observations during testing however showed that above approach is too general and yields too many false positives. Additionally, it does not match all names. This can easily be explained when looking at the data transferred over the network. Logically, network requests and responses do not only contain well formed protocols like JSON, but can also include HTML content and its corresponding style information, HTTP-Headers, as well as Cyrillic characters or umlauts. Thus, a too simple regular expression would match terms like `<meta name="description" content="text">` or `content_name=sample content` but would not match names like "Müller" or similar.

Still, even with a more sophisticated solution, we need to apply filtering in order to reduce the false positive rate. When we speak of name detection, a whitelisting approach arguably is a fairly good approach to limit the amount of false positives. Thus, only defined sequences are allowed and matched. The drawback is obvious. If the whitelist is too small, some results might not be found.

5.3 Email Plugin

Nowadays, it is hard to think of a modern web without emails. They are virtually everywhere. Be it to send important business information, access data, or simply for fun. Still, revealing one's email address can bring certain disadvantages. Just think about it: on a daily basis more than 300.000 spam emails are sent [Cis15]. This results in roughly 200 billion(!) unwanted mails per year.

When transmitting an email address through the world wide web, the user should therefore be absolutely sure that this process really is essential.

5.3.1 Detection

Compared to name detection, matching emails in a network-dump can be achieved fairly easy. Based on the definition of an email-address [P Ro8] it consists of a local-part and a global part divided by the @ symbol. The global as well as the local-part follow a strict convention. Matching (almost) all e-mail addresses can thus be achieved by applying a simple regex:

```
\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b
```

Above regular expression has been taken from Goyvaerts [Goy15].

5.3.2 Challenges

Despite the fact that there exists a clear defined structure for a valid email-address some problems arose during the development process.

In general, email-addresses will either be submitted in a POST request body or in a GET request url string. In the second case however, the @ symbol has to be url-encoded in order to become a valid identifier. Thus, this process has to be undone once this is analyzed. To avoid unnecessary calculations of this process each time a url-encoded string is analyzed, this feature has already been implemented in the pre-filtering mechanism mentioned before.

Another problem was that the results during testing showed that a high percentage of applications contained image names that were successfully matched as valid email addresses. This can be attributed to the introduced convention of naming those files for Retina-Displays⁵ with the suffix @2x.png (a convention indicating twice the image size). Thus, these matches will be manually filtered in the plugin itself.

⁵High-Resolution Displays usually found in Apple devices

5.4 Username & Password Plugin

Especially when talking about sensitive private data like usernames and passwords, special care has to be taken to ensure that non of this information can be read when transmitted via an insecure network.

A User-Credentials plugin would be a perfect example for the need of (plugin-) intercommunication mentioned in Section 4.2.2. From a user perspective, it might for instance not only be important to see which credentials are sent, but also whether they are sent in a secure way.

5.4.1 Detection: Usernames

We can define a username as a unique identifier for a person. This can for instance be an existing email-address, a chosen nickname or a randomly generated sequence of numbers or letters. The key aspect here to note is, that a username has to be unique within a system, or, in other words: there must not exist a second username in a system, which equals a given one.

This definition is important to find a regular expression which matches usernames. However, similar to the name detection plugin, randomly matching usernames per above mentioned definition probably would not lead to the desired results as the false positive rate would be too high. We therefore rely again on the describing parameters for the server requests and responses. As a result detecting usernames, can be achieved with the following regular expression:

```
(user(_)?|u|g)name[=:] [a-z0-9_-[. ]@]
```

5.4.2 Detection: Passwords

Although, in the last years several alternative methods have been developed as user authentication (e.g. iris-scan, fingerprint, vein detection, cloud-based) [MJ00] a password can be specified as a secret token to authorize an

5 Privacy Leakage Plugins

identity. In the traditional sense, however, a password can be defined as a combination of letters, numbers or special characters.

Again, matching a password by above definition might be too generic and results in too many false matches. Thus, in the majority cases we will not be able to avoid matching passwords by a predefined delimiter. Additionally, most modern architectures require a minimum password length. Hence, the resulting regular expression looks as follows:

```
(^[;\n,.&])((pw)|(passw(or)?d)) [=:] [^,;& ]{4,}
```

5.4.3 Challenges

Similar to the regular expression for name detection above mentioned terms might look like they go far beyond the objective of matching usernames or passwords. The results during testing however showed, that matching user credentials (be it usernames or passwords) cannot be achieved completely in a generic way. On the contrary: finding passwords or usernames in a network-dump with the aid of a regular expression requires a considerable amount of experience. At the same time, the objective is to maximize the detection rate but keep the false positive rate as low as possible.

Additionally, the fact that there exists hardly any common definition for password- or username- delimiters has complicated the work. For instance, found username separators were: `username`, `user_name`, `uname`, `gname`. Similarly, password indicators included, but were not limited to: `pw`, `password`, `pass`, `passwd`. Whats more, finding the last character of a password, can be difficult: In a form-encoded body, the `"&"` symbol usually serves as a separator between different components. Simultaneously, a password can contain this character and still be valid. Thus, a mechanism has to be found to distinguish delimiters from valid password characters.

5.5 Credit Card Plugin

Especially with the adaption to technical progress and the increasing capabilities of smartphones, mobile-payment, that is money-transactions via smartphones [Heno2], has by now become standard practice. In fact, we nowadays use our credit-cards more than ever [Sta14a]. In the meantime, it's particularly the easy access to those cards that makes debit-cards not only our daily companion, but also a part of our ever day life.

For this very reason, it is all the more important that this private information is dealt with in a responsible way. Analyzing network-dumps for credit-card information can therefore be very useful as it might give possible insights into whether the entered information is sent over the network or not. For instance: when using a password-manager, an application which goal it is to securely store entered information, the network-dump should not show signs of credit-card information being sent. In contrast, applications which require payment, e.g. online-shops or software-stores, probably will show signs of the credit-card information in their associated network traffic, in order to function properly.

5.5.1 Detection

Due to the special structure of credit cards, matching this information in a network-dump can be achieved comparatively simple, because commonly speaking, a credit card consists of a 12- to 16- digit number (in practice the trend is more towards the latter one), a checksum as well as a corresponding PIN-code. The first four digits represent the credit-card company associated with the card, the fifth digit the type (e.g. gold, platinum) and the sixth digit can be an indicator whether the card is partner or a business card. The remaining ones give information about the associated account.

Because of the clear structure, and the small amount of credit-card companies, matching credit-cards can thus be performed with the following regular expression [Goy13]:

5 Privacy Leakage Plugins

```
\b(?:\d[ -]*?){12,16}\b
```

The attentive reader might have already noticed that the above mentioned regular expression matches all numbers with 12 to 16 digits. Additionally, these figures may be separated by an arbitrary number of dashes or spaces. This, however, does not necessarily mean that matched numbers actually represent valid credit-cards identifiers. The returned data might only represent a sequence of randomly generated numbers. For this reason, the found (possible) credit-card information still has to be checked for its validity.

This can be achieved with the Luhn, also known as the Mod 10 algorithm [LY11]. As a result, most cards contain a validation digit at the end, in order for the algorithm to function properly. By applying the Luhn validation three steps have to be performed.

1. In order to perform the validation process, first of all, all alternate digits, beginning from the second from right, have to be multiplied by two.
2. In the next step, the sum of the now generated figures is calculated and added to the numbers not yet used.
3. In the last step, the calculated sum is divided by ten. If the remainder is zero (thus modulo operation) the card is valid.

When detecting credit-cards, it might be of interest to not only detect the card number but also the card type. As mentioned before the first four digits of a credit-card number indicate its type. It is thus possible to detect the card types with the following regular expressions taken from Bauman [Bau14]

```
Visa - ^4[0-9]{12}(?:[0-9]{3})?$  
Master Card - ^5[1-5][0-9]{14}$  
American Express - ^3[47][0-9]{13}$  
Diners Club - ^3(?:0[0-5]|[68][0-9])[0-9]{11}$  
Discover - ^6(?:011|5[0-9]{2})[0-9]{12}$  
JCB - ^(?:2131|1800|35\d{3})\d{11}$
```

5.5.2 Challenges

Even though detecting credit-card numbers seems to be a very generic approach, there still exist some pitfalls one might not see right away.

For instance: there is no common definition on how to enter credit card information. As a result, numbers may be split in groups of four (e.g. 1234 5678 ...) or may contain different separators like dashes or spaces. Thus, matching these separators should be taken into consideration.

Additionally, as credit card numbers only consist of digits, a primitive approach was to remove all non-numerical characters from the network-dump. As a result, it was possible to check the resulting figure whether it contained any credit card information. However, we have found that the false positive rate was too high. By removing all non-numerical characters, the resulting network-dump consisted of a combination of all containing numbers. Thus digits from dates, sessions or IP-addresses were combined.

Last but not least, it should not be forgotten that credit-card numbers do not have a defined exact length. As a consequence, it is not possible to build, for example, figures with a character length of 16 and check for their validity.

5.6 Geo Data/Position Plugin

With the introduction of GPS in smartphones, a lot of useful applications have been developed making use of this feature (e.g. navigation, tracking, etc.). Yet, at the same time malicious software takes advantage of this technology by using it to spy on users or similar.

However, leakage of a users location may not always be the fault of vicious software as recent reports show. In 2011, researchers proved that a faulty implementation of the Apple iOS operating system allowed attackers to not only find out the current location of a user but also track his movements [Der11]. By connecting the phone to a computer it was possible to read out a location file which was stored in the backup and contained all recent movements. As a result, a map showing all recent places could be created.

This again shows how network analysis can detect undesired leakage of a users location, because for most applications like navigation this location does not necessarily have to leave the device. Signs of GPS coordinates in the network traffic care therefore give a hint, that the location may be used for other purposes.

5.6.1 Detection

Although there exist a lot of different geographic information systems like UTM, UPS or similar, in the mobile sector the usage of longitude (360° : $180^\circ\text{E} \leftrightarrow 180^\circ\text{W}$) and latitude (180° : $90^\circ\text{N} \leftrightarrow 90^\circ\text{S}$) to define locations on a map is widely spread by now [UNLo9]. Usually, one degree ($^\circ$) can be converted into 60 minutes ($'$), and one minute itself into 60 seconds ($''$). As a result the coordinates of Graz would be represented as $47^\circ 4' 0''\text{N}/15^\circ 27' 0''\text{E}$. For easier transmission over the network, these coordinates are sometimes displayed as decimal numbers by dividing minutes by 60 and dividing the seconds by 3600. Last but not least, these calculated figures are added up. Before mentioned location would therefore be converted to: 47.06666666666667, 15.45. Attention should be paid to the fact that degrees range from north to south or east to west. For this reason, the resulting floating point number ranges from -90.0 to +90.0 (or -180.0 to +180.0 for longitude).

With above definition matching longitude and latitude within a network-dump can thus be achieved with one of the following regular expressions:

```
latitude[=:]( )*(-)?[0-9]+[.][0-9]+
```

```
longitude[=:]( )*(-)?[0-9]+[.][0-9]+
```

```
( )*(-)?[0-9]+[.][0-9]+[,]( )*(-)?[0-9]+[.][0-9]+
```

5.6.2 Challenges

Even though matching latitude and longitude within a network-dump works great with above regular expressions, the matching process is very

5 Privacy Leakage Plugins

limited. For example, coordinates in any other geographic system like UTM cannot be found with above definition.

Additionally, above regular expressions only match floating point numbers. Thus, latitude and longitude coordinates which are transmitted in any other format (e.g. degrees) will be skipped.

Whats more, matching latitude and longitude only works under the assumption that the sample for analysis uses a clear delimiter to describe the values. This assumes that either term (latitude or longitude) is used to describe the variables. Sometimes, however, coordinates are submitted as one parameter, divided by a comma. This simplifies the matching process, because it restricts the characters to be matched.

Last but not least, latitude and longitude should always be considered as a pair. Thus the location plugin matches latitude and longitude, which appear consecutively within a dump, as such. If the coordinates were, for some reason, out of order, the matching process would not work. Results during testing have showed, that not all applications, abide by the standard, that latitude coordinates rank first. The values for latitude and longitude are, in that case, interchanged in the analysis report.

5.7 Phone Number Plugin

Similar to geo-locations, phone-numbers can have a big impact on the privacy of a user. Not only can they give information about your current whereabouts (e.g. country-code, area-code or carrier) but disclosure of a phone number can also lead to an interference with the private life. For instance: it is possible to intercept phone-calls or to find out the users location by triangulation just by knowing his phone-number.

Who now thinks that this is far-fetched could be wrong. We are actually not so very far off this situation considering what was revealed by Edward Snowden [BBC14]. Although, this probably could be considered an extreme case.

5 Privacy Leakage Plugins

In contrast to this, phone-numbers are nowadays increasingly used either to verify the authenticity of a person or as an additional measurement to increase account security, like 2-Factor-Authentication [Dis13]. For this reason, it is important to take measurements to properly protect this information and to only submit phone-numbers via the network if absolutely necessary.

Usage of a phone-number detection plugin can help identify which data really is transmitted over the network and can also give some indication of its transmission security.

5.7.1 Detection

Even though phone numbers primarily consist of digits, they heavily vary from country to country. This is based on the fact that phone numbers are prefixed with an area-code, a carrier-code or an international exit code (usually "00" for European countries).

Moreover, with progress in technology and networks, additional features have been implemented. It is therefore now possible to dial international numbers without the need of a specific international exit code. This can be done by entering a plus sign in front of the number.

Additionally, modern smartphones provide the possibility to format phone-numbers for better readability. This includes setting parentheses around area-codes or separating certain parts with dashes. Even spaces might appear in such telephone numbers.

Keeping this in mind it is necessary to rely on a proper annotation in the captured network-dump to detect phone-number identifiers:

```
((phone_number)|(phone)|(telefon)) [=:]  
([+ 0-9]*(\([0-9]+\))?[0-9- /]{3,})
```

5.7.2 Challenges

Phone-number detection can be achieved fairly straight-forward if before mentioned definition is taken into account. Still, there exist some pitfalls one might stumble over.

First of all, it is necessary to consider the fact that phone-numbers are not formatted in a unique way. Thus, there most certainly exist country specific peculiarities. For instance a phone-number in the US could be formatted as following: (123) 456-7891. The same number in Austria however could be formatted as follows: 123 / 45 67 891.

What's more, there even exist country-wide differences. Numbers which reside in the same country do not need an international exit code. Thus, dialing +43 123 45 67 891 and 0123 45 67 891 from within Austria result in the same number.

Last but not least, it should be considered that phone-numbers require a minimum length. In above example, a phone number has to have a minimum length of three characters. This accounts for the fact that most emergency numbers consist of three digits. The tests showed that in rare cases, the number zero was chosen for not available or omitted phone numbers. By defining a minimum length these numbers could be filtered.

5.8 Session/Token Plugin

A session or a token is a unique identifier to keep track of a server-user-interaction. They are commonly used in stateless protocols like HTTP to uniquely identify a person and keep track of their actions [Bar11]. Usually, after successful authentication, a randomly generated character sequence is generated on the server and submitted to the client. On each following request this token is then sent in order to identify the client. Sessions are a necessity, because other implementations, like IP-addresses are not unique on a user basis.

Transmitting these identifiers over the network is obviously essential in order for the communication to succeed. However, if the implementation is

5 Privacy Leakage Plugins

done wrong, sending a session can be of serious consequences. If the token for some reason can be captured this in return entails that the attacker can hijack a session and thus impersonate the affected individual.

A user-session plugin can help identify whether the data which is submitted, is protected appropriately.

5.8.1 Detection

In general, there are several different ways of implementing session-identifiers:

- **HTTP Header Cookie:** Sessions can be stored in cookies. This way, they will be automatically submitted by the browser on each request. When defined as session, the cookie will be deleted after the browser is closed.
- **Part of URL:** In a GET-Request, the session identifier can also be transmitted via the URI part. This can either be achieved via an additional parameter or as part of the URL itself.
- **Body:** Most certainly it is possible to transmit session information in a HTTP-Body. The format may then be almost arbitrary (e.g. JSON, Form Encoded, ...).

With above conventions in mind, matching sessions is only possible if a clear parameter descriptor is available. Technically, it would be possible to search for transmitted UUIDs, this however results in too many false matches. We can construct the following regular expression:

```
(session|token|sid) [=:] [a-z0-9-]+
```

5.8.2 Challenges

Detecting sessions can be a certain challenge because there is no unique defined standard on how a session should be submitted. Even worse, if the session is part of the URL, automatic detection of sessions is hardly possible, without at the same time drastically increasing the false positive rate.

As a result, it is necessary to rely on a proper annotation of the submitted parameters.

5.9 Device Information Plugin

Sometimes, it is necessary for an application to retrieve device information in order to optimize the user experience. For this reason, on most mobile operating systems there exist several mechanisms in order to receive data about the used phone. On one hand, the operating system provides functionality to read out minor records like screen size or the device manufacturer, on the other hand there exist more invasive techniques like getting the device identifier or the device name. Usually the latter is chosen by the user or at least automatically generated by user input. Thus, in most cases it includes parts of or the complete name of a person.

Although device information might not be seen as private information per se, test data showed that in certain cases the revealed data can in fact allow to draw inferences about the user. A device data detection plugin can thus provide information about the personal records which are being sent.

5.9.1 Detection

To classify which data is seen as device data, it is, first of all important to know which information generally exists. In fact, modern operating systems allow the developers of applications to retrieve all kinds of data which might be necessary to run or to debug an application. In most cases this includes data about the device like screen size, crashes, the operating system (including version) or the remaining memory. However, there is more. Some

5 Privacy Leakage Plugins

operating systems not only allow to get the device identifier (usually a unique ID for a device) but also more specific parameters like device-color, device-name or the device-locale. By combining all these items, it is possible to get a pretty good profile of a person, under certain circumstances.

In fact, results have shown that in most cases this device information is used in so called "tracking" or "advertisement" -frameworks. Typically, the usage of those frameworks allows a developer to compile statistical data (e.g. usage, devices, operating systems) as well as to gain money as most advertisement frameworks pay to embed their ads.

Due to the fact there exist several different parameters, matching device information cannot be achieved with one single regular expression, otherwise the information about the parameter would get lost. However, we have observed, that most of the time the submitted data contains an identifier like "device". Matching this information can thus be achieved with the following expressions:

```
(device(_)?id) [=:] [a-z0-9-]+
```

```
(device(_)?name) [=:] [a-z0-9-]+
```

```
(battery[a-z_.*]*) [=:] [0-9.]+
```

5.9.2 Challenges

As mentioned before, matching device information can, in most cases, be achieved fairly simple as most parameter descriptors contain the term "device". Additionally, most of the time, the device information is used in tracking or advertisement frameworks. Thus, there exists almost always a predefined structure which hardly changes.

However, some frameworks use a self-defined protocol with arbitrary parameter separators. This includes non-ASCII characters and binary data like null terminators, tab stops or the like. As a result, this information was never analyzed, because the Private Data Plugin filters binary content by default.

5 Privacy Leakage Plugins

For this reason, a threshold for the binary filter has been defined. Depending on the underlying data set, this filter has to be changed accordingly in order to analyze this information.

By optimizing the filter and replacing these binary characters with proper delimiters (e.g. "="), a significant detection improvement could be achieved. In addition, since the format of this tracking provider stays the same for all applications which rely on this kind of framework, no additional modifications are necessary for other applications.

5.10 Sample Data

To illustrate the different ways of data representation, this section contains a selected list of example data, found in the captured network traffic of various types of applications. Although this represents only a small excerpt of all available data it demonstrates the prevailing situation of a missing standard for data representation in mobile applications. Additionally, it already gives an outline of the varying parameters used to describe variables.

Json

```
{
  "rt": "1",
  "uid": "testiaik",
  "profile": {
    "fullName": "Testfirst",
    "birthDate": "14.06.1990",
    "uid": "testiaik",
    "email": "iaik.analysis.ios@gmail.com",
    "lang": "en"
  },
  "ts": "1434302468",
  "pass": "testpassword",
  "country_code": "AT",
  "gmt_offset": 120
}
```


6 Evaluation

One of the main goals of the developed application was to create an analysis tool to automatically inspect captured network traffic. As a result, achieving a detection rate of 100 percent was indeed desirable but not first priority. In fact, the application should provide possibilities to gain an overview of the software under investigation without much effort. Therefore, having false positives in the detection report can indeed be unpleasant, but should in general not affect the analysis process in a substantial way.

However, we still need mechanisms to determine the usability of this approach and to verify the applications functionality and its process in practice.

For this reason, we use the system to analyze the private data leakage of popular applications on two major mobile operating systems, in order to detect the performance on real life applications. This gives us the possibility to test the developed application in terms of usability, stability and accuracy. However, due to the nature of network traffic analysis and the before mentioned missing standards in transmission protocols, it is very difficult to test this system in terms of detection rate. In the analysis phase we therefore concentrate only on the found characteristics without validating if the proposed system has found all private data being sent.

Afterwards an evaluation of the generated report will be conducted. This evaluation should investigate whether the pursued approach is in fact practical or whether it is necessary to make changes to the detection mechanisms or the overall approach of network analysis in order to achieve meaningful results.

6.1 Goals

The evaluation of the proposed design comprises of investigating top applications on the Android and iOS mobile operating systems to test the system with real data. At the same time however, using the data we can perform an analysis of the impact of the different operating systems on the developed applications. This can give us a hint on their vulnerability to unintended private data leakage as well as an overview of the prevailing situation.

One of the main characteristics of those operating systems is related with their way of data access. In other words, mobile operating systems rely on different permission systems to handle access to private information. Specifically, the Android platform focuses on an "all or nothing" approach. This implies that on application installation, all necessary permissions (e.g. access to private information, sensors or operating system functionality) have to be granted by the user in advance. If not, the software cannot be downloaded and thus not be executed. However, once an application is successfully installed, there is no need to ask for permission to access any previously approved sensor or functionality. This way, depending on the implementation, the user may not have any idea whether data actually is accessed and whether it is sent over the Internet.

In contrast to this, the iOS platform relies on a more "access-centric" approach. In short, this means that on first access of any sensor (e.g. location, microphone, camera, d) or private information (e.g. contacts, photos, ...) the user has to give his consent to allow the processing of private information. However it is also possible to deny this access. The application still has to work properly afterwards.

With this information in mind, one goal of this evaluation is to see whether the two different approaches have a divergent impact on the data which is sent. Specifically, it is worthwhile to see if the user knows without inspecting the network-traffic if access to sensors is happening, whether this data is sent over to the Internet or not and if it is really vital to do so.

At the same time, this examination allows us to investigate if found differences occur because of carelessness in the development process or on purpose. In the first case we can assume that the permission system does

6 Evaluation

not affect the applications at all. In the second case the permission system may indeed affect the transmitted data in some cases. We will therefore examine identical applications from the same vendor to see if the behavior differs. Particularly this includes divergences in data being sent, e.g. device information or user-owned resources, different implementations (e.g. certificate pinning) or security leaks (e.g. missing encryption).

However, in order to achieve a meaningful evaluation, test data have to be chosen accordingly. For this reason the top 50 free applications on each platform have been downloaded and the network-traffic was captured, if possible, using Burp-Suite. This decision was made because in general the top applications entail a large user basis. As a result, usually a lot of work has been put into the development of those applications, reducing the probability of unintended leakage of private information. In the following, the captured dumps were then analyzed using the proposed framework.

Still, it should be noted at this point that even though the top 50 free applications might differ on each platform it is still a valuable investigation as we are only interested in the overall usage and transmission of private data. Different applications due to diverging operating systems and thus applied permission systems can indeed have great impact on the developed applications and as a result their tendency towards sending private information. However, to have equal prerequisites, usage of sensors or private information on the iOS platform has always been granted, associated with the knowledge that data or certain functionality was accessed.

6.2 Limitations and Assumptions

As previously mentioned, the developed toolset as well as all corresponding plugins serve as a platform to automatically detect (private) information flow within captured network-dumps. However, there exist certain limitations for analyzing network traffic.

One of them is, that the transmitted data can be chosen arbitrarily by the developer. As a result, there is hardly any solution for every application

6 Evaluation

to automatically predict and therefore analyze how the data is composed, without manually inspecting it in the first place.

As a matter of fact, we can therefore not guarantee that every information has been detected by the provided solution. Even though there is no uniquely defined standard for transmitting content, we have observed that most applications rely on a similar naming convention. For example most applications which transmit the first name have used a similar parameter to describe this value.

Additionally, for the analysis process we have always provided the same values for required user information. Hence, if this data has not been detected in the network dump, we have manually inspected the traffic in order to either adapt the plugin to the changed structure or to verify that the information has indeed not left the device.

By combining these two mechanisms we have observed an increase in the detection rate of private information

Another limitation of the presented application is that the proposed design is not suitable for obtaining unauthorized access to data, by for example monitoring network-traffic from uninvolved individuals and using the application to extract private-data. Rather, it should be utilized as a tool to gain insight into technologies used and to reveal which kind of data is being transmitted. In fact, it is the application's goal to simplify the way network analysis is performed and thus it should serve as an additional and important source of information (e.g. for to detecting potential security by data leakage of (smartphone-) applications). It should therefore under no circumstances be misused to attack applications and their users.

Based on this fact, it is no difficulty to trick the detection framework in order to fail to notice sent private data. In general, it is sufficient to define a proprietary protocol, or by encoding the data to be sent in an unconventional way, that the plugins' matching algorithm will most certainly fail. Obviously, it would be possible to adapt the plugins to these mechanisms, but as stated above, this is not the intention of the application.

Below analysis results are thus based on the assumption that the applications which have been analyzed rely on a common standard like JSON, form-

encoded parameters or the like. The collected data depends on a clear structure and a well formed network traffic.

Last but not least, it should be noted that some applications rely on certificate pinning. That is, defining the server certificate identifier within the application. As Burp-Suite relies on injecting its own HTTPS certificate, no secure communication can be established in that case. As a result, (HTTPS) network-traffic from these applications cannot be captured and thus not be analyzed. If an application implemented certificate pinning it was removed from the analysis process. However, the number was low enough to disregard.

6.3 Setup

To receive comparable results, a consistent approach while capturing network-traffic was necessary. Otherwise, the outcome of the analysis would have been subject to constant fluctuations. To avoid this, the following procedures have proven to be effective and have been adhered to:

- **Device:** Each device used in the analysis process was wiped and restored to its factory settings. This made sure that there is no unwanted interference in the network-traffic due to previously installed applications or adjustments.
- **First Start:** If possible, each application under investigation was newly installed. This way, data which might have been sent only once were sure to be captured. Additionally, this strategy avoids (accidentally) skipping a possible registration or login process.
- **Disturbance:** As of this writing, there is no possibility in Burp-Suite to filter for a specific application. As a result, each outgoing network traffic is captured and saved. To avoid this, the following precautions have to be taken:
 - Turn off automatic application updates
 - Turn off automatic system updates

6 Evaluation

- Disable email sync
- Close all running applications
- Disable all automatic system services
- **Accounts:** To avoid falsifying the network-dump, each time an application offered the possibility to register a new account a fictitious user was created. For this purpose, two email addresses were registered in advance:
 - iaik.analysis.ios@gmail.com
 - iaik.analysis.android@gmail.com

For obvious reasons, the first was being used on the iOS platform, the latter on the Android operating systems.

- **Phone-Number:** While most applications did not require a valid phone-number, some applications (like Tinder or Handyparken) need a valid identifier in order to use the application. For privacy reasons a disposable SIM-Card was used in this process. If number validation was not necessary usually a combination of 123 45 67 890 or 0664 12 34 567 has been used.
- **Credit-Card:** As revealing credit-card identifiers can cause great damage, no valid credit-card numbers have been used during the analysis process. Rather, a randomly generated, but still valid (according to the verification process) figure was used, whenever credit-card information was required.
- **Credentials:** Last but not least, whenever an account was created, attention was given to stick to common convention. In general, the username was either `testuser` or `testuser-android` with the password `testpassword`. This made the analysis process much easier as it was possible to see whether this data was being sent and if associated plugins correctly identified and classified the transmitted information.

6.4 Top 50 iOS Applications

As of this writing, the following applications were listed under the top 50 in the Austrian iTunes charts¹. However, the ranking changes almost daily, which makes it virtually impossible to generalize which applications really are downloaded the most and whether this data-set is representative enough. Still, test data have shown that the evaluation results do not deviate considerably when interchanging the top 50 applications with the top 100. Thus, it can be assumed that below listed applications give a good representation of the most commonly downloaded software.

When analyzing the categories (see Figure 6.1), we can see that they are nearly equally distributed with a slight tendency towards games. Although this only represents a snap-shot, previous studies have shown that on iOS, the application categories are approximately equally frequent with a trend toward games [Sta13].

However, this does not reflect buyer habits for paid application. In fact, users on iOS tend to spend four times more money on applications, mostly games, than Android users [Elm14]. This is aggregated, on the one hand by the fact that users of Android originate mostly from countries with lower income, on the other hand by the lack of credit cards or a dismissive attitude towards mobile payment [Hix14].

¹<https://www.apple.com/at/itunes/charts/free-apps/>

6 Evaluation

No.	Application	Category	No.	Application	Category
1	WhatsApp Messenger	Social Networks	26	Triple Jump	Games
2	Facebook Messenger	Social Networks	27	Cow Evolution	Games
3	YouTube	Photo and Video	28	ÖBB Scotty	Travel
4	Facebook	Social Networks	29	Viber	Social Networks
5	PlayTube	Utilities	30	Musify Video Tube	Music
6	Loop Drive: Crash Race	Games	31	Lockscreen-Designer	Lifestyle
7	Instagram	Photo and Video	32	Amazon	Lifestyle
8	Spotify Music	Music	33	H&M	Lifestyle
9	Google Maps	Navigation	34	iTunes U	Education
10	Slayin	Games	35	Musicify	Music
11	Snapchat	Photo and Video	36	Mein iPhone suchen	Utilities
12	Layout from Instagram	Photo and Video	37	Zalando Shopping	Lifestyle
13	Shpock	Lifestyle	38	Dude Perfect 2	Games
14	UnWeatherzentrale Österreich	Weather	39	Google	Utilities
15	Shazam	Music	40	LOVOO	Social Networks
16	Playtube	Entertainment	41	3 Kundenzone	Utilities
17	aa	Games	42	SoundCloud	Music
18	AlphaBetty Saga	Games	43	Dino Empire	Games
19	Light Player PRO	Entertainment	44	Music für iPhone PRO	Music
20	Angry Birds Fight	Games	45	Google Übersetzer	Dictionaries
21	PlayTube	Music	46	QR Code Scanner	Utilities
22	Skype für iPhone	Social Networks	47	Retrica	Photo and Video
23	Jurassic World: Das Spiel	Games	48	Dropbox	Productivity
24	willhaben.at	Lifestyle	49	PicsArt Photo Studio	Photo and Video
25	Runtastic PRO	Health and Fitness	50	Try Harder	Entertainment

Table 6.1: Top 50 iOS Applications (22. June 2015)

6 Evaluation

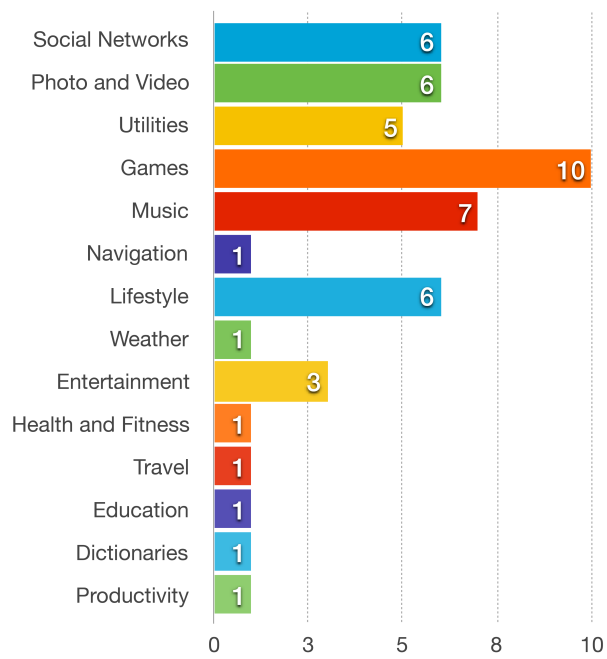


Figure 6.1: Top 50 iOS Application Categories (22. June 2015)

6.5 Top 50 Android Applications

Similar to the top applications on iOS, the app-ranking in the Google Play Store changes almost daily which makes it almost infeasible to make a general statement about app installation times and usage statistics. Naturally, applications can receive updates and changes to their version number. Thus, even if the ranking does not change, this does not imply that the same application versions are still present in the application store after a certain time period. Additionally, because of the missing certification process, applications on the Google Play Store receive updates even faster. For this very reason, the below list has been chosen as a representative snapshot and for better analysis possibilities.

No.	Application	Category	No.	Application	Category
1	WhatsApp Messenger	Communication	26	Clash of Clans	Games
2	Facebook Messenger	Communication	27	LOVOO - People like you	Social Networks
3	Facebook	Social Networks	28	ÖBB Scotty	Traffic
4	Cow Evolution	Games	29	WoodBall	Games
5	Nebulous	Games	30	DU Battery Saver & Akku Doktor	Productivity
6	Clean Master	Games	31	Dude Perfect	Games
7	Google Fotos	Photography	32	COOKING DASH 2016	Games
8	Instagram	Social Networks	33	iTube Pro	Music
9	AlphaBetty Saga	Games	34	Google Übersetzer	Tools
10	Viber	Communication	35	Pou	Games
11	Skype	Communication	36	Retrica	Photography
12	Shpock	Shopping	37	Extreme Car Driving Simulator	Games
13	aa	Games	38	Wish - Freude am Einkaufen	Shopping
14	willhaben.at	Lifestyle	39	Gewicht Abnehmen - Lost Weight	Games
15	Amazon Shopping	Shopping	40	LEGO Ninjago Tournament	Games
16	Ich Einfach Unverbessert	Games	41	Criminal Case	Games
17	Battery Doctor	Tools	42	Weather.com	Weather
18	Snapchat	Social Networks	43	ZEDGES: Ringtones & Wallpapers	Personalization
19	Spotify Music	Music	44	Kleine Zeitung	News & Magazines
20	Kiwi	Social Networks	45	Layout from Instagram	Photography
21	Shazam Music	Music	46	Candy Crush Saga	Games
22	Spider Square	Games	47	Twitch	Games
23	Plastic Surgery Simulator	Games	48	Giraffe Evolution - Clicker	Games
24	Superhelle LED Taschenlampe	Productivity	49	CM Locker (Bildschirmsperre)	Personalization
25	Subway Surfers	Games	50	PicsArt: Foto-Studio	Photography

Table 6.2: Top 50 Android Applications (22. June 2015)

In contrast to the categories listed in Section 6.4, we can see a clear trend towards games in the Google Play Store. Roughly 50% of the downloaded applications are games, whereas approximately 20% of all free applications are intended for entertainment purposes in the Apple App Store. This fact,

6 Evaluation

however, is reversed when looking at paid applications. In the Apple App store, almost 40% of paid applications are games. This statement is also supported by a recent survey [Gau12], stating that more users tend to buy games on iOS than on Android. It furthermore states that app piracy is a big problem on Android due to the different installation and app-certification process.

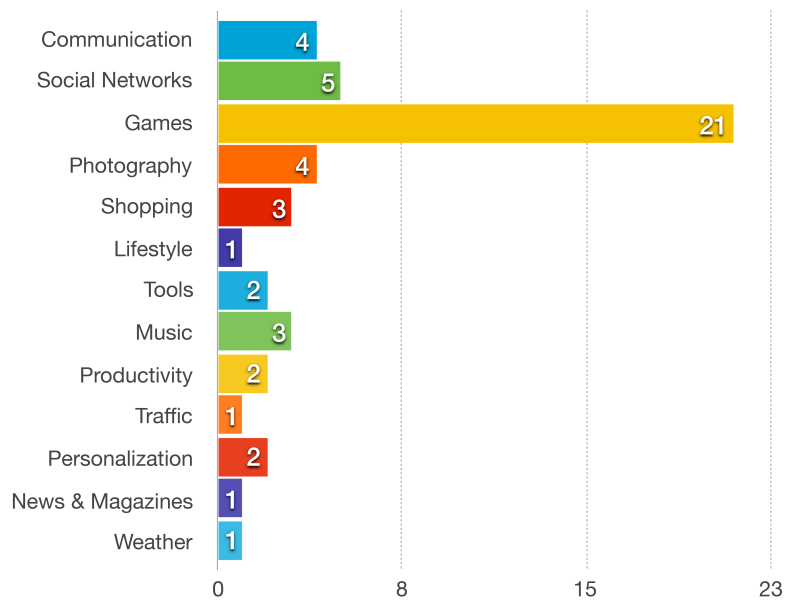


Figure 6.2: Top 50 Android Application Categories (22. June 2015)

6.6 Approach

In general, a well planned approach as well as best-practices are necessary in order to achieve comparable and usable results. For this reason the before mentioned top applications on each platform have been downloaded and the network traffic for each software separately saved and analyzed.

Additionally, the captured network-traffic has been manually filtered to reduce its size, as this thesis focuses only on private data leakage and their possible effects. Specifically binary data, such as images or zip files have been removed a priori, which not only reduces file size, but a positively impacts the analysis processing time. Additionally, special care has been taken to avoid interfering traffic from another applications or the operating system itself.

However, to achieve comparable test results, first of all the different security systems of the operating systems have to be dealt with [Ahm+13]:

As previously discussed, on iOS, each (first) attempt to access a sensor (e.g. location, microphone etc.) needs to be granted by the user. Figure 6.3b illustrates the associated dialog. Naturally, if access has been declined, no information can be retrieved. For this reason, whenever an application has tried to access private data during the evaluation phase, permission has been granted, coupled with the information, that the data might be sent over the network.

In contrast to this, on Android, all permissions which may be required during runtime of the application, have to be granted in advance, meaning before installation (see also figure 6.3a). The user thus might not know, depending on the implementation of the respective application, whether the sensor is accessed or not. Moreover, there currently exist 152 system permissions² to access all kinds of data as well as to modify the operating systems behavior. Hence, this vast amount of permissions might also lead to careless installations of applications from inexperienced users without looking at the required permissions.

²<http://developer.android.com/reference/android/Manifest.permission.html>

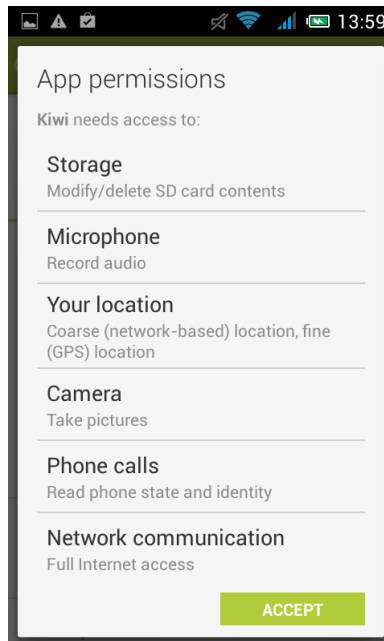
6 Evaluation

In the following, using the obtained data, the developed plugins as well as the proposed framework have been utilized and subjected to an assessment-test. In order to be able to draw conclusions from the analyzed network-dumps, the reports of each plugin have then been documented in a spreadsheet.

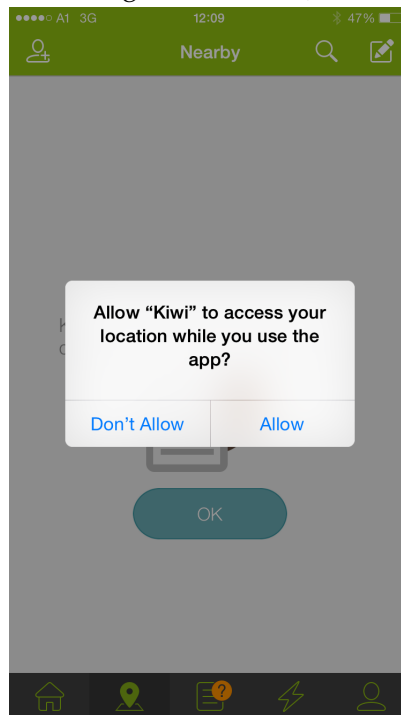
Still, the intended results should not put the different approaches per se to the test. After all, the granted permissions already indicate, that access to the sensor might happen. Hence, the user knows that the application might use this data at some point. Instead the goal of this evaluation is to identify whether user-owned resources (like geo-position or contact addresses) are only used internally (e.g. for navigation or messaging purposes) or if they are sent over the internet. Additionally, this approach helps to identify whether different data is sent more likely if one or the other permission approach is used. Thus, the proposed framework helps to visualize (unwanted) leakage of private data.

Last but not least, it should be noted that, even though the top lists of Android and iOS and their respective application stores differ from each other, it does not influence the outcome of this evaluation. In fact, due to the different nature of the two operating systems under discussion, there exist certain peculiarities on each platform, and thus different capabilities as well as certain applications (e.g. Battery Monitor, Antivirus, Payment - Providers). The point is, that this study, should document the influence of these possibilities for each operating system and possibly give an approximate estimate of the likelihood to private data leakage.

6 Evaluation



(a)Permission Dialog in Android (before installation)



(b)Permission Dialog in iOS (at access)

Figure 6.3: Permission Systems on Android and iOS

6.7 Analysis

In the first step of the analysis phase the network traffic for the top applications for each platform have been inspected using the proposed work and the above mentioned strategy. However, because the developed application can only inspect network-traffic, one might expect that not every application listed requires Internet access and can therefore not be analyzed. Nevertheless, during the implementation and test phase, reports have shown that there exists no application, under investigation which did not require access to the world-wide-web. This includes, on the one hand, software, like social-media or newspapers, which, for obvious reasons, rely on Internet access to function properly. On the other hand however, rather unexpected applications like flashlights or battery-saver applications, which at first glance do not implicitly require network access.

Indeed, the reason for the virtually ubiquitous access can be explained fairly easily. That is, because most of the analyzed applications rely on tracking or advertisement software. This implies, that either content for commercials is being downloaded or logs are sent to dedicated analysis servers. Moreover, mobile analytics software can be used to capture application crashes or to document user behavior like app usage statistics, current time or view changes [Spi+10]. However, most of the time, private data is also associated with these frameworks. As a matter of fact, the developed application can therefore not only help identify whether an application is sending private data but also whether this is due to tracking or advertisements frameworks.

In the next step, the so gained evaluation results of the top applications of each platform have been documented in separate spreadsheets. Tables 6.5 as well as 6.4 provide an overview of these outcomes. In that regard, each row represents the analysis result of one application. Cells marked in red represent a (possible) unwanted data breach or information which has been sent in plain text without proper encryption mechanisms like HTTPS or associated warnings that data access might happen. Similarly, cells in green indicate that the user has been specifically warned that data access is imminent and that it will be transmitted over the Internet. This information has then been used for further analysis.

6 Evaluation

No.	Application	Credit Card	Device ID	Device Name	Battery Status	Email	Location	Name	Phone Number	Carrier	Session	User Credentials
1	Kiwi Q&A	no	no	no	yes	yes	inconclusive	yes	no	no	yes	yes
1	WhatsApp Messenger	no										
2	Facebook Messenger	no	yes	yes	yes	no	yes	yes	yes	no	yes	inconclusive
3	YouTube	no	yes	yes	yes	yes	inconclusive	yes	no	yes	yes	yes
4	Facebook	no	inconclusive	inconclusive	no	inconclusive	inconclusive	yes	inconclusive	no	inconclusive	inconclusive
5	PlayTube	no	no	no	yes	no	no	no	no	yes	yes	no
6	Loop Drive : Crash Race	no	yes	no	no	no	no	yes	no	yes	yes	no
7	Instagram	no	yes	no	no	yes	yes	yes	no	no	yes	yes
8	Spotify Music	no	yes	yes	yes	yes	inconclusive	no	no	no	yes	yes
9	Google Maps	no	yes	yes	yes	yes	inconclusive	no	no	no	yes	yes
10	Slayin	no	no	no	no	no	no	no	no	no	no	no
11	Snapchat	no	no	no	yes	yes	no	no	yes	no	yes	yes
12	Layout from Instagram	no	yes	no	no	no	no	no	no	no	yes	yes
13	Shpock Flohmarkt	no	no	no	yes	yes	yes	yes	yes	no	yes	yes
14	Unwetterzentrale Austria	no	no	no	no	no	yes	yes	no	no	yes	yes
15	Shazam	no	no	no	no	no	no	no	no	no	no	no
16	Playtube	no	no	no	yes	no	no	no	no	yes	yes	no
17	aa	no	no	no	no	no	no	no	no	no	no	no
18	AlphaBetty Saga	no	no	no	no	no	no	no	no	no	no	no
19	Light Player PRO	no	no	no	no	yes	no	no	no	no	yes	yes
20	Angry Birds Fight!	no	no	no	yes	no	inconclusive	no	no	yes	no	no
21	PlayTube	no	no	no	yes	no	no	no	no	yes	yes	no
22	Skype	no	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive
23	Jurassic World: Das Spiel	no	yes	yes	no	no	no	no	no	yes	yes	no
24	willhaben.at	no	no	yes	yes	yes	yes	yes	yes	yes	inconclusive	yes
25	Runtastic PRO	no	no	yes	no	yes	yes	yes	no	no	yes	inconclusive
26	Triple Jump	no	yes	no	no	no	no	no	no	yes	inconclusive	no
27	Cow Evolution	no	yes	no	yes	no	no	no	no	yes	no	no
28	OEBB Scotty	no	no	yes	yes	no	no	no	no	no	yes	no
29	Viber	no	no	no	no	no	no	no	yes	no	no	no
30	Musify Video Tube For YouTube	no	yes	no	yes	no	no	no	no	yes	yes	no
31	Lockscreen-Designer	no	no	no	yes	no	no	no	no	no	yes	no
32	Amazon	yes	no	no	no	yes	no	yes	no	no	yes	yes
33	H&M	yes	no	no	no	yes	no	no	no	no	yes	yes
34	iTunes U	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive
35	Musify	no	yes	no	yes	no	no	no	no	no	yes	no
36	Mein iPhone suchen	no	no	yes	no	yes	no	no	no	no	yes	no
37	Zalando Shopping	no	no	yes	no	yes	no	yes	yes	yes	yes	yes
38	Dude Perfect 2	no	yes	no	no	no	no	no	no	yes	yes	no
39	Google	no	no	yes	no	no	no	no	no	no	yes	no
40	LOVOO	no	no	no	no	yes	yes	yes	no	no	yes	yes
41	3Kundenzone	no	no	no	no	no	no	no	yes	no	yes	yes
42	SoundCloud - Musik & Audio	no	yes	no	no	no	no	no	no	no	yes	yes
43	Dino Empire	no	no	no	no	yes	no	no	no	no	no	yes
44	Music for iPhone PRO	no	yes	no	yes	no	no	no	no	no	yes	no
45	Google Translate	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive
46	QR Code Scanner	no	no	no	no	no	no	no	no	no	no	no
47	Retrica	no	no	no	no	no	no	no	no	no	no	no
48	Dropbox	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive
49	PicsArt Photo Studio	no	no	no	no	yes	yes	yes	no	yes	yes	yes
50	Try Harder	no	no	no	no	no	no	no	no	yes	no	no
		2	15	11	18	17	8	14	7	15	32	19

Figure 6.4: Top 50 iOS Applications (22. June 2015) Analysis Report

6 Evaluation

No.	Application	Credit Card	Device ID	Device Name	Battery Status	Email	Location	Name	Phone Number	Carrier	Session	User Credentials	
1	WhatsApp Messenger	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	
2	Facebook Messenger	no	yes	yes	yes	yes	yes	no	no	yes	yes	yes	
3	Facebook	no	yes	yes	yes	yes	no	no	no	yes	yes	yes	
4	Cow Evolution	no	yes	yes	no	no	no	no	no	yes	yes	no	
5	Nebulous	no	yes	no	no	yes	no	no	no	yes	inconclusive	no	
6	Clean Master	no	yes	yes	yes	yes	inconclusive	no	no	yes	yes	inconclusive	
7	Google Fotos	no	no	no	no	yes	no	yes	no	no	yes	no	
8	Instagram	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	
9	AlphaBetty Saga	no	no	no	no	no	no	no	no	no	no	no	
10	Viber	no	no	yes	no	yes	no	yes	no	no	inconclusive	inconclusive	
11	Skype	no	no	yes	no	yes	no	no	no	no	inconclusive	inconclusive	
12	Shpock	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	
13	aa	no	no	no	no	no	no	no	no	yes	yes	no	
14	willhaben.at	no	no	no	no	yes	yes	no	no	yes	no	yes	
15	Amazon Shopping	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	yes	inconclusive	inconclusive	
16	Ich Einfach Unverbesserlich	no	yes	yes	no	yes	yes	no	no	yes	yes	no	
17	Battery Doctor	no	yes	no	yes	no	no	no	no	yes	yes	no	
18	Snapchat	no	no	no	no	yes	yes	no	yes	no	yes	yes	
19	Spotify Music	no	yes	yes	no	yes	no	no	no	no	no	yes	
20	Kiwi	no	no	no	yes	yes	yes	yes	yes	yes	yes	yes	
21	Shazam – Musik	no	yes	no	no	yes	yes	no	no	yes	yes	no	
22	Spider Square	no	yes	yes	no	yes	yes	no	no	yes	yes	no	
23	Plastic Surgery Simulator	no	yes	yes	no	yes	no	no	no	yes	yes	no	
24	Superhelle LED Taschenlampe	no	yes	no	no	no	no	no	no	yes	yes	no	
25	Subway Surfers	no	yes	yes	no	yes	yes	no	no	yes	yes	no	
26	Clash of Clans	no	no	no	no	no	no	no	no	no	yes	no	
27	LOVOO - People like you	no	yes	yes	no	yes	yes	yes	no	no	yes	yes	
28	ÖBB Scotty	no	no	no	no	yes	inconclusive	no	no	no	no	no	
29	WoodBall	no	yes	no	no	yes	no	no	no	yes	yes	no	
30	DU Battery Saver & Akku	no	yes	no	no	no	no	no	no	yes	yes	no	
31	Dude Perfect	no	yes	no	yes	yes	yes	no	no	yes	yes	no	
32	COOKING DASH 2016	no	yes	yes	no	yes	no	no	no	yes	yes	no	
33	iTube Pro (Decibel Meter?!)	no	no	no	no	no	no	no	no	no	no	no	
34	Google Übersetzer	no	no	no	no	no	no	no	no	no	no	no	
35	Pou	no	yes	no	no	no	yes	no	no	no	yes	no	
36	Retrica	no	no	no	no	no	no	no	no	yes	yes	no	
37	Extreme Car Driving	no	yes	no	no	yes	yes	no	no	yes	yes	no	
38	Wish - Freude am Einkaufen	no	no	yes	no	yes	no	no	no	no	yes	no	
39	Gewicht Abnehmen	no	yes	yes	no	no	no	no	no	no	yes	no	
40	LEGO® Ninjago Tournament												
41	Criminal Case	no	no	yes	no	no	no	no	no	yes	yes	no	
42	wetter.com	no	no	no	no	no	yes	no	no	no	yes	no	
43	ZEDGE™ Ringtones	no	yes	no	yes	no	no	no	no	yes	yes	inconclusive	
44	Kleine Zeitung	no	no	no	no	no	yes	no	no	yes	yes	yes	
45	Layout from Instagram												
46	Candy Crush Saga	no	no	no	no	no	no	no	no	no	yes	no	
47	Twitch	no	yes	no	no	no	no	no	no	yes	yes	yes	
48	Giraffe Evolution	no	yes	yes	no	yes	no	no	no	yes	yes	no	
49	CM Locker	no	no	no	yes	no	yes	no	no	no	no	no	
50	PicsArt	no	yes	no	no	yes	yes	yes	no	yes	yes	yes	
		0	26	18		9	26	17	6	2	29	35	11

Figure 6.5: Top 50 Android Applications (22. June 2015) Analysis Report

6.8 General Comparison: Android and iOS

One important aspect during the analysis phase was to evaluate, whether the two operating systems and their associated permissions systems have different influence on the data being sent. With this in mind, each time private data has been detected with our analysis framework, three factors have been noted:

- **Notice:** Has the user been informed that data access is imminent?
- **Awareness:** Does the user know that data might be transmitted over the Internet?
- **Security:** Have appropriate measures been taken to protect the data (e.g. encryption)?

As a result, we can combine this information with the knowledge that data access has occurred, in order to gain further insight into the application structure. This can give us hints if this information really was required or if the application tries to exploit certain functionality. Additionally, this allows us to draw conclusions whether data access has occurred due to operating system peculiarities or possible carelessness of a user.

6.8.1 Incidents

By looking at the total incidents per plugin, we can see that in general more incidents have been reported for the Android platform. One of the most noticeable differences between those reports however, is that on Android more device related data (e.g. id, name or carrier) has been submitted whereas on iOS more user credentials have been detected in the associated network traffic. Still, in principle this does not allow to draw specific conclusions on the influence of the operating system on the developed applications. In fact the divergence of the data could be explained due to different applications being analyzed. For example, on iOS more applications in the categories

6 Evaluation

“Social Networks” as well as “Utilities” have been analyzed. Likewise on Android more Games have been inspected.

Another important aspect however is, that on Android not only more private data have been found in the network traffic, but also more cases where data have been transmitted over an insecure connection. Additionally, we have documented a higher number of unnoticed usage of sensors. Especially the access of the device-ID or the location sensor has been reported more than twice as often on Android than on iOS. In contrast, this could indeed already be a first indicator that the more the user is aware of data usage the greater is the probability that he questions the necessity of this access. Hence, on Android, developers might exploit this situation more easily.

6.8.2 Categories

In the next step, each private data category (e.g. credit-card, phone number, location, etc.) has been evaluated individually and per platform. Figures 6.6 and 6.7 depict the number of private data being sent, which has been detected with the developed framework and above mentioned plugins. It is worth adding that the figures below and the interpretations are based on the assumption that all sent data have been properly detected as previously discussed. Again, a “no” in the dedicated tables does not necessarily mean that no data have been transmitted but that the current implementation did not detect any possible private data in the network-dump.

When inspecting the individual category results we can immediately see that almost twice as many applications on Android have sent the current user-location than on iOS. This might perhaps not be surprising because it might be due to social networking or location based applications. After all, different software was analyzed for both operating systems. However, looking at the applications and their categories, we can clearly see that almost 35% of those apps can be categorized as games, a phenomenon which could not be observed on iOS at all. No game which was inspected was trying to use the location sensor on the latter platform. This enforces the previous made statement that the different permission systems could have quite an impact on the private data being transmitted. Indeed, one

6 Evaluation

might wonder why a game would need access to the current user location if this warning was displayed during usage.

Location

Speaking of location: in some cases, not only user-names but also the associated locations of other users were found in the network-traffic. Even though this information was not depicted in the app itself by extracting this data from the captured dump, it was possible to display a map of all users nearby who had been using the application and a list of associated usernames or in some cases full names. This process was possible on both operating systems, iOS and Android. This illustrates one of the problems of ubiquitous access to all this information and the careless handling of certain data. In most cases, it was not necessary to send a precise location of a user, but rather a parameter of how far away (e.g. in meters) this person is situated. A potential attacker could use this information to track another user of the same application. In particular, this can be especially dangerous when thinking of dating applications or apps which can be used by minors where revealing an exact location is not always desired. As a result, by sharing this information, it is possible to reveal the whereabouts of a user without him noticing it. One solution to this problem is to calculate the distance to another user not on the device itself, but rather on a centralized server. In this way it is not possible for an attacker to get precise GPS data of other users, while still providing the same user experience. Simultaneously, it shows that even though a user might be warned about data-access, the possible consequences are not always obvious.

Phone Numbers

In contrast to this, on iOS, the phone number is sent approximately twice as many times as on Android. However, there is, as of this writing, no possibility to programmatically retrieve the phone number of a device running iOS. The user thus has to enter it manually. As a matter of fact, in all cases when the phone number was transmitted, it was necessary to enter it by hand. Additionally, virtually every time it served as a security

6 Evaluation

mechanism to verify the user's identity. Still, it is alarming to see that this information has been sent in plain text in some cases, especially, now that techniques such as HTTPS have been widely established on both operating systems.

User Credentials

Another observed difference, is that on iOS more user-credentials, that is either passwords or usernames, have been detected in the associated network-traffic. However, if we once-again look at the category distributions listed in Section 6.1, this anomaly can be explained by the high percentage of apps which offer user registration. In contrast, games usually do not require user-credentials to be played.

Device Information

Last but not least, a very high percentage of applications sends device-related information on both platforms. However, in contrast to other user owned resources, on iOS no special permission is necessary to read out hardware-related information like device-name or the current battery status. Developers can therefore use this data without notifying the user in the first place.

If we now compare these figures with the number of location-based incidents on iOS we can observe twice as many hits. Whereas on Android these figures are approximately the same. Again, based on these observations it could be another indicator, that app developers are more likely to send private data if the user is not asked for permission while using the app.

6.8.3 Results

All in all, based on these facts and the individual results, it can be assumed that on Android, due to the different permission approach and the substantial amount of possibilities, more application developers tend to misuse

6 Evaluation

this system by excessively transmitting user-owned resources. However, this does not imply that the operating system or the permission approach is insecure per-se. Instead, great caution should be taken when installing applications and the required permissions should be inspected carefully.

Additionally, we can confirm our assumption that network-analysis is not only necessary to detect applications with malicious intentions but that it can also help to identify poorly implemented apps which might not be aware of the consequences when transmitting certain information like above mentioned locations.

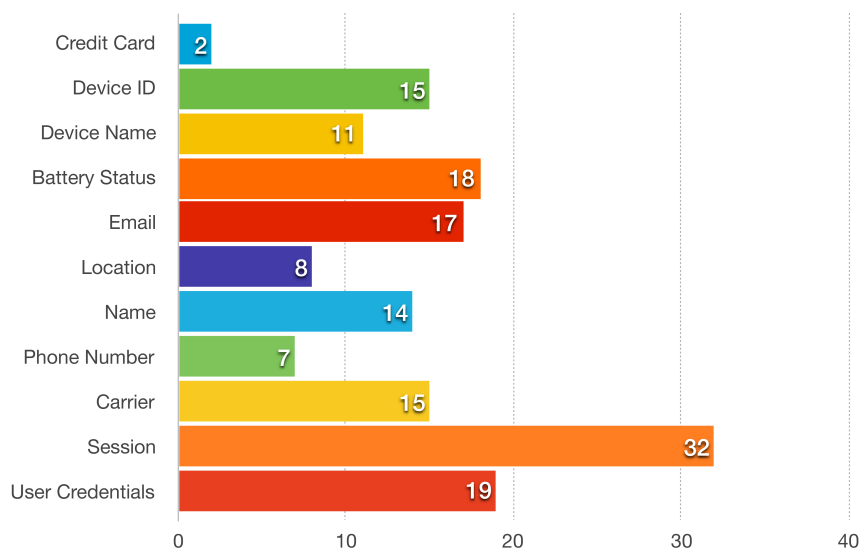


Figure 6.6: Number of detected private data incidents per category on iOS

6 Evaluation

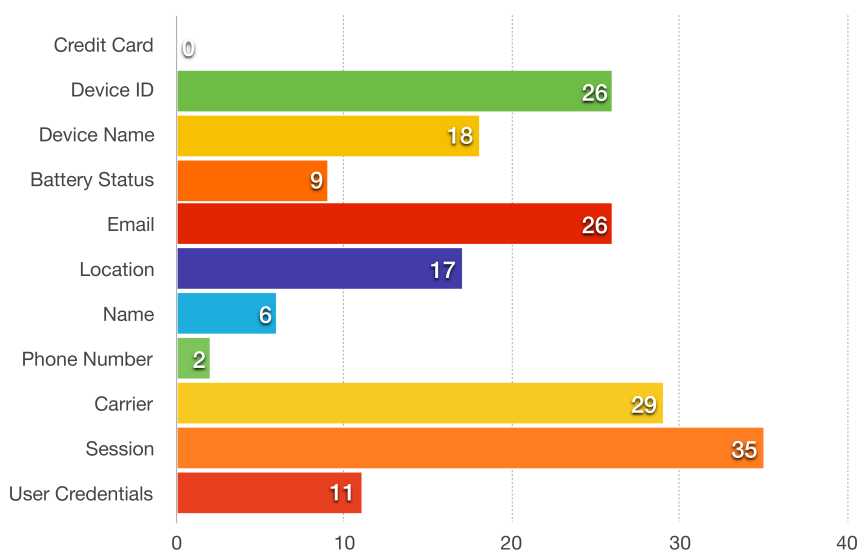


Figure 6.7: Number of detected private data incidents per category on Android

6.9 Comparison: Common Application per OS

Even though the top lists of the two operating systems under discussion differ from each other, there exist common elements between the application-charts depicted in Section 6.1.

These items include the following apps:

- Kiwi
- Whatsapp Messenger
- Facebook
- Spotify Music
- aa
- Sykpe
- willhaben.at
- Cow Evolution
- ÖBB Scotty
- Viber
- Amazon
- Dude Perfect 2
- LOVOO
- Google Translate
- Retrica
- PicsArt Photo Studio

If we now compare these apps in terms of data being sent and their total number of detected private data based, in general no big difference can be detected. On Android, private data has been sent 57 times, whereas on iOS incidents have been reported 52 times. A difference which can, in general, be neglected. This however does not necessarily imply that the implementations of the applications differ from each other. Instead, the slight deviations might be due to different program execution paths, application versions or operating system's dependent capabilities and limitations. After all, the generated network-traffic is influenced by the app usage. To get an overview, Table 6.8 documents this comparison.

However, stepping further into detail, the results in some cases look completely different. For this, the comparison mode of developed application can be used.

Certificate Pinning

For instance: while "Whatsapp Messenger" on both Android and iOS runs on a different port and thus prevents a detailed app analysis, reports for

6 Evaluation

other applications show that different app implementations have been realized. To demonstrate, on Android the Amazon Shopping and Instagram apps do not allow a complete analysis, as certificate pinning prevents content to be received via Burp-Suite. In contrast, on iOS, this limitation is not present and thus allows a complete inspection of all web traffic. Likewise, Facebook on iOS blocks HTTPS traffic routed via Burp-Suite whereas on Android no such mechanism has been detected. Additionally, more games of the observed applications on Android rely on certificate pinning than on iOS.

Tracking Frameworks

However, certificate pinning is not the only difference between those applications. The analysis reports have revealed, that, most likely due to different mobile tracking frameworks, different data is sent for each platform. Specifically this affects device information like battery status or device identifiers.

Another explanation for this phenomenon could be, that there exist more Android devices with changing hardware than Apple devices. As a matter of fact, developers therefore rely on the received device information, in order to optimize their application.

Permission-System

Another strong indicator for the statement, that the different permission approaches can indeed affect the data which is transmitted, can be found when inspecting network-traffic at an app basis. For instance: Whereas "Dude Perfect" on iOS does not require the users location, on Android this information is transmitted over the network.

In general, one would not expect location functionality in this type of game. However, if the user is not notified of the access during gameplay, it is much easier to hide this functionality.

6 Evaluation

No	OS	Application	Credit Card	Device ID	Device Name	Battery Statu	Email	Location	Name	Phone Numt	Carrier	Session	User Credentials	Incidents
1	iOS	Kiwi Q&A	no	no	no	yes	yes	inconclusive	yes	no	no	yes	yes	5
	Android	Kiwi Q&A	no	no	no	yes	yes	yes	yes	yes	yes	yes	yes	7
2	iOS	WhatsApp Messenger	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	0
	Android	WhatsApp Messenger	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	0
3	iOS	Facebook	no	inconclusive	inconclusive	no	inconclusive	inconclusive	yes	inconclusive	no	inconclusive	inconclusive	1
	Android	Facebook	no	yes	yes	yes	yes	no	no	no	yes	yes	yes	7
4	iOS	Instagram	no	yes	no	no	yes	yes	yes	no	no	yes	yes	6
	Android	Instagram	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	0
5	iOS	Spotify Music	no	yes	yes	yes	yes	inconclusive	no	no	no	yes	yes	6
	Android	Spotify Music	no	yes	yes	no	yes	no	no	no	no	no	yes	4
6	iOS	aa	no	no	no	no	no	no	no	no	no	no	no	0
	Android	aa	no	no	no	no	no	no	no	no	yes	yes	no	2
7	iOS	Skype	no	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	0
	Android	Skype	no	no	yes	no	yes	no	no	no	no	inconclusive	inconclusive	2
8	iOS	willhaben.at	no	no	yes	yes	yes	yes	yes	yes	yes	inconclusive	yes	8
	Android	willhaben.at	no	no	no	no	yes	yes	no	no	yes	no	yes	4
9	iOS	Cow Evolution	no	yes	no	yes	no	no	no	no	yes	no	no	3
	Android	Cow Evolution	no	yes	yes	no	no	no	no	no	yes	yes	no	4
10	iOS	OEBB Scotty	no	no	yes	yes	no	no	no	no	no	yes	no	3
	Android	OEBB Scotty	no	no	no	no	yes	inconclusive	no	no	no	no	no	1
11	iOS	Viber	no	no	no	no	no	no	no	yes	no	no	no	1
	Android	Viber	no	no	yes	no	yes	no	yes	no	no	inconclusive	inconclusive	3
12	iOS	Amazon	yes	no	no	no	yes	no	yes	no	no	yes	yes	5
	Android	Amazon	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	yes	inconclusive	inconclusive	1
13	iOS	Dude Perfect 2	no	yes	no	no	no	no	no	no	yes	yes	no	3
	Android	Dude Perfect 2	no	yes	no	yes	yes	yes	no	no	yes	yes	no	6
14	iOS	LOVOO	no	yes	no	no	yes	yes	yes	no	no	yes	yes	5
	Android	LOVOO	no	yes	yes	no	yes	yes	yes	no	no	yes	yes	7
15	iOS	Google Translate	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	inconclusive	0
	Android	Google Übersetzer	no	no	no	no	no	no	no	no	no	no	no	0
16	iOS	Retrica	no	no	no	no	no	no	no	no	no	no	no	0
	Android	Retrica	no	no	no	no	no	no	no	yes	yes	yes	no	2
17	iOS	PicsArt Photo Studio	no	no	no	no	yes	yes	yes	no	yes	yes	yes	6
	Android	PicsArt – Foto-Studio	no	yes	no	no	yes	yes	yes	no	yes	yes	yes	7

Figure 6.8: Common Applications of iOS and Android (22. June 2015)

6.9.1 Results

Concluding, we can see that although one might assume that the same applications show similar behavior, independent from the operating system, the analysis has actually proved the opposite. There might exist several reasons for this.

On the one hand, application-analysis tools for mobile operating systems might require diverging data to be sent. Additionally, due to the large number of different devices, application developers might need to gain more insight into the used hardware, in order to optimize their application.

On the other hand the previously mentioned differences in the permission system and thus user-interface might also play an important role. We have observed, that on one platform applications have been accessing the users' location without explicitly notifying them during usage or explaining the necessity for this action whereas on the other platform the same application was not using the location at all.

Generally speaking, we have shown that the operating system and their associated permission systems might indeed affect the applications and that developers tend to exploit all possible functionality.

6.10 Challenges

Analyzing network-traffic can be a very time-consuming task. Not only is it necessary to prepare the device in order that no interferences from other applications or the operating system are detected in the network-traffic, but also to set up the detection proxy and the deployment of the associated certificate, which all in all can be accompanied by great effort. Additionally, each capture process may last between 5 to 20 minutes, depending on the applications functionality. Analyzing only 100 applications may all too soon last several days.

Furthermore, some applications could not handle the injected certificate from Burp-Suite right away. Sometimes it was necessary to try a couple of times until an app would proceed. Moreover, some applications required a

6 Evaluation

large amount of data to be downloaded in addition to the existing binary (e.g. game-data). In some cases, the proxy would cause a failure of this download, and as a consequence another attempt was necessary to achieve this task.

Even though, that alone is time consuming enough, several other problems have been observed during the implementation and test phase.

In particular certificate pinning used in some applications would make it virtually impossible to capture network traffic. In general, certificate pinning is defined as the process of storing the server-certificate identifier within the application itself. Despite the fact that this approach in general is to prefer from a security point of view as it prevents man in the middle attacks it is simultaneously impractical for network-analysis. These applications could for this reason not be analyzed.

Another limitation is that Burp-Suite can only analyze network-traffic which has been routed via a HTTP-Proxy. This setting however can only be applied on the device itself. As a result, only HTTP Ports, that is Port 80 and 443 are routed via these settings by the operating system. Applications which rely on a different protocol (e.g. FTP (Port 21), or XMPP (e.g. Port 5223)) can thus not be captured with above mentioned approach. Although, the remaining traffic could be captured with other tools like Wireshark, usage of SSL-Certificates and encryption would make it virtually impossible to decrypt this traffic. Additionally, only a small amount of all analyzed applications was using a different port. The high effort to capture this traffic would therefore go beyond the scope of this thesis.

Last but not least, the varying communication standard and the differences in the used protocols made the analysis process a major challenge. In fact, adapting to the different notations without limiting the results or causing too many false positives requires a lot of research in this field.

6.11 Notable examples

Although not necessarily subject of this investigation, this section contains several noticeable examples which have been observed during the analysis

6 Evaluation

process. This includes cases of bad practice, private data leakage due to carelessness as well as critical errors.

At the same time, this shows the effectiveness of network analysis, as we could make these observations in a fast and easy way without inspecting the code of the applications under investigation.

6.11.1 Location Map

We have already mentioned that some applications provide functionality to show the approximate location of users nearby. However, this feature was limited most of the time to show only distances without revealing exact coordinates. This helps to protect the privacy of the users while still providing the same functionality.

Although this approach is favorable in principle, we still observed that in fact all GPS coordinates were transmitted over the network. As a result, distances to nearby users were calculated with these data on the device itself.

As a consequence, we can extract this information from the network-dump and place those coordinates on a map, as Figure 6.9 depicts. Hence, we are able to see exactly the whereabouts of nearby users.

Obviously, this was not the intention of the developers. Using the proposed application during development can therefore reveal such errors, allowing developers to react at an early stage.

6.11.2 Credentials in Network Traffic

Another incident was revealed during the testing phase of the User Credentials-Plugin when the analysis report showed an unknown username and an associated password.

Further inspections then revealed, that the application was in fact connection to another service, using these credentials. As result, we could not only

6 Evaluation

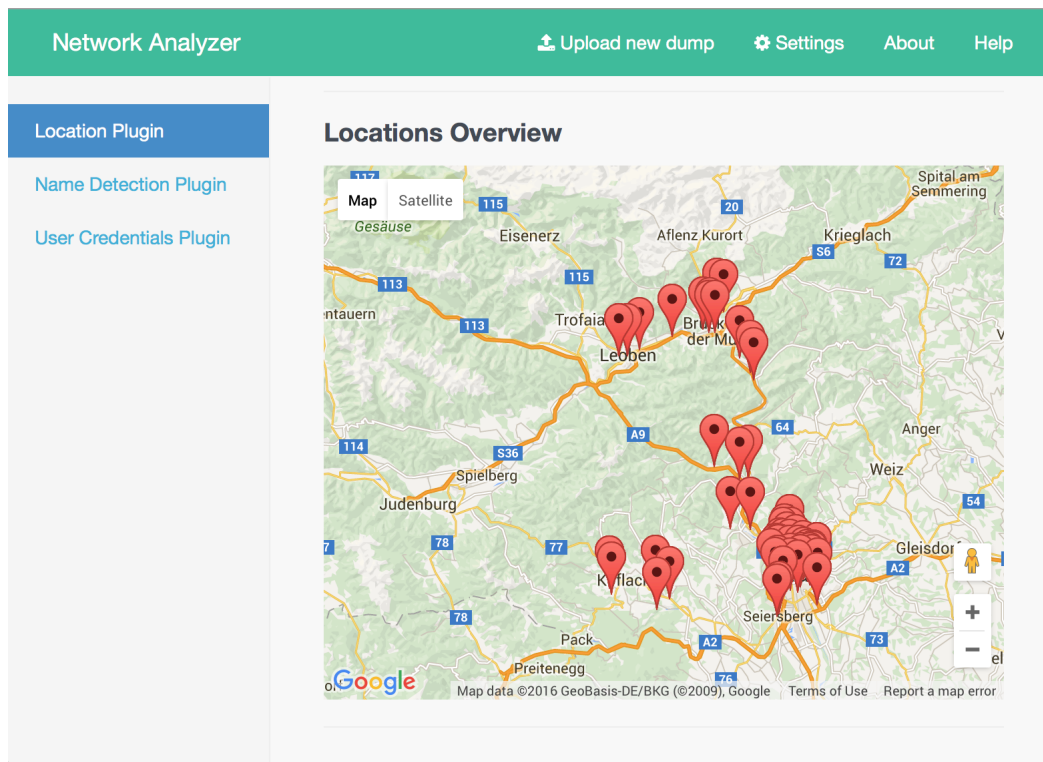


Figure 6.9: Locations of nearby users extracted from the network traffic

identify the service but also log in, using the provided username and password.

Additionally, these credentials have been transmitted over an insecure connection, basically revealing them to anybody in the same network.

6.11.3 SQL Queries

In general, when using databases to store information on a web-server, a special query language can be used in order to retrieve or modify data. However, when exposing this information via a web service it is generally not advisable to provide a direct interface, which means executing the

6 Evaluation

query from a client, but rather only provide means to control the query and therefore modify or retrieve data.

Still, some applications and their associated network traffic revealed that the server accepts complete queries and executes them. Depending on the user rights it might therefore be possible to execute any arbitrary query or even modify the database structure itself. In the worst case the attacker can then elevate its user rights or worse delete all data.

6.11.4 Passwords

When dealing with user credentials, we have observed several errors being made, the most frequent one being password which have been sent without any security mechanisms at all. In general it is not recommendable to submit any kind of user credentials without using proper encryption mechanisms.

However, some applications go a step further by sending a unique identifier derived from a one way function (e.g. hash) from the original password instead of sending the real credentials in the first place. While in general this does not improve the security of the overall system, it increases the security in case the password gets hijacked via a man in the middle attack. This way, it does not compromise any other system if the password has been reused by the user.

Summarizing this shows that network analysis can also be used to verify the security mechanisms of a system and whether they have been deployed correctly.

7 Future Work

Even though automatic network analysis can entail a huge advantage in terms of analysis speed and detection rate, there is still room for improvement:

- **Automatic Capture/Detection:** As of this writing, each application has to be manually analyzed, meaning traffic has to be captured using Burp-Suite (or any other HTTP Proxy) and afterwards dumped to disk. The result then has to be uploaded to the developed application server. As discussed this process can be very time consuming. A more automatic approach (e.g. capturing traffic from within the framework) would be more suitable.
- **Network Traffic Capture:** Likewise, it is time consuming to execute applications by hand. A more automatic approach similar to user-interface tests would be preferable. In this way, bulk analysis of applications in app stores would be more feasible and the results could be reproduced.
- **Machine Learning:** The current implementation of the plugins is based on experience and test results. Changes in protocol (e.g. different delimiters for usernames and passwords) would automatically cause the plugins to either fail or produce wrong results. An additional mechanism to detect false positives or new delimiters could improve the detection rate. However, the current results have showed that most applications rely on a common standard. Excessive changes are thus not to be expected.
- **Distributed Access:** With the usage of elastic-search, a basis for a global deployment has been established. Especially pervasive usage

7 Future Work

would entail a large knowledge base for already analyzed applications. As a result, changes between application versions or even changes between different countries could in that way be easily achieved. This is however based on the prerequisite of global collaboration.

- **Plugin Distribution:** With the current implementation, all plugins have to be deployed in conjunction with the application. However, as mentioned in Section 4.2, it is easy to develop new plugins. Extending this functionality by dynamically deploying new plugins (e.g. via a separate plugin store) could help improve the analysis process.
- **Android 6.0:** We have shown that the permission system of an operating system can indeed influence the capabilities of applications (e.g. location) and the data being sent. However, as of Android 6.0 a new permission system, similar to the one on iOS will be introduced. As a result, this offers a great opportunity to inspect the changes to applications, if any, due to the new permission approach.

8 Conclusion

Altogether, by using the presented work we have shown that the operating system or more precisely the underlying permission system can have huge impacts on the developed applications. As a result, the less user interaction required for certain actions or functionality (e.g. all permissions asked before installation or no permissions at all (e.g. device id, device name)), the more frequent developers tend to access and transmit private data. In this context, it is especially worth noting that on Android a lot of applications were indeed accessing the users location without providing context based experience or explicitly stating the cause.

Comparing the presented work to other techniques likes code analysis or manual traffic inspection, we have observed a fast and easy but especially operating system independent experience, allowing us to inspect software without much effort. All in all, we have shown that an automated approach can indeed speed up the analysis phase in contrast to manual strategies, without losing accuracy. This is on the one hand caused by the fact that most of the time a large amount of data is being transmitted, on the other hand mechanisms such as Base64-Encoding cause additional effort, which can easily be handled within the architecture itself. Additionally, by using a modular architecture, we can enhance the functionality to a virtually arbitrary extent, making the system future prove and even more versatile to use.

However, we have not only shown the possible application of the presented work but have also learned that there was no app listed under the top 50 applications which did not use Internet access, a fact which was not expected prior to this study. As a matter of fact, automated network analysis can be used for most available applications.

8 Conclusion

Moreover, we have experienced that a large number of applications are using a similar naming schema for parameters. This allows us to optimize the application for a great number of apps without knowledge of their structure.

Last but not least, there exist a vast amount of applications which transmit not only less critical information but also more sensitive data like usernames and passwords in plain text, even though secure communication can, by now, easily be achieved on any platform. Informational work therefore has to be done especially in this area.

Summarizing, we can therefore argue that by automatic network analysis, a huge benefit can be derived. However, is important to keep in mind, that this approach can only inspect part of an application. As a result, a lot of functionality can still be hidden within the software under investigation itself. Additionally, the presented approach relies on good structured content. The possible applications for network analysis therefore heavily depend on the use case and desired results.

Appendix

Bibliography

- [Ahm+13] Mohd Shahdi Ahmad et al. "Comparison between android and iOS Operating System in terms of security". In: *2013 8th International Conference on Information Technology in Asia - Smart Devices Trend: Technologising Future Lifestyle, Proceedings of CITA 2013* (2013), pp. 2–5. DOI: 10.1109/CITA.2013.6637558 (cit. on p. 69).
- [Ama14] Ron Amadeo. *The state of Android updates: Who's fast, who's slow, and why*. 2014. URL: <http://arstechnica.com/gadgets/2014/08/the-state-of-android-updates-whos-fast-whos-slow-and-why/3/> (visited on 12/06/2015) (cit. on p. 11).
- [App15] Apple. *App Review*. 2015. URL: <https://developer.apple.com/app-store/review/> (visited on 07/07/2015) (cit. on p. 10).
- [Bar11] A Barth. "{HTTP} State Management Mechanism". In: *Internet Engineering Task Force, University of California 2070-1721* (2011) (cit. on p. 52).
- [Bau14] Gabriel Bauman. *Credit card type and validation*. 2014. URL: <http://stackoverflow.com/questions/20377978/credit-card-type-and-validation> (visited on 06/11/2015) (cit. on p. 47).
- [BBC14] BBC. *Edward Snowden: Leaks that exposed US spy programme*. 2014. URL: <http://www.bbc.com/news/world-us-canada-23123964> (visited on 06/14/2015) (cit. on p. 50).
- [Ben15] Katie Benner. *Apple Confirms Discovery of Malicious Code in Some App Store Products*. Sept. 2015. URL: http://www.nytimes.com/2015/09/21/business/apple-confirms-discovery-of-malicious-code-in-some-app-store-products.html?%7B%5C_%7Dr=0 (cit. on p. 9).

Bibliography

- [Blu14] Blue Coat Systems. "Blue Coat Systems 2014 Mobile Malware Report". In: (2014) (cit. on p. 11).
- [Bod13] A Bodhani. "Bad... in a good way [Information Technology Security]". In: *Engineering & Technology* 8.12 (2013), pp. 64–68. ISSN: 1750-9637 (cit. on p. 6).
- [Cal+09] Arthur Callado et al. "A survey on internet traffic identification". In: *IEEE Communications Surveys and Tutorials* 11.3 (2009), pp. 37–52. ISSN: 1553877X. DOI: 10.1109/SURV.2009.090304 (cit. on pp. 3, 16).
- [Cis14] Inc. Cisco Systems. "Cisco 2014 Annual Security Report". In: (2014) (cit. on p. 11).
- [Cis15] Inc. Cisco Systems. *SpamCop.net - Total Spam Report*. 2015. URL: <https://www.spamcop.net/spamgraph.shtml?spamyear> (visited on 06/17/2015) (cit. on p. 42).
- [Der11] DerStandard.at. *iPhone und iPad speichern Aufenthaltsorte ihrer Nutzer*. 2011. URL: <http://derstandard.at/1303291098336/Ueberwachung-iPhone-und-iPad-speichern-Aufenthaltsorte-ihrer-Nutzer> (visited on 06/12/2015) (cit. on p. 48).
- [Des+08] Lieven Desmet et al. "Security-by-contract on the .NET platform". In: *Information Security Technical Report* 13 (2008), pp. 25–32. ISSN: 13634127. DOI: 10.1016/j.istr.2008.02.001 (cit. on p. 16).
- [Dis13] Stephen T. Dispensa. *Multi factor authentication*. 2013. URL: <https://www.google.com/patents/US8365258> (cit. on p. 51).
- [Elm14] Philip Elmer-DeWitt. *Apple's users spend 4X as much as Google's*. 2014. URL: <http://fortune.com/2014/06/27/apples-users-spend-4x-as-much-as-googles/> (visited on 06/29/2015) (cit. on p. 64).
- [Enc+10] William Enck et al. "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones". In: *OsdI '10* 49 (2010), pp. 1–6. ISSN: 03601315. DOI: 10.1145/2494522 (cit. on p. 15).

Bibliography

- [EOM09] William Enck, Machigar Ongtang, and Patrick McDaniel. "On lightweight mobile phone application certification". In: *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09* (2009), pp. 235–245. ISSN: 15437221. DOI: 10.1145/1653662.1653691. URL: <http://dl.acm.org/citation.cfm?id=1653662.1653691> (cit. on p. 17).
- [Fal+10] Hossein Falaki et al. "A first look at traffic on smartphones". In: *Proceedings of the 10th annual conference on Internet measurement - IMC '10* (2010), p. 281. ISSN: 10636897. DOI: 10.1145/1879141.1879176. URL: <http://portal.acm.org/citation.cfm?doid=1879141.1879176> (cit. on p. 15).
- [For13] Forbes. *Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe*. 2013. URL: <http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/> (cit. on pp. 8–10).
- [Fox15] Thomas Fox-Brewster. *Chinese Cybercriminals Breached Google Play To Infect 'Up To 1 Million' Androids*. Sept. 2015. URL: <http://www.forbes.com/sites/thomasbrewster/2015/09/21/chinese-hackers-beat-google-bouncer/%7B%5C%7D3999864a2e19> (cit. on p. 10).
- [Fse14] F-secure. "Threat Report H1 2014". In: (2014), p. 40 (cit. on p. 12).
- [Gau12] John Gaudiosi. *New Research Shows Apple Still Winning the Video Game War Against Android*. 2012. URL: <http://www.forbes.com/sites/johngaudiosi/2012/05/05/new-research-shows-apple-still-winning-the-video-game-war-against-android/> (visited on 06/25/2015) (cit. on p. 68).
- [Goy13] Jan Goyvaerts. *Finding or Verifying Credit Card Numbers*. 2013. URL: <http://www.regular-expressions.info/creditcard.html> (visited on 06/20/2015) (cit. on p. 46).
- [Goy15] Jan Goyvaerts. *How to Find or Validate an Email Address*. 2015. URL: <http://www.regular-expressions.info/email.html> (visited on 06/07/2015) (cit. on p. 43).

Bibliography

- [Heno2] Joachim Henkel. "Mobile Payment". In: *Mobile Commerce*. Wiesbaden: Gabler Verlag, 2002, pp. 327–351. DOI: 10.1007/978-3-322-90464-5{_}18. URL: http://link.springer.com/10.1007/978-3-322-90464-5%7B%5C_%7D18 (cit. on p. 46).
- [Hix14] Todd Hixon. *What Kind Of Person Prefers An iPhone?* 2014. URL: <http://www.forbes.com/sites/toddhixon/2014/04/10/what-kind-of-person-prefers-an-iphone/%7B%5C#%7D2715e4857a0b648eae233e5a> (visited on 01/13/2016) (cit. on p. 64).
- [Ira+08] D Irani et al. "Evolutionary study of phishing". In: *eCrime Researchers Summit, 2008* (2008), pp. 1–10. DOI: 10.1109/ECRIME.2008.4696967 (cit. on p. 6).
- [Loc12] Hiroshi Lockheimer. *Android and Security*. 2012. URL: <http://googlemobile.blogspot.co.at/2012/02/android-and-security.html> (cit. on p. 10).
- [Loo14] Lookout. *2014 Mobile Threat Report*. Tech. rep. 2014, pp. 1–9 (cit. on p. 6).
- [LY11] Chiyuan Li and Zhiqiang Yao. "The validation of credit card number on wired and wireless internet". In: *Journal of Networks* 6.3 (2011), pp. 432–437. ISSN: 17962056. DOI: 10.4304/jnw.6.3.432-437 (cit. on p. 47).
- [MJ00] Ruud M and Anil Jain. "Biometrics : The Future of Identification It is too early to predict where , how , and in which form reliable biometric". In: *Technology* (2000), pp. 46–49 (cit. on p. 44).
- [Ng+14] Yi Ying Ng et al. "Which Android App Store Can be Trusted in China ?" In: (2014). DOI: 10.1109/COMPSAC.2014.95 (cit. on p. 11).
- [Ong+12] Machigar Ongtang et al. "Semantically rich application-centric security in Android". In: *Security and Communication Networks* 5 (2012), pp. 658–673. ISSN: 19390122. DOI: 10.1002/sec.360 (cit. on p. 16).

Bibliography

- [Oxf] Oxford University Press. *definition of name in English from the Oxford dictionary*. URL: <http://www.oxforddictionaries.com/definition/english/name> (visited on 06/03/2015) (cit. on p. 41).
- [P Ro8] Ed. P. Resnick. "RFC 5322 - Internet Message Format". In: *Network Working Group* (2008) (cit. on p. 43).
- [PA02] D. Primeaux and J.E. Ames. "Personal, private, secret, public [ethics of data privacy]". In: *IEEE 2002 International Symposium on Technology and Society (ISTAS'02). Social Implications of Information and Communication Technology. Proceedings (Cat. No.02CH37293)*. IEEE, 2002, pp. 157–161. ISBN: 0-7803-7284-0. DOI: 10.1109/ISTAS.2002.1013811. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1013811> (cit. on p. 37).
- [Pla13] Chismon Dave Plaskett Alex. *Security Considerations in the Windows Phone 8 Application Environment*. 2013. URL: <https://www.mwrinfosecurity.com/articles/security-considerations-in-the-windows-phone-8-application-environment/> (visited on 01/11/2016) (cit. on p. 10).
- [Pri98] D. Primeaux. "Using an alternative ethical paradigm for analysis". In: *Proceedings of the ethics and social impact component on Shaping policy in the information age - ACM POLICY '98*. New York, New York, USA: ACM Press, 1998, pp. 52–55. ISBN: 1581130384. DOI: 10.1145/276755.276776. URL: <http://portal.acm.org/citation.cfm?doid=276755.276776> (cit. on p. 37).
- [RS10] Alejandro Russo and Andrei Sabelfeld. "Dynamic vs. static flow-sensitive security analysis". In: *Proceedings - IEEE Computer Security Foundations Symposium* (2010), pp. 186–199. ISSN: 19401434. DOI: 10.1109/CSF.2010.20 (cit. on p. 14).
- [Seb14] Sebastian. *ElasticSearch vs. mySQL: Das zweite Rennen*. 2014. URL: <http://www.pal-blog.de/entwicklung/elasticsearch-vs-mysql-das-zweite-rennen.html> (visited on 07/07/2015) (cit. on p. 29).
- [Sec13] G Data Securitylabs. "Mobile Malware Report Half-Year Report July". In: December (2013) (cit. on p. 12).

Bibliography

- [Spi+10] A W Spivey et al. *Embedded mobile analytics in a mobile device*. 2010. URL: <https://www.google.com/patents/US20100041391> (cit. on p. 72).
- [Sta13] Statista. “Most popular Apple App Store categories in January 2013, by share of available apps (in percent)”. In: June (2013), p. 1 (cit. on p. 64).
- [Sta14a] Statista Inc. *Credit cards in the United States - Statista Dossier*. Tech. rep. 2014 (cit. on p. 46).
- [Sta14b] Statista Inc. *Mobile internet usage - Statista Dossier*. New York, New York, USA, 2014 (cit. on p. 5).
- [Sta15a] Statista Inc. *PCs - Statista Dossier*. Tech. rep. New York, New York, USA, 2015 (cit. on pp. 4, 5).
- [Sta15b] Statista Inc. *Smartphones - Statista Dossier*. Tech. rep. New York, New York, USA, 2015 (cit. on p. 4).
- [Sva12] Vanjal Svajcer. *When Malware Goes Mobile: Causes, Outcomes and Cures*. Tech. rep. Boston, USA — Oxford, UK, 2012, pp. 1–5. URL: http://www.sophos.com/en-us/medialibrary/Gated%20Assets/white%20papers/Sophos%7B%5C_%7DMalware%7B%5C_%7DGOes%7B%5C_%7DMobile.pdf (cit. on pp. 8, 9).
- [Sva14] Vanjal Svajcer. *Sophos Mobile Security Threat Report Launched at Mobile World Congress , 2014*. Tech. rep. Oxford, UK: Sophos Ltd., 2014, p. 10 (cit. on p. 1).
- [Tec11] Techtargt Security Media Group. “Information Security”. In: February (2011). ISSN: 13613723. DOI: 10.1016/S1361-3723(02)00104-5 (cit. on p. 40).
- [TKN12] Alok Tongaonkar, Ram Keralapura, and Antonio Nucci. “Challenges in network application identification”. In: *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats* (2012). URL: <https://www.usenix.org/system/files/conference/leet12/leet12-final35.pdf%7B%5C%7Dembedded=true> (cit. on p. 1).
- [UNLo9] UNL Astronomy Education. *Units of Longitude and Latitude*. 2009. URL: http://astro.unl.edu/naap/motion1/tc%7B%5C_%7Dunits.html (visited on 06/12/2015) (cit. on p. 49).

Bibliography

- [Xu+11] Qiang Xu et al. "Identifying diverse usage behaviors of smartphone apps". In: *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference - IMC '11*. New York, New York, USA: ACM Press, 2011, p. 329. ISBN: 9781450310130. DOI: 10.1145/2068816.2068847. URL: <http://dl.acm.org/citation.cfm?id=2068847%20http://dl.acm.org/citation.cfm?doid=2068816.2068847> (cit. on p. 1).