UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School
University of Trento, Italy

Smartphone Data Transfer Protection According to Jurisdiction Regulations

Mojtaba Eskandari

Advisor
Prof. Bruno Crispo, Università degli Studi di Trento, Italy.

Co-Advisor
Dr. Anderson Santana de Oliveira, SAP Labs, Mougins, France.

Examiners
Prof. Francesco Bergadano, Università degli Studi di Torino, Italy.
Prof. Luigi Vincenzo Mancini, Sapienza-Università di Roma, Italy.
Dr. Roberto Carbone, Fondazione Bruno Kessler, Trento, Italy.

Submission: **31st January 2017**, Revision: **24th June 2017**, Defense: **3rd July 2017**

Abstract

The prevalence of mobile devices and their capability to access high speed Internet have transformed them into a portable pocket cloud interface. The sensitivity of a user's personal data demands adequate level of protection in the cloud. In this regard, the European Union Data Protection regulations (*e.g.*, article 25.1) restricts the transfer of European users' personal data to certain locations. The matter of concern, however, is the enforcement of such regulations. Since cloud service provision is independent of physical location and data can travel to various servers, it is a challenging task to determine the location of data and enforce jurisdiction policies.

In this dissertation, first we demonstrate how mobile apps mishandle personal data collection and transfer by analyzing a wide range of popular Android apps in Europe. Then we investigate approaches to monitor and enforce the location restrictions of collected personal data. Since there are multiple entities such as mobile devices, mobile apps, data controllers and cloud providers in the process of collecting and transferring data, we study each one separately. We introduce design and prototyping of a suitable approach to perform or at least facilitate the enforcement procedure with respect to the duty of each entity.

Cloud service providers, provide their infrastructure to data controllers in form of virtual machines or containers; therefore, we design and implemented a tool, named VLOC, to verify the physical location of a virtual machine in cloud. Since VLOC requires the collaboration of the data con-

troller, we design a framework, called DLOC, which enables the end users to determine the location of their data after being transferred to the cloud and probably replicated. DLOC is a distributed framework which does not need the data controller or cloud provider to participate or modify their systems; thus, it is economical to implement and to be used widely.

Keywords
[Personal Data, Data Transfer, Privacy, Mobile Apps, Cloud]

Acknowledgments

All praises to almighty Allah for His unconditional love and countless blessings and throughout my life, in particular, to accomplish this achievement. I would like to thank my beloved family for their support, love, and prayers. My dearest brother-like friend Maqsood Ahmad for his sincere, and unconditional willingness to help me overcome my shortcomings on several occasions.

During the almost four years as a Ph.D. student, I was blessed to meet amazing people who supported me a lot in various ways at the University of Trento and in my staying in SAP Labs France. First and foremost, I would like to thank my supervisors, Professor Bruno Crispo and Doctor Anderson Santana De Oliveira for their excellent guidance and mentor-ship - I love the way they introduced me to research and shaped me as a researcher. I would always be grateful for their availability - they were always available to provide insightful on the research challenges, I faced. I will always be thankful to them for their nice and easy way of expressing complex aspects of my research work and the efforts they made on co-authoring our papers. Second, my colleagues/friends, especially, Waqar Ahmad, Kashif Ahmad, Attaullah Buriro, Nasrullah Khair, Sayed Ali Mirheidari, and Mohammad Lamine who helped and encouraged me in the starting days of my PhD.

I would like to take this opportunity to acknowledge SECENTIS (FP7-PEOPLE-2012-ITN) project which partly supported this PhD by the EU under grant 317387.

I would also like to thank the whole Unitn ICT administrative staff especially Andrea Stenico and Francesca Belton for their support and help in various aspects during my PhD study.

Abbreviations

Back Tracking Pattern	**BT Pattern**
Cloud service provider	**CSP**
Data Protection Authority	**DPA**
Distributed data Location tracker in cloud	**DLoc**
European Union	**EU**
European Economic Area	**EEA**
EU Data Protection Directive	**DPD**
EU General Data Protection Regulation	**GDPR**
Google Play Top Apps Downloader	**GTAD**
Infrastructure as a Service	**IaaS**
Message Authentication Code	**MAC**
Personal Data Transfer Location Analyzer	**PDTLoc**
Platform as a Service	**PaaS**
Proof of Retrievability	**PoR**
Round Trip Time	**RTT**
Service Level Agreement	**SLA**
Software as a Service	**SaaS**
Third Party Auditor	**TPA**
Verifier for physical Location of a virtual machine	**VLOC**

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The popularity of mobile devices has grown drastically in the last decade. International Data Corporation (IDC) reported 1.43 billion smartphone shipments in 2015 and anticipated a steady rise to 1.92 billion in 2020 [77]. In Europe alone, according to the *Ericsson ConsumerLab's* report, there were 475 million user subscriptions in 2014 and this number is estimated to reach 815 million subscriptions in 2020 [31].

Smartphones often store personal data such as contacts, financial information, photos, location, etc. Essentially, "personal data" refers to any information relating to an identified or identifiable natural person, *i.e.*, data subject [76]. It is a common practice for mobile apps to collect, process and transfer personal data to back-end servers (cloud) for further processing and storage. Cloud service provisioning usually is independent of the service provider's location; thus, it raises the issue of identifying in which jurisdiction, personal data is stored and processed. Data protection regulations, such as article 25.1 of the European Union Data Protection Directive (`DPD'25.1`) [76, 44], restrict personal data transfer to particular jurisdictions. DPD'25.1 prohibits the transfer of personal data to any country that does not ensure an adequate level of protection. This principle drove the creation of the *EU-US Safe Harbor* agreement in July 2000 [34].

Years later, an Austrian privacy activist, Maximillian Schrems, sued Facebook Ireland claiming that the company made his personal information available to the US intelligence agencies without any consent or notification [16]. As consequence of that filed case, the European court of Justice decided to invalidate the Safe Harbor agreement [24]. This decision caused numerous debates on the legal aspects of transferring and processing European Citizens' personal data to outside the European Economic Area (EEA) [6, 94].

In February 2016, the European Commission and the United States agreed on a new framework for transatlantic data flows, the *EU-US Privacy Shield* [23]. The new arrangement imposes stronger obligations on companies in the US in order to protect the European users' personal data. Moreover, it provides more robust monitoring and enforcement mechanism that allows the European users to raise any inquiry or complaint in this context with a dedicated new Ombudsman.

However, the major problems are: a) lack of an enforcement mechanism for jurisdiction regulations on both server side and mobile devices; b) there is no analysis system to monitor the data collection and data transfer practices of mobile apps.

1.1 Motivation and Problem Statement

Since data is a passive entity and can be easily replicated and transferred through network, tracking its physical location and enforcing jurisdiction regulations require a robust and widespread monitoring system observing the entire network.

Cloud service providers (CSPs) serve multiple consumers using a multi-tenant model by dynamically assigning resources on consumers' demand [63]. There are multiple roles in a cloud service provision scheme which we use

in this text. *Data Processor* refers to the entity which provides cloud infrastructure. *Data Controller* is an entity (usually an organization) which collects data from individuals (users) and process them in cloud using the infrastructure provided by Data Processor. *Data Subject* indicates an entity/person about whom the data is collected and processed by the Data Controller.

Since the cloud services are consumed over the Internet, they are independent of the location of the provider. CSPs wish to be free to relocate data for various purposes such as load balancing in order to reduce the maintenance cost. However, knowing and controlling the physical location of data for storage and processing is a key requirement for enforcement of the jurisdiction regulations. Moreover, it could be very important for an organization (*i.e.* data controllers) using cloud in some particular scenarios dealing with compliance [3]. Due to lack of appropriate monitoring mechanisms, personal data may be transferred amongst various data centers situated in different jurisdictions, and consequently it might lead to violations in data privacy as there are various data protection regulations in different countries.

This dissertation mainly focuses on providing demonstration of compliance as the violation from jurisdiction regulations significantly impacts on engaged businesses. For instance, the EU General Data Protection Regulation (EU GDPR) imposes considerable fines (up to 4% of the global annual turnover) to organizations that fail to comply with the new framework effective in May 2018 [71]. Two of the most important points of the regulation features are new responsibilities for data processors and new constraints of trans-border data flows, in particular with the emergence of a new EU-US agreement on the topic, the Privacy Shield, replacing the Safe Harbor agreement.

It is necessary to understand how personal data is mishandled, identify

the major barriers and introduce an approach to enforce the regulations on data collection/process. The first step is to analyze the most used mobile apps in order to observe their data collection practices and the ways they handle personal data transfer. The second step is to study the feasible approaches to handle properly the collected personal data according to the data protection regulations. Since there are multiple entities playing role in this procedure, it is crucial to design the observation and the enforcement routines for each entity separately. In other words, how to make the whole set of entities compliant with the data transfer regulations. As cloud service providers, provide their infrastructure to the data controllers in forms of virtual machines or containers, in order to maintain the compliance, data controllers are required to monitor the location of their virtual machines in cloud. The last step is to determine the location of data after being transferred to the cloud from a user's smartphone. This procedure must be independent of the cloud provider or the data controller.

1.2 Research Challenges and Contributions

In the enforcement of jurisdiction policies on mobile devices (*e.g.* Article 25.1 of EU DPD) we need to find the answers of the following research questions:

- **RQ1.** How much and what type of personal data is collected by mobile apps, currently?
- **RQ2.** What are the locations of the remote servers to which the collected personal data is transferred?
- **RQ3.** How many of the popular apps used in the EEA, violate European Data protection regulations on the data transfer restriction?
- **RQ4.** How such regulation can be enforced on users' devices?

- **RQ5.** What is the most effective and practical solution to verify the location of a virtual machine in cloud even if the cloud provider is not collaborative?

- **RQ6.** While maintaining the present-day underlying cloud services and infrastructures, how an end-user can track the location of her data in cloud from her smartphone?

1.2.1 App Analysis for Data Location Investigation (RQ1-3)

Understanding the data collection and data transfer practices of mobile apps requires the analysis of data-flow from where the data is generated to where it is stored in a remote server. We need to infer whether a user's personal data leaves the boundary of the app and to identify the geographical location of the data recipient. Transferring personal data consists of a data flow between the framework APIs called to access personal information and the APIs that provide potential transfer points; such as network, files, log, etc. Code analysis is able to determine some of such data flows; however, due to code obfuscation, reflection, and dynamic code loading, which are widely used in mobile apps, static analysis fails to completely cover the code. Instead, dynamic analysis is able to cover those dynamic parts of the code if a proper triggering mechanism is used.

We design and develop a tool named PDTLoc which inspects mobile applications and extracts information about the collected personal data and the jurisdictions of the remote servers to which the data is transferred. As use case we investigate the current state of the privacy protection of the European smartphone users with regard to DPD'25.1, we used PDTLoc to analyze the $1,498$ most popular apps in the EEA. We obtained evidence confirming that 16.5% (that is 242 apps), transfer data outside the EEA without user consent. This signifies that these apps collect and transfer

personal data to servers located outside the EEA escaping the control of a data protection framework (*e.g.*, Safe Harbor), thus violating the users' data protection rights. We also analyzed the privacy policies provided by the app developers. One striking finding is that 51% of the most used apps in Europe do not provide any privacy policy. Furthermore, out the apps providing a privacy policy, only 53 apps (3.5% of all) had Safe Harbor certification, whose agreement was anyway declared invalid, as we mentioned above. Perhaps the situation will be clarified when the EU-US Privacy Shield framework will be finalized.

This work is published and to be presented in *The 17th International Symposium on Privacy Enhancing Technologies (PETS), 2017, Minneapolis, USA* [32].

1.2.2 Data Location Enforcement on Mobile Devices (RQ4)

Policy enforcement approaches on devices (particularly Android devices) fall into three major categories. The first category consists of modifying the Android framework in order to insert monitoring modules at key interfaces to enable the interception of data collection and data transfer activities as they occur on the device [64, 14, 28, 47, 13, 45, 30]. The problem is that such approaches require extensive modification to the operating system which leads to significant usability issues and widespread adoption. The second category involves decomposing the applications, injecting an inline reference monitor into the code and repackaging it again [92, 53, 25, 52, 10]. The major issue with these approaches is that they are not able to cover the code completely due to dynamic features of the code such as reflection, dynamic code loading, native code, and obfuscated/encrypted code. The third does not need to modify the framework or instrumenting the app code. It hooks the APIs at runtime in order to allow the monitoring of the data collection and data transfer activities of the app.

We design a module on top of *Xposed framework* [91] which intercepts the APIs transferring data to remote servers and analyzes their destination location by using the service provided by PDTLoc. This module then enforces the given jurisdiction policies on the device at the run time.

1.2.3 Verify the Location of a Virtual Machine in Cloud (RQ5)

Once personal data is transferred to a remote server, the user (data subject) has no control over it. Data controllers have to comply with the data protection regulations; however, cloud service providers (CSPs) wish to be free to relocate data among their data centers for multiple purposes including load balancing, energy consumption optimization, reducing maintenance cost, etc. Since CSPs have control over the infrastructure and the network traffic of virtual machines running the data controllers' services, it is quite challenging to verify the location of the virtual machine from inside the cloud. Therefore, there are a number of approaches which perform such verification by employing a widespread network of servers, referred as landmarks [37]. They estimate distance by computing network latency of a challenge message transmitted between each landmark and the server and then determine the location of the server accordingly. There are two major challenges here. First, the network measurements are not reliable due to dynamic nature of the Internet. Second, a huge network of landmarks is quite expensive to implement.

We have overcome these challenges by introducing *VLOC* (a Verifier for physical LOCation of a virtual machine), which is able to verify the physical location of a virtual machine by taking advantage of nearby randomly chosen web-servers. Since VLOC does not rely on a network of fixed landmarks, its implementation is simpler and requires far less maintenance cost than other proposed solutions. VLOC is implemented as a software component which is installed and initialized on a virtual machine.

This work is published in *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014, Singapore* [33].

1.2.4 Distributed Data Tracking in Cloud (RQ6)

Cloud storage services such as Amazon S3, DropBox, or Google Drive allow users to store their data on remote servers independent of geographical location. Cloud storage services utilize a federation schema by maintaining data at different providers which then distribute and replicate the data among different cloud storage providers. This reduces vendor lock-in and increases data availability through additional redundancy which, at the same time, can raise issues with data security and compliance requirements. Particularly, such replication and transparent data distribution limit the user's direct control over data flows, leading to potential violations of data transfer compliance constraints. In such scenario the cloud provider and data controller are not willing to collaborate in compliance with the regulations. Therefore, they do not modify their underlying services in order to enable monitoring the location(s) of data flow.

We already introduced a practical approach when data controller is willing to collaborate by installing VLOC on their virtual machines. We move one step forward and propose a framework, named DLoc, which does not require a network of monitoring servers (dedicated landmarks) and does not need to reside and running within the cloud. DLoc runs in a distributed manner on a significant number of mobile phones, uses a proof of data possession technique to guarantee that the cloud storage service possesses a particular file and estimates the location of all copies of files publicly available in the cloud.

This work is submitted to *International Workshop on Data Privacy Management. Springer International Publishing, 2017, Norway.*

1.3 Organization of the Dissertation

This dissertation consists of the following chapters:

Chapter 2 proposes an analysis mechanism which addresses the issue of transferring privacy related information collected by mobile apps. Moreover, it discusses the current state of privacy protection in mobile apps and suggests a number of actions to improve the control over trans-border personal data flows (RQ1-3). Then it describes the enforcement of location relevant policies for mobile apps' end users. A third party service is under development to analyze mobile apps and notify the user whether the app is compliance with the given jurisdiction privacy policies. Finally we introduce the on-device enforcement mechanism designed for jurisdiction regulations (RQ4).

Chapter 3 continues the previous chapter with focusing on policy enforcement at the service provider's side. Here, we introduce VLOC [33] and its usage for data controllers (RQ5).

Chapter 4 introduces a distributed auditing approach for data tracking in cloud (RQ6). As the presented approach neither needs a modification on the server side or a huge network of landmark servers, it is quite economic and practical to be implemented.

Chapter 5 describes that how the results of this dissertation in demonstrating compliant privacy management practices impacts industry.

Chapter 6 concludes the dissertation by summarizing the chapters presented.

Chapter 2

Analyzing Remote Server Locations for Personal Data Transfers in Mobile Apps

These days, smartphones are homes for a wide range of users' personal data and the apps running on them often use cloud servers for storage and processing. The sensitivity of a user's personal data demands adequate level of protection at the back-end servers. We design and implement an app analysis tool, PDTLoc (Personal Data Transfer Location Analyzer), to detect and study the violation of the jurisdiction regulations.

2.1 Introduction

Several studies highlight that mobile applications actively collect and exfiltrate personal data from smartphones [2, 43, 29]. The main concern here is how to enforce jurisdiction regulations such as DPD'25.1 in mobile apps. As the first pace to tackle the problem, we design and implement *PDTLoc* (Personal Data Transfer Location Analyzer), which employs both static and dynamic analysis techniques to infer whether the apps violate DPD'25.1. PDTLoc inspects mobile applications and extracts in-

formation about the collected personal data and the jurisdictions of the remote servers to which the data is transferred. In order to investigate the current state of the privacy protection of the European smartphone users with regard to DPD'25.1, we used PDTLoc to analyze the 1,498 most popular apps in the EEA. We obtained evidence confirming that 16.5% (that is 242 apps), transfer data outside the EEA without user consent. This signifies that these apps collect and transfer personal data to servers located outside the EEA escaping the control of a data protection framework (*e.g.*, Safe Harbor), thus violating the users' data protection rights. We also analyzed the privacy policies provided by the app developers. One striking finding is that 51% of the most used apps in Europe do not provide any privacy policy. Furthermore, out the apps providing a privacy policy, only 53 apps (3.5% of all) have Safe Harbor certification, whose agreement was anyway declared invalid, as we mentioned above. Perhaps the situation will be clarified when the EU-US Privacy Shield framework will be finalized.

Contributions:

- We design and implement PDTLoc, an Android app analysis tool that employs a backward program slicing technique to detect DPD'25.1's violation by mobile apps [76].

- We collect a dataset of 1,498 Android apps which are the most used apps in the EEA. We analyze these apps using PDTLoc to investigate the recipient server locations of the users' personal data. To our knowledge, this is the first study of the kind conducted so far.

- In order to demonstrate the gravity of the problem, we also analyze the privacy policies of the apps in the dataset in order to check if the data controller and the processing locations are clearly identified.

2.2 Problem Statement

Let $A = \{a_0, a_1, a_2, \ldots, a_n\}$ be a set of android apps. There is a jurisdiction regulation denoted by $R = \{l_0, l_1, l_2, \ldots, l_q\}$, which restricts $a_i \in A$ to transfer data to particular locations *i.e.*, $l_0 ... l_q$. Let $S = \{s_0, s_1, s_2, \ldots, s_m\}$ be the set of all servers used by A to store and process data. Each $s_j \in S$ is situated in a physical location indicated as s_j^l. There is a `T(a,s)` function showing the app a transfers personal data to the remote server s. Formally speaking, we have to enforce the regulation R on the set A by using Equation 2.1:

$$\forall\ a \in A\ \exists\ s \mid \ \ T(a, s) \Rightarrow s^l \in \ R \tag{2.1}$$

The problem we solve is to determine, with a particular level of certainty, whether $a_i \in \ A$ violates R. For each app, we have to discover the list of the remote servers to which it transfers data; then, we need to find their locations and match them against the allowed list, R, to discover violations.

In the analysis we perform in this work, the major goal is to expose the status of privacy protection with respect to DPD'25.1 by the most popular mobile apps in the EU. More specifically, we analyze the type of personal data accessed by these apps, the number of apps that collect/transfer personal data to servers over the network, and the locations of the recipient servers.

Caveat: In this work, we do not try to assert the compliance of the apps but rather to detect if they are likely violating user's privacy rights concerning international data flows limited by the DPD'25.1.

2.3 Data Flow Analysis

The purpose of our data flow analysis is to understand how an app retrieves and transfers personal data. In the context of the problem, we need

to infer whether a user's personal data leaves the boundary of the app and to identify the geographical location of the data recipient. Transferring personal data consists of a data flow between the Android framework APIs called to access personal information; such as device Id, location, contacts, calendar, photos, etc., *i.e.*, "source APIs"; and the APIs that provide potential transfer points; such as network, files, log, etc. *i.e.*, "sink APIs". Source and sink APIs are discussed in detail later in Section 4. An automated data flow analysis tool detects data flows between the source and sink APIs. We can perform such analysis both statically, *i.e.*, extracting information from the bytecode/source code; and dynamically, *i.e.*, running an app on a device/emulator and monitoring its behavior. Here we describe the fundamental concepts regarding static and dynamic analysis that form the basis of our approach.

- **Backward Program Slicing:** In the bytecode representation of a program, two types of data structures are used for storing and performing operations on data, *i.e.*, stack and register. Operations are performed on stack or register variables using program instructions (I). In this text, we use registers (r) as we perform analysis on Android apps and Android is based on a register based virtual machine. Backward program slicing is a data flow analysis technique that, with respect to a register r used at point P in a program, considers all the instructions I that can be executed before P and have a direct or indirect effect on the value of r at P. The combination of r and P, a certain API call in our case, forms a slicing criterion, whereas the set of instruction I that effect the value of r at P is called a backward slice. For instance, Line 2–6 and Line 9 represent a backward slice corresponding to the variable `Sum` used at Line 9 (Point P) in Listing 2.1.

Listing 2.1: Backward Slicing Example

```
1 ...
2 int i, sum, count;
3 i = 1;
4 sum = 0;
5 for( ; i != 50; i++){
6   sum += i;
7     count = count * 100 / i;
8 }
9 System.out.println( "Sum = " + Sum); // point P
```

As a backward slice usually starts from a sink API, an inspection of a backward slice corresponding to a particular register, can provide information about its source and, thereby, infer the data flow path between the source and sink APIs. The source, here, is represented by the API that retrieves the personal information and the sink is represented by the API that transfers the information to the outside world through the Internet.

Such data flow paths can also be inferred by other analysis techniques outlined in the literature (Section 2.10), such as [8]. These techniques, generally, identify access to sources of personal information and then track its flow in the program. Since the basic motivation of this work is to find the location of the recipients of personal information, starting the analysis from the sink APIs yields a better performance by filtering out the apps that do not transfer personal information to the outside world. Therefore, we use backward program slicing and effectively avoid analyzing those apps which might access personal data, but do not send it outside.

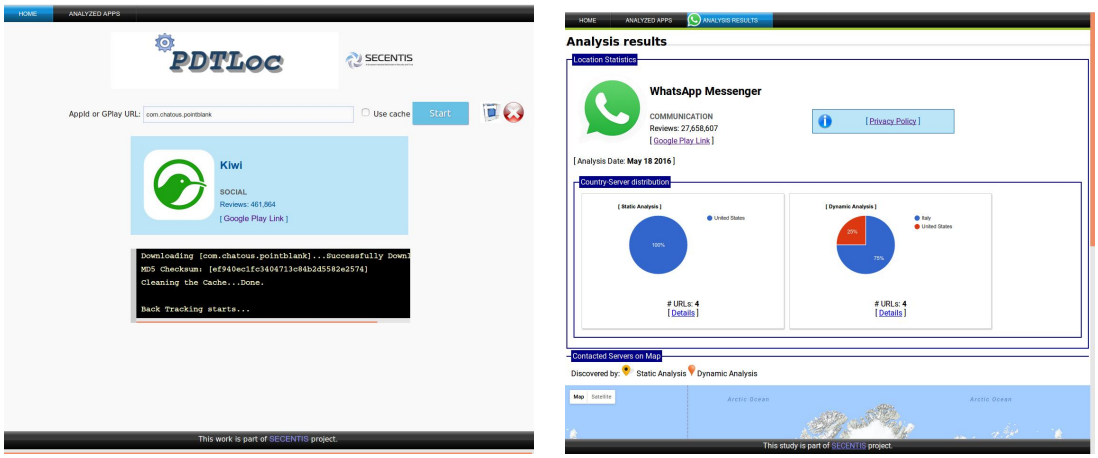
- **Dynamic Tracking:** Dynamic analysis is the process of executing an app and observing its behavior. Tracking the behavior exhibited by an app at execution time, usually, involves monitoring the system resources accessed by the app, such as file-system, network, telephony, etc. Since, this work focuses on personal data transfer over the net-

work, we monitor the outgoing traffic.

Both of these techniques, static and dynamic, come with their respective pros and cons. Static analysis is able to reach all possible data flows in the source code and not only those executed in a specific run of the app. On the other hand, there are programming features, such as various types of code and data obfuscation, reflection, and dynamic code loading, that yields an incomplete result. Reflection is a programming feature that enables apps to operate on strings, *i.e.*, instantiate objects of a class, invoke its methods and access/modify its fields where the class, method and field names are represented by strings that may not be readily available for a static analyzer[78]. Similarly, dynamic code loading allows an app to extend its code base after installation[20]. Therefore, it is impossible for a static analysis tool to fully analyze such cases of dynamic nature. On the other hand, dynamic analysis is able to overcome these limitations in many cases. However, the app must be executed to trigger the critical data flows for dynamic analysis to capture sensitive behavior, which is a challenge for dynamic analysis tools. There are limitations with each technique; however, if combined, static and dynamic analysis techniques can complement each other in designing a more effective data flow analysis system.

2.4 *PDTLoc*

In order to effectively detect violation from DPD'25.1 in popular mobile apps, we design PDTLoc, a tool that takes advantage of both static and dynamic analysis. Two screen shots of PDTLoc's web interface are provided in Figure 2.1 and a complete video demo can be found here: `https://youtu.be/q7fSpq7knV4`.



(a) Analyzing (b) Results

Figure 2.1: Screen shots of DLoc where we can analyze a mobile app and explore the analysis results.

2.4.1 Overview

Figure 2.2 shows an overview of the basic blocks and the workflow of PDTLoc. PDTLoc consists of three major modules: a static analysis, a dynamic analysis, and a location investigator module. Both the static and the dynamic analysis modules take an `.apk` file; extract a list of accessed personal data; and a list of server names, URLs and IP addresses to which the personal information is sent. The dynamic analysis module complements the static module and it is only activated when the given app uses reflection. The lists of URLs extracted by both the modules may wary due to different nature of these analysis techniques. Therefore, we consider a union set of both the lists.

The lists of URLs along with the identifiers of the respective personal information, which is sent to these URLs, are stored in a repository for further analysis. The Location Investigator module (shown in Figure 2.2) reads URL/IP addresses from the repository and creates a list of the server locations where the app sends the collected personal data.

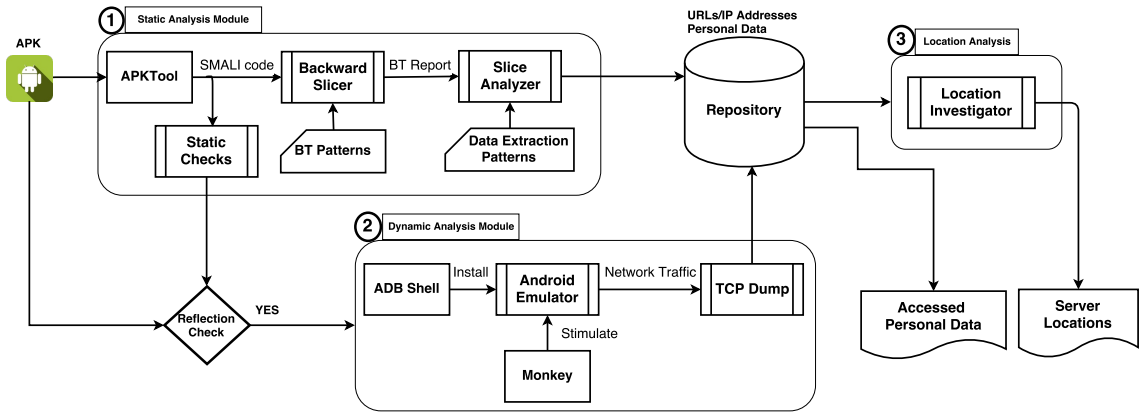


Figure 2.2: A general overview of PDTLoc.

2.4.2 Static Analysis Module

PDTLoc's static analysis module, represented by module (1) in Figure 2.2, takes an `.apk` file and extracts the URLs/IP addresses of the destination servers. APK is an archive file format that represents the Android app and contains all the compiled code and the compiled/raw resources. Android apps are usually written in Java, compiled into Dalvik bytecode and then all the compiled classes are packed into a `classes.dex` file. Therefore, we have to analyze this file to understand the behavior of the app.

To analyze the `classes.dex` file, the static analysis module extracts and translates it into Smali code by employing ApkTool [81]. Smali code is a disassembled representation of the Dalvik bytecode [39]. We use Smali disassembly over Java because the decompilation process is more prone to be thwarted by obfuscation, whereas the disassembly is more resilient [79]. The *Static Check* component inspects the Smali code for the use of reflection in order to pass the app to the dynamic analysis module. The *Backward Slicer* performs backward program slicing on the Smali files to discover the information flow to certain sink APIs. This component employs an extension of SAAF (Static Android Analysis Framework for Android apps) that is able to extract the backward slices corresponding to a given

sink API [46].

Class	Method	Parameter
javax/net/ssl/SSLSocketFactory	createSocket	host, port
android/net/Uri	parse	uri
java/net/URL	<init>	*
java/net/Socket	setRequestProperty	key, value
org/apache/http/client/methods/HttpGet	<init>, setURI	uri
org/apache/http/client/methods/HttpPost	<init>, setURI, setEntity	uri
org/apache/http/client/methods/HttpPut	<init>, setURI	uri
java/io/OutputStream	write	*
java/io/Writer	write	*

Table 2.1: The sinks of personal information. (*: All possible parameters)

Table 2.1 lists the sink APIs along with the corresponding parameters of interest used in our analysis. We carefully analyzed the lists of sink and source APIs provided in the literature, such as [8, 85], and considered only those sink APIs that take the name or IP address of a particular server and transfer data to it. The *Backward Slicer* receives these APIs in the form of an `.xml` file referred to as BackTrack Patterns (*shortly* BT Patterns). A BT Pattern provides information about the API, such as class name, method name, position of the parameter in the parameters list and its type. For example, Listing 2.2 instructs the *Backward Slicer* to backtrack parameter `0`, which is of type `Ljava/lang/String;`, of `setURI` method of class `org/apache/http/client/methods/HttpPost`. Similarly, information about the rest of the APIs in Table 2.1 is also provided to the *Backward Slicer*.

Listing 2.2: BT Pattern Example

```
1 <backtracking-pattern
2   active="true" class="org/apache/http/client/methods/HttpPost"
3     description="Apache HTTP POST" method="setURI" parameters="Ljava/lang/String;"
4     interesting="0" />
```

The *Backward Slicer* spots the position of a given BT pattern in the Smali files, backtracks the target parameter and extracts the corresponding code slice. A code slice contains all the code statements that have a

Listing 2.3: Backward Slice Example

```
1 const-string v0, "facebook.com"
2 sput-object p0, Lcom/facebook/Settings;->facebookDomain:Ljava/lang/String;
3 sput-object v0, Lcom/facebook/Settings;->facebookDomain:Ljava/lang/String;
4 sget-object v0, Lcom/facebook/Settings;->facebookDomain:Ljava/lang/String;
5 const-string v0, "https://graph.%s"
6 ...
7 move-result-object v4
8 invoke-direct {v3, v4}, Ljava/net/URL;-><init>(Ljava/lang/String;)V
```

direct/indirect impact the register holding the value of the target parameter. Listing 2.3 depicts an example of a backward slice corresponding to the API `java/net/URL;-><init>`. Similar slices representing each of the BT patterns are extracted in the form of BT report and provided to a *Slice Analyzer*.

The *Slice Analyzer* component traverses the slices and extracts URLs/IP addresses to which user's personal data might be transferred. Its major role is to analyze the slices for certain data extraction patterns that represent access to personal data. A typical data extraction pattern consists of a class name, a method name, and a parameter as listed in Table 2.2. This list includes only the APIs provided by the framework, used to acquire a user's personal information. However, it does not consider other methods of acquiring user personal data, such as data input through text fields or that stored on files on the device, etc. At this stage, PDTLoc can only guarantee the existence of such data flow paths and flags them suspicious with respect to their potential violation of DPD'25.1. Finally, the *Slice Analyzer* generates a mapping of the personal data accessed by the app and the corresponding server-locations to which the app might transfer the personal data, and stores it in a repository for further processing.

Class	Method	Parameter	Parameter example
android/content/ContentResolver	query	uri	content://media/external/video/media content://sms/inbox content://com.android.browser/history
android/net/Uri	parse	uri	
android/content/Context	getSystemService	name	location connection wifi netstats batterymanager
android/telephony/TelephonyManager	getAllCellInfo getCellLocation getDeviceId getSimCountryIso ...	*	

Table 2.2: The sources of personal information

2.4.3 Dynamic Analysis Module

We dynamically analyze the apps using reflection (around 90% of the apps in this study) that can potentially conceal data flows when only statically analyzed [95]. Therefore, based on the static checks, the PDTLoc's dynamic analysis module, (module ② in Figure 2.2), is activated in case of the app making use of reflection. The dynamic analysis module utilizes a number of tools provided as part of the Android SDK. It executes the given app, monitors its network traffic, and captures the URLs/IP addresses to which the personal data is transferred. It employs `adb` to manage the dynamic analysis process that follows certain steps for each app:

- Launch the emulator and configure WiFi and GPS.
- Run the TCP-Dump tool to monitor the network traffic [80].
- Install the app and unlock the emulator.
- Launch the app with `Monkey` to stimulate it [38]. `Monkey` injects random events into the app including touch, drag, type, change the screen orientation, etc.

- After completion of the execution, URLs/IP addresses and the parameters (*i.e.*, the transmitted data) are extracted from the TCP-Dump, stored in the repository and all the data is erased from the emulator in order to make it ready for next app analysis.

This process is repeated for each app that uses reflection. Since the goal of using the dynamic analysis is to complement the static analysis, the results are stored in the repository as a union set of both the analysis modules. The dynamic analysis module captures all those data flows (including those involving in reflection, native code, dynamic code loading, etc.) that are properly executed during the analysis run. In order to extract the personal information traveling through those data flows, we analyzed the URLs by looking for particular patterns like ‘‘`lat=[\.\-0-9]*`’’, ‘‘`city=[a-Z]*`’’, ‘‘`deviceIds=[0-9]*`’’, ‘‘`macAddress=[0-9a-f]*`’’, etc.

2.4.4 Location Investigator

The fundamental purpose of PDTLoc is to analyze an app and tell if it sends user's personal data to servers hosted at locations outside the jurisdiction defined by the given policies, *e.g.*, the EU DPD'25.1. Therefore, we need to investigate the physical locations of the machines represented by the extracted URLs/IP addresses. The PDTLoc's third module is *Location Investigator* (module ③ in Figure 2.2). This module reads the URLs/IP addresses of the remote servers from the repository and determines their physical locations. There are a number of online databases that bind IP addresses (or server names) to their corresponding geographical locations. We configure the Location Investigator to use `IPaddressAPI.com` [50]. The Location Investigator retrieves the locations using this online IP-Location service and reports a mapping of URLs and geographical locations. It also

marks those locations which are outside the declared jurisdiction.

2.5 Empirical Analysis

This section describes the criteria and the procedure of dataset collection followed by the experimental setup and the evaluation goals.

2.5.1 Dataset Collection

Since this work considers the analysis of the transfer of European users' personal data outside the EEA as a case study (*i.e.*, violating DPD'25.1), we have targeted the popular mobile apps in the EEA. For the app selection, we relied on AppFigures which is an app tracking platform that monitors the downloads and sales of the apps from Google and Apple app stores [7]. We downloaded AppFigures's list of the 400 most popular apps for each EEA state. We identified $1,498$ distinct android apps for the entire EEA and downloaded them. We use android apps because they have over 80% of the market share [49], also for the availability of analysis tools and their simple downloading mechanism. However, we searched for the apps in our list, on iTunes in order to check their availability for iOS. In the dataset we have collected, 80% of apps are available for both Android and iOS and 20% are available only for Android. Therefore, our research results are meaningful to iOS users as well; assuming the destination cloud servers are the same for both OS, which is reasonable.

We developed a fully automated tool, named GTAD (Google Play Top Apps Downloader), which crawls Google play store, identifies the popular apps and downloads them. GTAD can also be configured to download a custom list of apps; therefore, it downloads the apps that we need for this experiment. Moreover, GTAD collects additional information about each app including title, download hits, ratings, category, description, developer

website and email, privacy policy link if available, etc.

2.5.2 GTAD

Figure 2.3 illustrates the four major modules of GTAD. The "Crawler" module annotated by ① queries Google Play store for the list of the top apps. Google by default shows only 540 of top apps[1] while this number is not enough for performing experiments in large scale. We realize that if we query Google Play store for top apps based on category, we can have maximum 600 apps per category and since there are about 50 categories, we can find a significant number of top apps.

The crawler module queries Google Play website and parses the received HTML page. It extracts the apps package names and adds them into the "Apps List". Please note that since package name is unique, we use it as a unique key to store and retrieve data.

The second module, "App Info Extractor" indicated by ②, receives the app package names from the list and queries Google Play website for that app. It extracts the information provided in the app's page including the title and the logo of the app, category and subcategory, number of reviews, user rates/score, number of downloads/installs, developer's organization, website and email, the link to the privacy and policy of the app if there is any, published date, and the description of the app. This module requires a set of regular expressions in order to find and extract the proper information from the given HTML pager. The used regular expressions are listed in Listing 2.4.

The third module is "APK Downloader" which reads the "Apps List" and download them from Google Play. Google Play provides a set of APIs used to authenticate the user and her device and then download the re-

[1] `https://play.google.com/store/apps/collection/topselling_free`

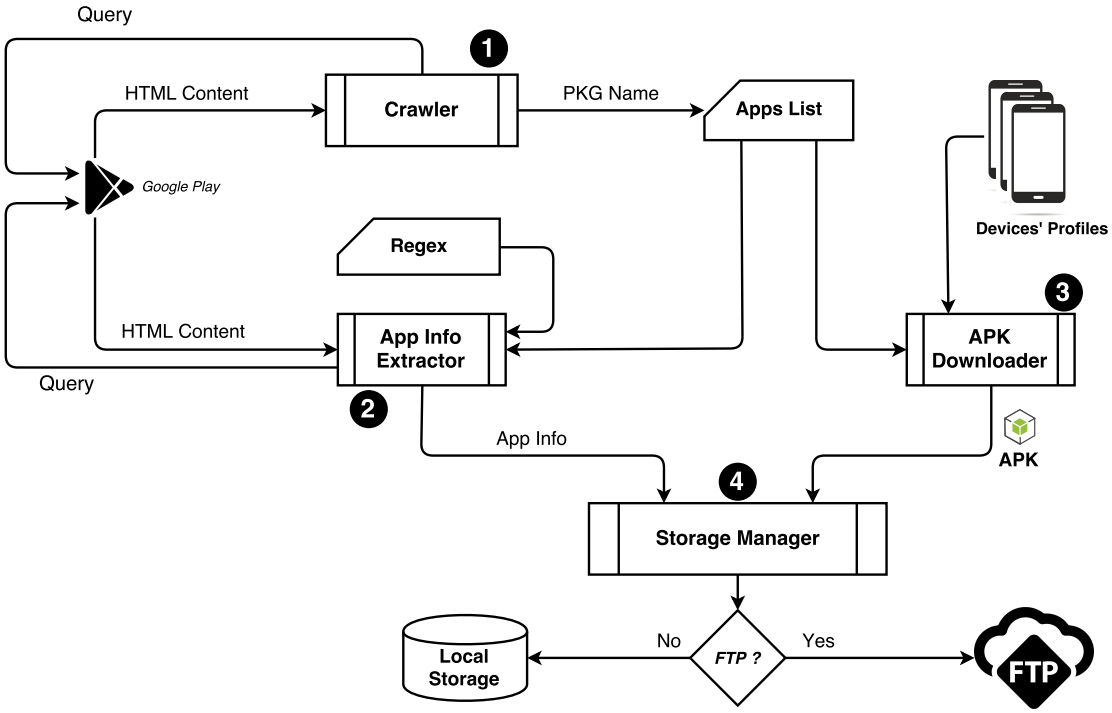


Figure 2.3: A general overview of GTAD.

Listing 2.4: Regular expressions used to extract information about the apps from Google Play website

```
title : r'<div class="id-app-title"[^<]*>([^<]*)<\/div>'
logo  : r'<img class="cover-image"[^>]*src="([^"]*)"[^>]*>'
category: r'<a class="document-subtitle category" href="\/store\/apps\/category\/([^"]*?)
    ">'
subcat  : r'<a class="document-subtitle category"[^>]*>[^<]*<span[^>]*>([^<]+)<\/span>+'
numOfReviews: r'<span class="reviews-num"[^>]*>([0-9\.,\s]*)<\/span>'
score : r'<div class="score"[^>]*>([0-9.,\s]*)<\/div>'
maxDownloads: r'<div class="content" itemprop="numDownloads">[0-9\.,\s]*\-([0-9\.,\s]*)
    <\/div>'
org : r'<div class="title">[\s]*Offered[^<]*<\/div>[^<]*<div class="content">([^<]*)<\/
    div>'
devLink : r'<a class="dev-link" href="[^"]*url\?q=([^"]*)\&amp;sa=D[^"]*"[^>]*>[^<]*<\/a>
    '
devEmail: r'<a class="dev-link" href="mailto:([^"]+)"[^>]*>[^<]*<\/a>'
policyLink: r'<a class="dev-link" href="mailto:[^"]+"[^>]*>[^<]*<\/a>[\s]*<a class="dev-
    link" href="[^"]*url\?q=([^"]*)&amp;sa=D[^"]*"[^>]*>[^<]*<\/a>'
datePublished:  r'<div[^>]*class="content"[^>]*itemprop="datePublished">([0-9a-zA-Z\.,\s
    ]*)<\/div>'
desc  : r'<div[^>]*itemprop="description"[^>]*>(.+?)<\/div>'
```

quested app. We use a python tool, called `gplaycli` [2], to download an APK file from Google Play. Since Google only permits to download the apps compatible with the user's device, GTAD enables us to define as many devices as we need in order to guarantee the APK download. If the APK Downloader module fails to download an app, it automatically tries it with the next device profile until having a successful download. Moreover, GTAD is able to use an FTP server for storing the downloaded information and the APK files.

2.5.3 Experimental Setup

As PDTLoc consists of a static and a dynamic analysis module, we designed the experiment in such a way to know the results from both modules separately as well as their combined results. The static module analyzes all the apps in the dataset, whereas the dynamic module analyzes those apps which pass the static reflection check.

Static analysis module configuration: This module analyzes all $1,498$ downloaded `.apk` files. We used a desktop computer with an `Intel Core i5 3.20 GHz` CPU and 8 GB memory running `Ubuntu 15.10` for the analysis. The static analysis took roughly 38 hours on this machine.

Dynamic analysis module configuration: The dynamic module analyzes those apps that are marked for the use of reflection. We call it Auto-Dynamic as it uses `Monkey` to stimulate the apps. It employs `Android Lollipop 5.0.1` on its emulator and the `Monkey` tool is configured to inject 700 random events into each app. We executed the experiment on the same machine and it took about 6 days to analyze all the given apps.

[2] `https://github.com/matlink/gplaycli`

2.5.4 Evaluation Goals

The experiments are designed to answer the following research questions:

- **RQ1. Accessed Data:** What are the types of personal data accessed/collected by the apps?
- **RQ2. Data Transfer:** What are the locations of the remote servers to which the collected personal data is transferred?
- **RQ3. DPD'25.1 Violation:** How many of the popular apps used in the EEA, violate DPD'25.1? Notice that there may be exceptions where transfers outside of the EEA are authorized, such as Binding Corporate Rules [22]. Data subjects need to be informed about the adoption of such legal mechanisms, through the terms of service and privacy policy. We took this in consideration when considering violations, looking for information about the agreements and certifications by the app developer when available.

2.6 Results and Discussions

We analyzed statically all 1,498 apps and out of these apps, 1,472 (98%) apps use reflection; therefore, we analyzed them also dynamically.

This section reports the analysis results and discusses them in the light of the consequent privacy concerns. The supporting data, the full list of the analyzed apps and the study's conclusions are accessible via this link: `http://titan.disi.unitn.it/pdtloc/`.

2.6.1 Personal Data Accessed

Pieces of personal data stored on a user's device are categorized into three broader groups as shown in Table 2.3. These groups, Content; Device; and

Network, represent user data stored on the device; device status data; and network data, respectively.

Category	Information
Content	Calendar
	Contacts
	Audio
	Video
	Image
	Files
	MMS & SMS
	Call log
	System settings
	User dictionary
Device	Device ID
	Online accounts
	Power state
	System alarm
	Device location
	Telephony services
Network	MAC Address
	Proxy settings
	Network Status
	Network connectivity
	Network usage
	history and statistics

Table 2.3: Types of personal data in each category.

Figure 2.4 provides a graphical representation of the number of apps that access the various types of personal data (**RQ1**). According to these results, device status data, marked as "Device", such as device id, notifications and power information, etc., as shown in Table 2.3, is accessed by almost all apps. Further examination revealed that 75% of the apps request device location. Similarly, network information is of interest to

65% of the apps. What is alarming here is that over 70% of the apps read "Content", which carries sensitive personal information.

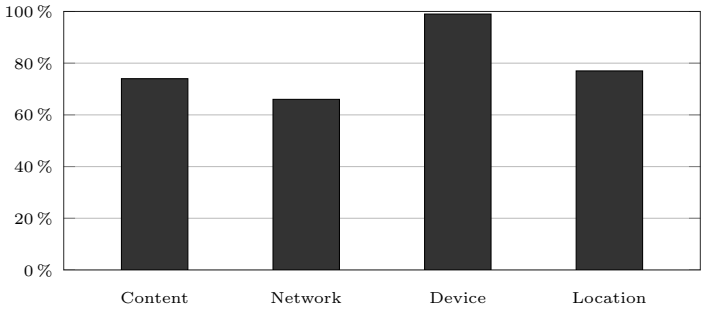


Figure 2.4: Type distribution of personal data collected by the analyzed apps.

2.6.2 Contacted Servers

The static analysis and the dynamic analysis module extracted, in total, 135 K and 21 K valid URLs/IP addresses, respectively. The number of URLs extracted by the static analysis module is much more as compared to those extracted by the dynamic analysis. The disparity in these numbers further endorses that static analysis provides an over-approximation of the program and extracts URLs which might not be contacted in an actual program execution, whereas the dynamic analysis extracts only those URLs which are contacted by the app in a single run. However, the presence of any of these URLs in the executable of an app provides a potential data transfer point and cannot be ignored. Since the purpose of dynamic analysis is to widen the analysis range, we use a combination of the URLs/IP addresses extracted by both the modules. Figure 2.5 illustrates the value of the dynamic analysis module to the static analysis results; where the red line shows the number of URLs found in a particular app and the blue line represents the number of new unique URLs discovered only by dynamic analysis in the same app. For certain apps the number of new URLs/IP addresses discovered by the dynamic analysis in comparison to the static

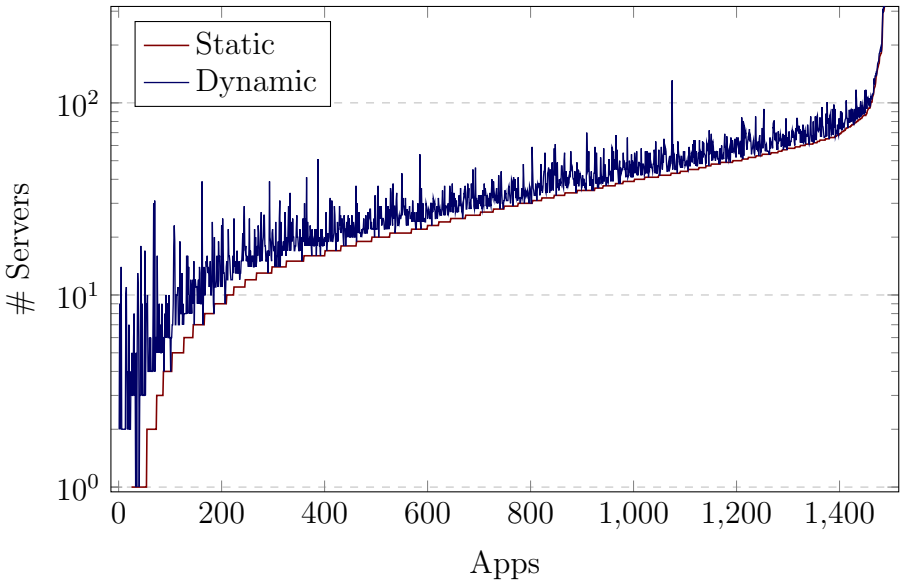


Figure 2.5: This graph shows that how effective is the dynamic analysis technique in covering the blind spots of the static analysis technique.

analysis is much higher than the others possibly because of heavy use of reflection.

It is important to mention here that the relation between servers and URLs is one-to-many, *i.e.*, on each server there can be multiple resources represented by different URLs. Therefore, the number of servers an app contacts is considerably less than the number of URLs.

Moreover, PDTLoc could extract data flow paths only for a portion of all the URLs due to known limitation of static and dynamic analysis. Therefore, we divide the servers into two groups, *i.e.*, those which are involved in an observed data transfer and those which are only contacted. Figure 2.6 provides the number of servers and apps for which at least a personal data transfer is observed. Overall for 505 (34%) apps, transfer personal data is observed among which 295 (20%) of the apps transfer personal data outside the EEA. Similarly, 401 servers are the recipients of data transferred by these apps among which 213 are located outside the EEA.

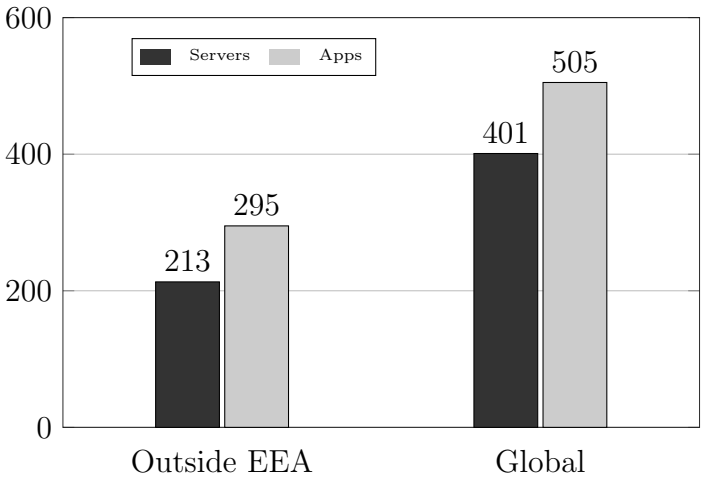


Figure 2.6: Number of servers and apps engaged in actual data transfer

2.6.3 Server Locations

Figure 2.7 illustrates the distribution of locations for servers engaged in the transmission of personal data (**RQ2**). As it reveals, only 23% of the servers are hosted in the EEA and the majority of the servers (67%) is in the US. Therefore, it is expected that the major portion of personal data to travel outside the EEA.

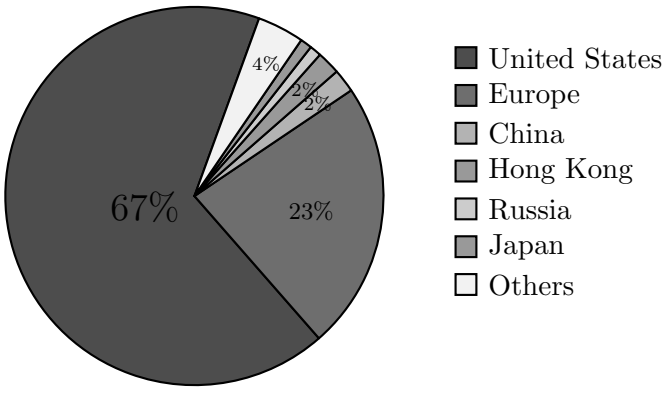


Figure 2.7: The distribution of the locations to which the European users' personal data collected by mobile apps travels.

The main focus of this work is to provide a location analysis of the

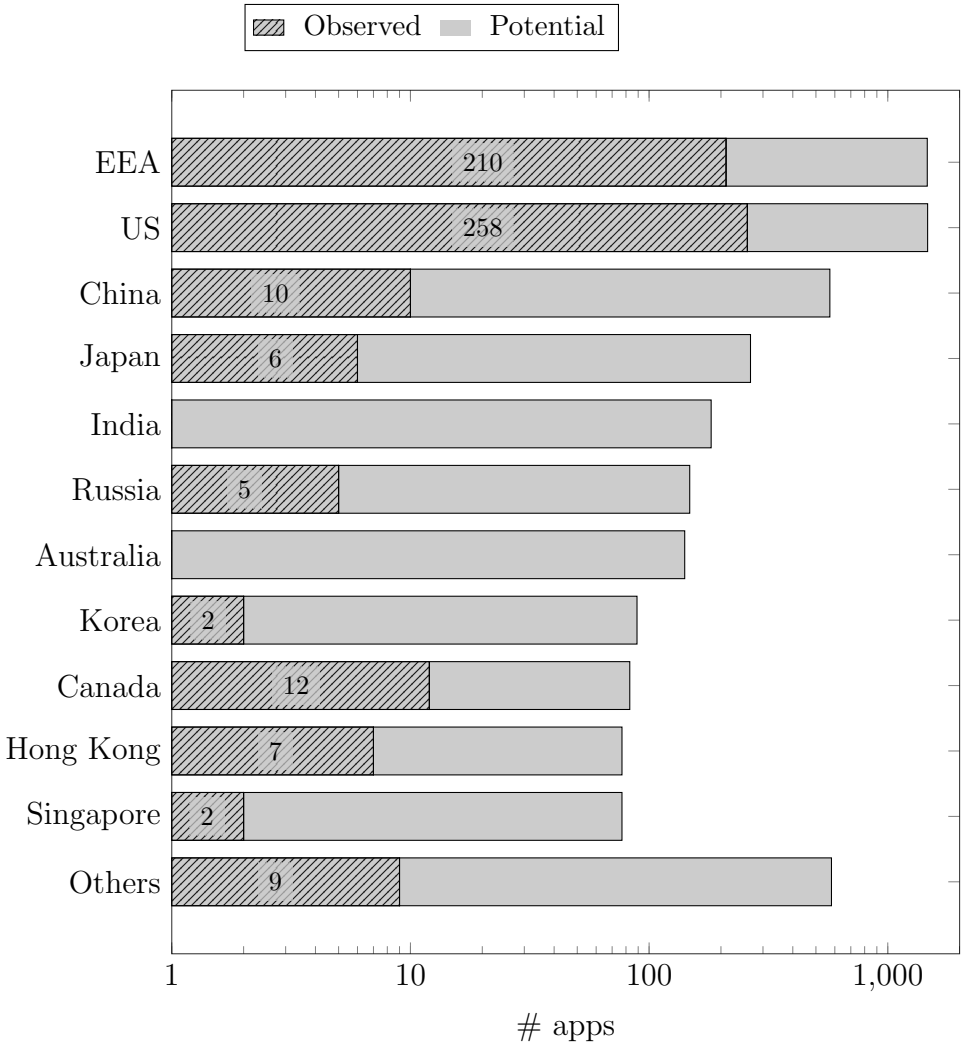


Figure 2.8: The target countries per apps.

servers contacted by the apps in our dataset. Figure 2.8 shows a graphical representation of the country-wise distribution of servers based on the number of apps. It illustrates that a reasonable portion of the apps contact (observed and potential data transfer) servers outside the EEA and US, especially China, Japan, India and Russia.

As most of the analyzed apps contact servers outside the EEA, it is interesting to know the number of apps transferring data only to a certain location/country. In this regard, Figure 2.9 illustrates the number of

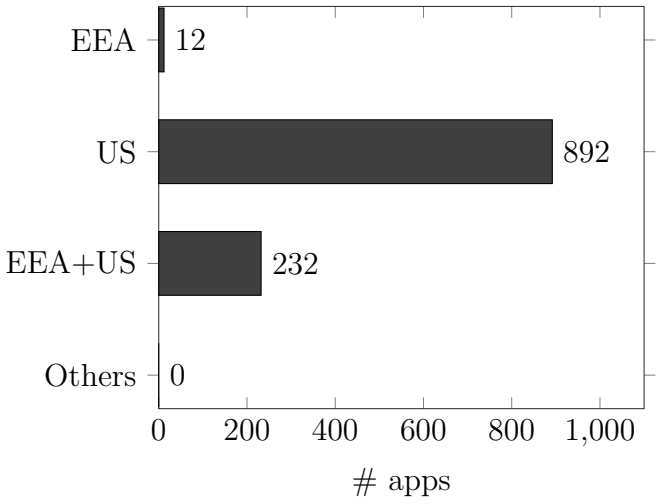


Figure 2.9: The number of apps that transfer the personal data exclusively to the EEA, US and other locations.

apps exclusively contacting servers located in the EEA, US, EEA & US and any other country. It shows that none of the apps perform exclusive data transfer to servers located outside the EEA and US. Only 12 (less than 1%) apps contact servers located only inside the EEA. In contrast, the number of apps contacting servers exclusively in the US is reasonably higher, *i.e.*, 892 apps. This implies that most of these apps either belong to the US-based companies or having their data centers located in the US. Similarly, the number of apps exclusively contacting servers in the EEA & US is 232. A similar reasoning applies to these apps as well where the apps either communicate to servers in the US or their local counterparts in the EEA.

2.6.4 Privacy Discussion

The new agreement between the EU and the US, the EU-US Privacy Shield, provides stronger obligation on the US based companies dealing with EU personal data. However, similar to the Safe Harbor, the EU-US Privacy Shield also control only a portion of the entities (service providers, apps)

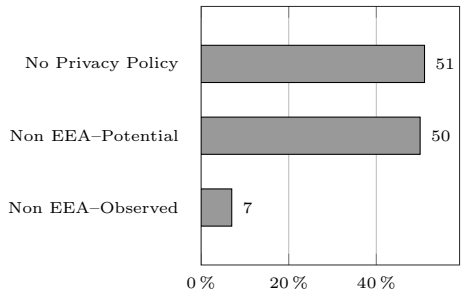


Figure 2.10: The apps that do not provide any privacy policy.

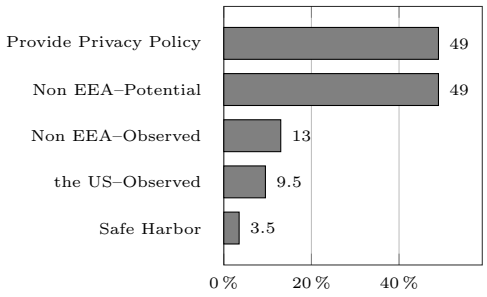


Figure 2.11: The apps providing privacy policy.

involved in personal data collection/transfer to US and other countries. Figures 2.10 and 2.11 depict the results of the analysis we have done on the mobile apps' privacy policy and terms of use. More than half of the most used apps in Europe, 51%, do not provide any privacy policy as shown in Figure 2.10. They simply do not tell their users what they do to the personal data they collect and where they store and process them.

In the app analysis, we observed that 7% (108) of the apps transfer personal data outside the EEA while do not provide any privacy policy; thus, this is a violation of the DPD'25.1 regulation (**RQ3**). Moreover, the analysis results reveal that 50% of the apps contact servers (potentially transfer personal data) outside the EEA and since these apps do not provide a privacy policy, they should anyways be considered suspicious.

Among all the apps providing privacy policy (49%), we observed that 13% transfer personal data to the non EEA based servers (*e.g.* , the US, China, Russia, etc.) while only 3.5% of them holding safe harbor certification (Figure 2.11). We concludes that 9.5% (134) of the apps certainly violate DPD'25.1 (**RQ3**) since users did not provide consent for those international data flows, and the available privacy policies are transparent about the data processing locations. Additionally, when we consider the apps which do not provide privacy policy and transfer personal data outside EEA 7% (108), in total, we confirm that 16.5% (242) of the analyzed

apps violate this regulation.

One of the major challenges for the Privacy Shield Agreement is that even if we assume that its enforcement will be practical, it will cover only a small portion of mobile apps dealing with European users personal data. The European Data Protection watchdogs would need to have a more proactive role in inspecting compliance with the Data Protection Regulations, in particular for widely used mobile apps.

2.7 Data Location Enforcement on Mobile Devices

There are three approaches to enforce jurisdiction policies on mobile phone: modifying the framework, repackaging apps, and hook APIs at run time. Since the two former approaches are quite costly and not practical in wide range usage, we choose the third category of approaches. Such approaches basically hook the APIs at runtime in order to allow the monitoring the data collection and data transfer activities of the app.

We design a module on top of *Xposed framework* [91] which intercepts the APIs transferring data to remote servers and analyzes their destination location by using the service provided by PDTLoc. This module then enforces the given jurisdiction policies on the device at the run time.

2.7.1 Xposed

"Zygote" is a process which is the heart of the Android runtime. Every application is started as a copy ("fork") of it. The process start is done with `/system/bin/app_process`, which loads the needed classes and invokes the initialization methods.

This is where Xposed comes into play. It copies an extended `app_process` executable into `/system/bin` on installation. This extended startup process adds an additional `jar`, which is called "Xposed Bridge", to the

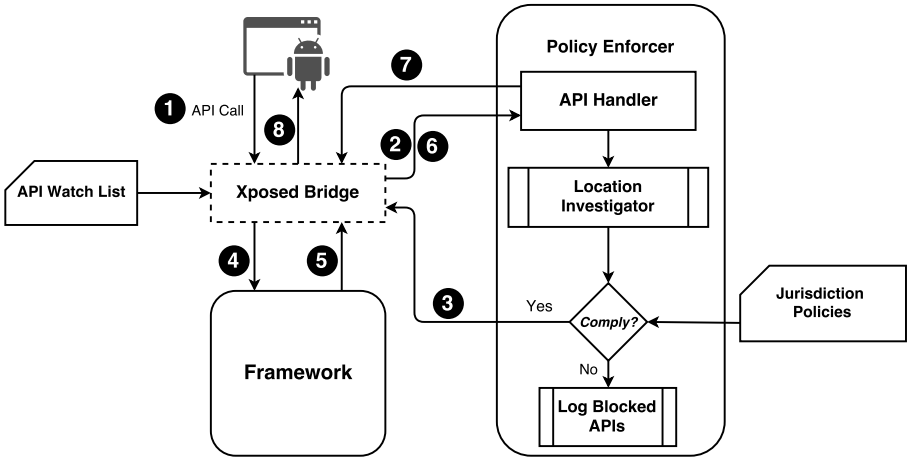


Figure 2.12: An overview of the jurisdictional policy enforcement mechanism on device.

`classpath` and `calls methods` at certain places. When an app launches, this `jar` file is executed in the very beginning of the process. It enables Xposed to "hook" method calls and inject a custom code before and after methods.

2.7.2 The Enforcer module designed on top of Xposed

Figure 2.12 illustrates an overview of the jurisdiction policy enforcement mechanism on an android based device. As the figure shows, `Xposed Bridge` is in between the app and the framework. It receives a list of APIs in order to intercept them. When the app calls an API, ①, Xposed Bridge intercepts it if it is in the list and forwards it to the `API handler` module, ②. API Handler performs pre-call or post-call executions in order to read or modify the parameters and the results of the called API. Moreover, it extracts the name/IP address of the server to which the incoming data is being transfered, and passes it to the `Location Investigation` module in order to determine the location of the remote server. Then the location of the server is compared against the given jurisdiction privacy policies and if there is a compliance, it proceeds the call, ③, otherwise

the call is dropped and logged. The Xposed Bridge forwards the call to the framework and receives the results, ④, ⑤. Then it sends the received results to the API Handler module again for further modification/logging if necessary, ⑥ and finally the results are forwarded to the app, ⑦,⑧.

2.8 Improving Transparency and Compliance

A number of actions are necessary to improve the control over trans-border personal data flows. It is important that the data protection authorities in Europe demand transparency from the application providers about the location of the data processing. This needs to be explicit in privacy policies. It is vital to clarify which parties have access to personal data and for which purpose. In our study, we observed some applications transferring sensitive personal data items to multiple servers across the globe. In addition to the jurisdictional issue, as all countries do not offer the same level of privacy protection to individuals, it is not possible to state that all those servers belong to the data controller, or even if the data controller is aware of them. Data Protection Authorities (DPAs) need to be proactive in protecting the privacy rights of individuals, by identifying international data which is not compliant with the regulations and agreements in place. The market place provider needs to make privacy policies mandatory, that is the minimum acceptable action that Google, Apple and Microsoft need to take, to mention the main companies who control mobile application marketplaces today. Ideally, the apps should display certification seal, but it is possible to go further. Research on machine readable and automated privacy policy enforcement has shown it is possible to offer more transparency and control to data subjects [9, 11, 17]. Moreover, the marketplace must have a mechanism to promptly remove applications signaled as non-compliant by DPAs or by the users. Furthermore, it is not difficult

to implement user notification features on the mobile OS, such that users can remove those non-compliant applications from their devices. On the other hand, it is extremely hard to reclaim the data that has already been leaked.

We are planing to provide PDTLoc as an online service. End-users, marketplaces and security agencies can utilize such service to perform privacy analysis of mobile apps.

2.9 Limitations

PDTLoc determines the physical location of the remote servers by employing a third party service *e.g.*, `IPAddressAPI.com`; thus, it relies on the information provided by this service.

Our server location analysis of the apps is based on the 1st hop server and do not consider if the personal data might be transferred to another server, e.g., App1 transfers data to `abc.com` and then the data is transferred from `abc.com` to `xyz.com`. In this case, PDTLoc only considers the transfer of data to the 1st server. It is impossible to trace data transfer once they are released to a server without collaboration of the target server. Furthermore, the mere transfer of the data towards another jurisdiction without explicit consent by the data subject and for which no international agreements are in place, already represents a violation. Therefore, PDTLoc only considers the transfer of data to the first hop server. Moreover, some applications might behave differently depending upon the location of the device they are running on. Since we have performed dynamic analysis in only one location (i.e., Italy), there is no guarantee about the behavior of such apps elsewhere.

The source and sink APIs considered in this work is a representative list of APIs which can be used by apps to retrieve personal data and transfer

it over the network. However, there are other methods to receive personal data and transfer it outside which is not considered in the analysis, e.g., apps can coordinate with each other to acquire and transfer data. Furthermore, we focus on only benign apps rather than malware that usually employ more sophisticated and stealthy methods to exfiltrate personal information.

The data flow paths extracted by the static analysis module only indicate the existence of potential personal data transfer in the app, but do not ensure if the app actually transfers personal data outside. However, even the existence of such data flow paths enables the app potentially violate DPD'25.1 and are, therefore, flagged in this work.

2.10 Related Work

Literature shows a number of research publications and tools which try to solve the problem of privacy leakages in Android apps. They focus on a wide range of private user data and are based on different strategies. Here we briefly discuss some of them in the context of our problem.

2.10.1 Static Privacy Leak Detection

A number of static analysis approaches have been proposed in literature which can serve to detect privacy leakage in Android apps. Based on the model of the Android framework, CHEX is an approach that performs data flow analysis to detect component hijacking vulnerabilities [60]. In principle, the same approach can be used to detect also privacy leakages.

Scandal [57] tries to detect leakage of private information, such as location information and phone identifiers, using media including Network, Files and SMS. It is based on identifying data flow using abstract semantics of the applications. Although, it provides a concrete representation of the

data flow, it consumes a lot of resources and would therefore suffer from performance and scalability issues.

Androidleaks is a WALA based solution to detect privacy leakages in Android apps [36] [48]. It uses a system dependence graph to perform taint analysis.

Based on bytecode analysis of Android apps, DroidAlarm is a tool designed to counter privilege escalation by detecting capability leaks [97]. It uses control flow graphs to detect and extract capability leak paths from to sensitive sources to public interfaces. However, it only supports Android 2.2 which is quite outdated.

AmanDroid is an inter component-data flow analysis framework for Android apps [85]. It is an extensible tool implemented in Scala and based on an intermediate representation of Dalvik bytecode. Amandroid performs data flow analysis by constructing an inter component data flow analysis graph. It provides a plugin for taint analysis which captures data flow between various sources and sinks of information. The sources and sinks are easily configurable in Amandroid's taint analysis plugin. Theoretically, these sort of tools are ideal for detecting privacy leakage. However, we practically tried it on some apps and it could not detect some very obvious data flows.

Bodden *et al.* presents, Flowdroid, one of the most sophisticated static analysis tool for Android [8]. It is a Soot based tool which performs data flow analysis on a representation of Java bytecode called Jimple [82]. They also publish a benchmark of applications, known as DroidBench, which can be used to test data flow analysis tools.

Epicc is another static analysis tool which focuses on privacy leakage considering inter component communication and inter app data flows [68]. They provide a cover for privacy leakage between various components of an app and among multiple apps, which most of the static analysis tools

do not consider. To make it a complete package, Didfail and IccTa are two other tools which combine Flowdroid and Epicc [15] [59]. They utilize the object/field/context sensitivity of Flowdroid and the inter-component data flow detection Epicc to construct superior tools. This chain of static analysis tools, however, is based on Soot that was designed for Java applications and some times fails to analyze Android apps.

As a matter of fact, some of these static analysis tools capable of detecting privacy leakage can be adopted to be used in our work. However, we preferred performing analysis on Smali code as it provides a direct representation of the Dalvik bytecode. Therefore, we used an extension of SAAF that performs analysis on Smali code and is based on backward program slicing of apps. The extension of SAAF overcomes some of its limitations, such as handling data flow through intents.

2.10.2 Dynamic Privacy Leak Detection

As static analysis usually suffers from over-approximation and, therefore, a higher number of false alarms, dynamic analysis solutions provide the answer.

SmartDroid detects sensitive APIs in an app, creates a static activity switch path and control flow paths leading to these sensitive APIs and dynamically executes these paths to generate trigger inputs which could be used to detect privacy leakage [96]. They rely on instrumentation of framework services to ensure dynamic execution.

TaintDroid is one of the most widely cited tools in Android dynamic privacy leakage detection [30]. It is based on tainting sensitive information and tracking it towards sensitive sources. Similarly, Droidbox is another tool which detects privacy leakage in Android apps by executing them in an emulator [27]. However, such tools require a dynamic triggering solution to effectively execute portions of the code that leak private information.

To counter this problem, some tools provide their own triggering solution along with privacy leakage detection, e.g., AppsPlayground, AppIntent, etc. [74, 93].

However, most of these tools still suffer from code coverage issues and increasing the code coverage when analyzing Android apps is an open research problem. Moreover, Shauvik *et al.* performed an analysis based study of the state-of-the-art open sourced test input generation tools for Android applications [19]. Surprisingly, random exploration strategies based tools performed far better than the other model based and systematic tools.

Since even the more sophisticated tools do not yield considerable improvement in the code coverage and unnecessarily complicate the process, we use the standard application exerciser provided with the Android SDK,*i.e.*, the Monkey tool, in the dynamic analysis module.

2.11 Chapter Summary

This chapter introduces a substantial contribution in the analysis of transborder personal data flows. It is a major debate that may impact how the regulatory framework around the digital economy will evolve. We have highlighted the main concerns in personal data transfers by in principle non-malicious applications, and shown a considerable number of them fail to comply with the EU personal data protection regulation, in the first study of the kind, up to our knowledge. While PDTLoc has been suitable in this case, we believe it can be extended to analyze other information flow properties as well.

This work is interviewed by CNIL, the French National Commission of Information Technology and Liberty, and they wrote an article about PDTLoc which is publicly available here: `https://linc.cnil.fr/where-does-all-data-go`.

Chapter 3

VLOC: An Approach To Verify The Physical Location Of A Virtual Machine In Cloud

In this chapter we introduce an approach, named VLOC, to verify the physical location of a virtual machine on which the data controller applications and the collected data from end users are stored. VLOC is implemented as a software tool which is able to estimate the geolocation of itself and notify the corresponding user if the location is unauthorized. VLOC uses a number of arbitrary web-servers as external landmarks for localization and employs network latency measurement for distance estimation. Due to the fluctuation in the network latency, VLOC employs a machine learning technique in order to adapt itself to various network latency tolerance.

3.1 Introduction

According to the National Institute of Standards and Technology (NIST), one of the essential characteristics of cloud computing is resource pooling which allows cloud service providers (CSPs) to serve multiple consumers using a multi-tenant model by dynamically assigning resources on consumers'

demand [63]. Cloud service provisioning is independent of the location of the provider, as the services are consumed over the Internet. CSPs wish to be free to relocate data for load balancing purposes in order to reduce the maintenance cost. However, knowing and controlling the physical location of data for storage and processing is a the key requirement for enforcement of the jurisdictional regulations. Moreover, it could be very important for organization (e.g. data controllers) using Cloud in some particular scenarios dealing with compliance [3]. Due to lack of appropriate monitoring mechanisms, a piece of sensitive data may be transferred amongst various data centers situated in different geographical locations, and consequently there might be violations in data privacy as there are various regulations for privacy protection in different countries.

Since data controllers need to guarantee that the country where the processing occur has an adequate level of protection to the rights and freedoms of the individuals from whom the data was collected, they would benefit from a service that could verify the physical location of their data/virtual machines. There are a number of approaches for finding the physical location of a piece of data or a host. Generally, they take advantage of network metrics such as round trip time delay for a transmitted message between two identical hosts and then calculate the distance or the physical location of one of hosts based on the measured latency from the other ones. The main drawback of this approach is dynamicity of the internet. As the network load changes frequently in time, it is not possible to find a constant correlation between network latency and physical distance. In addition, there are other factors which impose delay on a transmission such as authentication mechanisms, network delays, proxying, caching, and so on. Therefore, an adaptive approach is required to deal with the dynamic environment of the Internet.

In this chapter we introduce a geolocation approach, named *VLOC* (a

Verifier for physical LOCation of a virtual machine), which is able to verify the physical location of a virtual machine by taking advantage of nearby randomly chosen web-servers. Since VLOC does not rely on a network of fixed landmarks, its implementation is easier and maintenance cost lower than other proposed solutions. VLOC is implemented as a software component which needs to be installed and initialized on a virtual machine.

3.2 Related Work

Peterson et al. in [72] introduced the idea of combining the concept of Internet geolocation with Proof of Retrievability (PoR) for data localization. *GeoProof* is an implementation of such an idea [3]. It uses a tamper-proof physical component installed in the local network of cloud servers. As this component is GPS [1] enabled, it is able to recognize its own location. In addition, GeoProof employs a PoR protocol [54] by which it challenges the storage servers. The information gathered from the PoR protocol and the physical component enable it to verify the location of a piece of data.

The major drawback of GeoProof is the requirement of a tamper-proof and GPS enabled device situated inside the local network of each data center. Cloud providers may hesitate to adopt such solutions as it may leak sensitive information. In [4] GeoProof is enhanced by reducing the required computational overhead and improving its accuracy, but the mentioned drawback remains unresolved.

As distance bounding protocols such as [75, 41, 72, 3] use network latency for distance calculation, they are quite time critical. Therefore, network fluctuation significantly decreases their accuracy. Network latencies can be imposed by network equipment and servers. Such latencies can not be distinguished from message transmission latency. Hence, distance

[1] Global Positioning System

bounding protocols suffer from lack of accuracy in dynamic environments such as Internet. Gondree and Peterson proposed a schema to tackle such problem by employing a latency function built based on the current network traffic observation [37]. In their schema, there are a number of landmarks which observe the network traffic by transmitting a number of messages amongst themselves and then build a model based on that. The main disadvantage of this approach is the requirement of a dedicated network of landmarks which is quite costly. Moreover, in the model building phase the landmarks send messages amongst themselves in order to find a baseline for the Internet delay which does not quite represent the real environment. In fact, this scenario does not consider the latencies imposed by cloud mediation services such as authentication, decryption, etc. Therefore, the observation has an inherent error which influences the distance estimation.

DLAS provides a data localization assurance service based on cryptographic foundations that allows cloud users to select the preference regarding data location [65]. In order to provide such service, DLAS uses a Zero Knowledge System (ZKS) protocol to maintain secrets and verify them as mentioned in the Service Level Agreement (SLA) between parties. In DLAS, the CSP (called enterprise in that paper) is trusted and uses an external cloud storage service and guarantees not to move user's data according to her location preferences. The storage provider (SP) prepares a list of all data centers with their physical locations and informs the CSP once a piece of data is moved. Employing ZKS protocols enables the CSP to verify the region of a particular data center and prevents the CSP from violation of the data location preferences policies. Since DLAS does not use any external resource for geolocation and relies on logical characteristics of data centers, it is vulnerable to be bypassed by virtualization. A copy of network topology of all data centers (*i.e.* empty virtual machines and settings) can be stored on each data center and a piece of data can be

moved amongst them without awareness of DLAS. Our approach, VLOC, does not suffer from this kind of attack.

Massonet et al. introduced a system which monitors data transfers by making collaboration between cloud infrastructure provider and the service provider (i.e. user) [61]. In this system, data controller (*i.e.* tenant or cloud customer) is able to specify required locations for a piece of data allowing to be processed and the system prevents moving data to unauthorized locations. However, its major drawback is providing such a monitoring service only at infrastructure level. Therefore, it does not cover data items with finer granularities. This drawback is resolved by another work [26]. It introduces a vast monitoring framework being able to collect evidences about data transfers in various service levels. Basically this framework employs a dedicated monitor for each of service layers including $SaaS$, $PaaS$, and $IaaS$. Each monitor tracks the API calls related to data transferring and stores required logs. Furthermore, in order to track the movements of a piece of data in various layers, this framework keeps a map amongst different granularities for the data. This framework is promising; however, there is an assumption which says the CSP wishes to demonstrate compliance; therefore, it does not move user's data without authorization. This assumption is quite reasonable as there are many ways to make a copy of data without having authorization. However, restricting the known ways of copying and transferring data and employing a geolocation technique mitigate the risk of illegal data transferring. Due to this assumption, CSP provides a list of all data centers with their physical locations. In our attack model, we assume that the CSP is not trustworthy as its goal is to minimize maintenance costs by moving resources to less expensive data centers.

3.3 VLOC

The user can install the VLOC on her virtual machine and once the tool gets initialized, it notifies the user the physical location of the virtual machine. VLOC does not need a dedicated physical device nor a network of pre-arranged landmarks. The main requirement of VLOC is the availability of an online IP geolocation service like [50, 62], to get a list of websites like *Alexa 1-million* [5] and the current geolocation of the virtual machine. First, the tool chooses a, configurable, number of random websites (agreed with the CSP at the moment user buys its cloud hosting service) and then starts to collect geolocation information about them. Then, it can verify at any moment, the geolocation of the virtual machine.

3.3.1 System Model

In the following we describe the main ingredients of the geolocation system in which VLOC verifies the location of a virtual machine.

- The **list of websites**; A database of websites addresses. For instance, these addresses can be collected from Alexa [5].
- The **VLOC tool**; it is a software component installed on a virtual machine to verify its physical location. VLOC includes a **Data Collector** component which collects the required geolocational information for every website. The **IP location service** provides geolocational information of the web-server of a particular website. The **Round-Trip-Time (RTT) measurement module**, which measures the network latency between current virtual machine and a target web-server by sending multiple HTTP requests to the website hosted on that web-server. The number of HTTP requests can be specified through a parameter passed to this module. Finally, the av-

erage value of round trip time of the successful requests is returned as a result.

- The **target virtual machine**; This is the virtual machine that needs to be securely geolocated and on which the VLOC tool is installed. The virtual machine holds data as well web services users want to run on the cloud.

- **Current host**; it is the physical server on which the virtual machine is running.

- The **distance estimation function** which maps each RTT value to a distance between the pair of associated hosts. This function is a polynomial function and its coefficients are variable and updated during the initialization of the VLOC tool. The value of coefficients are calculated based on collected data and their corresponding measured RTT values. Therefore, the function is able to estimate the distance between two identical hosts based on former observations.

- The **learning module** calculates the coefficients of the distance estimation function by finding the correlation between measured RTT values of two identical hosts and their associated distance. This module attempts to find the best function approximation which represents the collected data.

- The **triangulation technique**; this is a technique used to specify the physical location of a point by having the latitude and longitude coordinates of at least three nearby points. This technique is used, in our model, to estimate the physical location of the current host based on three nearby web-servers. The technique is explain in details in the next section.

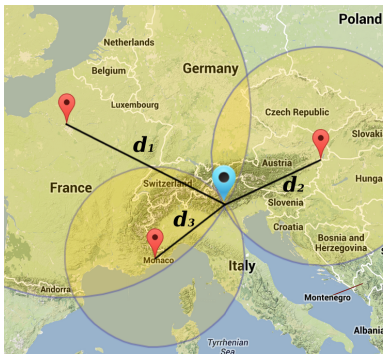


Figure 3.1: Triangulation for specifying the physical location of a host by knowing the physical locations and distances from other nearby hosts.

3.3.2 Determining the Physical Location of A Virtual Machine

To find the physical location of a server on which a virtual machine is installed, a feasible solution is to take advantage of quality of service metrics used in networks. In [72], multiple trusted landmarks with known physical locations are used. The distance between a landmark and a data center is obtained by sending specific messages and measuring transfer delay with a an error below a chosen threshold. At least three landmarks are required to achieve the required accuracy in triangulation procedure.

Computing the location of a point by triangulation

Computing the location of a point, p_x, on a surface is possible when we have the locations of at least three nearby points, $\{p_1, p_2, p_3\}$, and their distance from p_x. As Figure 3.1 illustrates, if we draw a circle with the center of each point and the radius of their distances (d_1, d_2, d_3) from p_x, all the circles meet each other at p_x. By finding their intersection point, we can determine the location of p_x. This technique is called "*Triangulation*" [90] or "*Trilateration*" [87].

Before we utilize triangulation technique, we need to consider that since Earth is not a surface rather a sphere, the location of objects on earth is

represented by the latitude (ϕ) and longitude (λ) values which are defined in polar system. The latitude of a point is the angle between the equatorial plane and the straight line that passes through that point and through (or close to) the center of the Earth. The longitude of a point is the angle east or west of a reference meridian to another meridian that passes through that point [88]. In order to calculate the intersection points of the circles, we convert this coordination into the Cartesian system by utilizing Equation 3.1 and for the reverse operation, Equation 3.2.

$$\begin{aligned} x &= \lambda . \cos(\phi) \\ y &= \lambda . \sin(\phi) \end{aligned} \tag{3.1}$$

$$\begin{aligned} \lambda &= \sqrt{x^2 + y^2} \\ \phi &= \tan^{-1}(\frac{y}{x}) \end{aligned} \tag{3.2}$$

We write the equation of a circle in the following form:

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2 \;\; i = 1 \ldots n \tag{3.3}$$

where (x_i, y_i) indicates the center of the circle (the location of the i^{th} device in our system) and d_i its radius, which is the distance between the server and the device. In order to find intersection point of multiple circles, we do it two by two *i.e.* in pairs. However, before trying to find intersection of two circles we have to figure out if they touch each other. Suppose that we have two circles i, j; if we draw a line between the two centers (d_{ij}), compare its length with the radii (d_i and d_j) and employing the triangle existing conditions [89], we can conclude that whether those circles can be used for our purpose or not. We obtain the distance between the centers of two circles by measuring their Euclidean distance as the following:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{3.4}$$

The situation of the two circles is determined by the following conditions:

- $d_{ij} < \sqrt{(d_i - d_j)^2}$: One circle is inside the other so there is no intersection.
- $d_{ij} > d_i + d_j$: The circles are too far apart to intersect.
- $d_{ij} = d_i + d_j$: The circles touch at a single point.
- $d_{ij} < d_i + d_j$: The circles touch at two points.

If two circles touch at least at one point, we subtract their two equations, in 3.3, to get the line equation. By solving that subtraction the following equation is yielded which determines the intersection point(s):

$$\begin{aligned}(x,y) = \frac{1}{2}(x_j + x_i, y_j + y_i) + \frac{d_i^2 - d_j^2}{2d_{ij}}(x_j - x_i, y_j - y_i) \\ \pm\frac{1}{2}\sqrt{2\frac{d_i^2 + d_j^2}{d_{ij}^2} - \frac{(d_i^2 - d_j^2)^2}{d_{ij}^4} - 1}(x_j - x_i, y_j - y_i)\end{aligned} \tag{3.5}$$

In order to compute the location of a point, this equation is applied on the locations of at least three nearby points and yields the intersection point which equals to the location of the first point.

Estimating Distance

The most important factor in determination of the location by triangulation is the distance of landmark points (hosts) with our virtual machine. The virtual machine sends HTTP requests to three hosts and measures the round trip time delay. Then, based on measured delays, distance from every host is estimated and by utilizing a triangulation technique, the physical location of the host on which the virtual machine is installed will be computed.

As mentioned before, VLOC needs an initialization which consists of three phases. The first phase is to collect geolocational information of the given list of websites. Algorithm 1 shows the procedure of data collection. This algorithm takes a list of websites, an online IP geolocation service, and the location of the current host (the virtual machine) and then finds the physical location of web-server of each website and calculates the distance between the web-server and the current host and stores them into a database. The second phase of initialization, which is depicted in Algorithm 2, is measuring the round trip time (RTT) delays of websites. This algorithm takes the list of websites, a range of operation which signifies the radius of a circle showing a geographical zone, and a confidence factor and then it measures the RTT value of an HTTP request for every website. As using long distances increases the error rate of distance estimation, our approach limits its range of operation to the nearby websites and in Algorithm 2 the range of operation refers to choosing websites situated in range of R KMs. Due to probability of failure in the requests and the delay of packet routing imposed on some requests, this algorithm takes a parameter named confidence factor C which repeats the HTTP transmission operation C times for each website and finally the average of successful HTTP requests is used. Since after initialization phase the physical location of the current host needs to be verified and the only trustworthy entity is network delay measurement, it is required to provide a function which maps an RTT value to the corresponding distance. Having distance from at least three hosts enables the current host to calculate its physical location by making use of a triangulation technique. Therefore, the last phase of initialization is to prepare a function being able to estimate the physical distance between the current host and an arbitrary host.

The distance estimator function, in VLOC, employs a distance bounding protocol in order to calculate distance between two geographical points

Input: L: list of websites; IPG: reference of IP geolocation service; H: current host information;

Output: L': List of websites with their collected geolocation information;

```
1 L' = new List();
2 for ( w in L) do
3     g = IPG.getInfo(w);
4     d = distance(H, g);
5     r = {w, g, d};
6     add r to L';
7 end
8 return L';
```

Algorithm 1: The data collection algorithm.

based on their measured RTT value. The following equation shows a simple distance calculator:

$$f(x) = a.x \tag{3.6}$$

where x is the given RTT value and a is a coefficient that converts the value of a round trip time delay to its corresponding distance. Unfortunately, due to dynamicity of packet transmission in the Internet, it is not possible to consider a constant coefficient for the distance estimation function. Moreover, the hierarchical architecture of cloud does not allow the protocol to work properly as in order to transmit and process a request, the request needs to pass through various service layers. In addition, each layer imposes an extra delay on the process and the number of participated service layers is vary per different types of requests. Therefore, in order to estimate the transmission latency of a request, it is not possible to consider a global constant coefficient for the distance calculation function. Thus, a technique is required being able to adapt itself with various circumstances and handle different delays in the distance calculation procedure. VLOC

Input: L': List of websites with their geolocation information;
R: Range of operation;
C: Confidence factor;
Output: L'': List of chosen websites with measured RTT;

```
1  L'' = new List();
2  for ( r in L' ) do
3      if ( r_d < R ) then
4          for i = 1 to C do
5              Send an HTTP request to r_w;
6              t_start = Now();
7              Wait for respond from r_w;
8              res = The received response;
9              t_end = Now();
10             if ( res was successful) then
11                 Δt_i = t_end − t_start;
12             end
13         end
14         rtt = ⟨Δt_{1...C}⟩; // Average
15         rec = {w, rtt};
16         add rec to L'';
17     end
18 end
19 return L'';
```

Algorithm 2: Measuring and collecting round trip time (RTT) latencies of the nearby websites.

uses the following equation:

$$f(x) = \sum_{i=0}^{n} a_i x^i \tag{3.7}$$

in this equation the coefficients are variable and they are updated according to the observation performed by Algorithm 2. This algorithm chooses a random subset of websites from the list and measures the RTT values for them and then updates the coefficients. In order to provide an accurate observation, a sufficient number of websites must be used (e.g. at least 500 websites). Therefore, the algorithm is able to cover the regular turbulences happening in the network as depicted in Figure 3.2, the coefficients do not face abrupt changes; they stay in a limited range.

Figure 3.3 illustrates an example of the distance estimation function. As this figure shows, each item (*i.e.* website) has an RTT value and a corresponding distance from the current host. The coefficients of the function are obtained by applying a machine learning technique (*i.e.* Polynomial Regression [86]) on the collected data. The function shown in this figure is an example of trained function which is able to perform distance estimation for further RTT values. Therefore, employing this technique enables us to handle the dynamicity of the Internet environment.

An interesting question might arise here, since the environment of the cloud computing is changing in time the distance estimation based on one time observation might not be accurate enough. In fact, this is likely to be true because network and host conditions may be different at the time of observation and at the time of estimation. Therefore, there must be a short time gap between observation and estimation. However, the size of this gap depends to the fluctuation of latency of the network. In order to maintain the accuracy above an accepted level, the observation needs to be performed periodically. In fact, while the estimation function is being used, the coefficients can be updated frequently and provide better accuracy

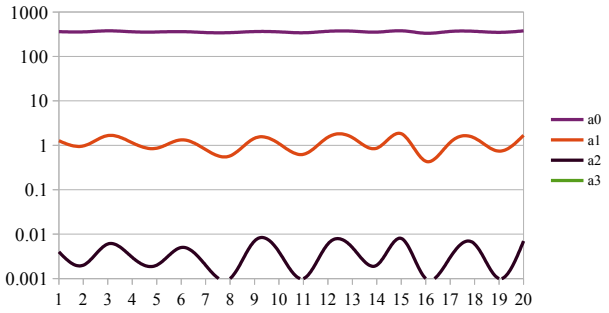


Figure 3.2: An observation on the changes of the coefficients of Equation 3.7 during the update process captured 20 times. These results show that choosing a random subset of websites for each update, does not lead to very different coefficients.

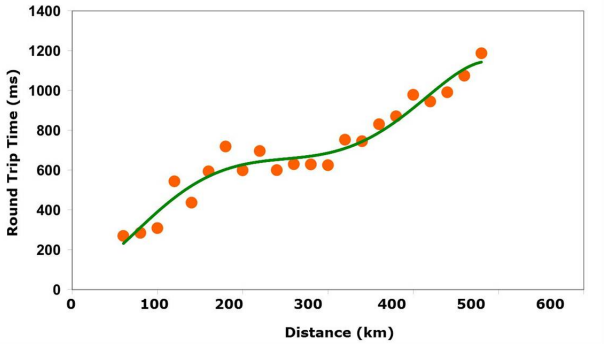


Figure 3.3: A sample of collected RTT values versus distances and the trained function representing the distance estimation procedure.

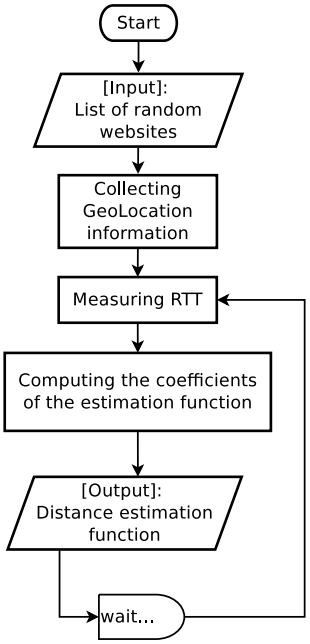


Figure 3.4: The initialization process. The process of frequently updating the coefficients of the distance estimation function is illustrated in this figure.

based on the most recent observations. In order to utilize such a technique, we need to perform the observation procedure in every predefined time slot; and then update the coefficients based on them. The entire process is depicted in Figure 3.4.

Once the initialization phase is finished, the distance estimation function is ready to be used. In order to verify the location of current host (virtual machine), at least three nearby websites get selected and then by making use of the mechanism employed in Algorithm 2, the RTT value for those hosts are measured. In the next stage, the distances between current host and the selected websites are estimated by utilizing the distance estimation function. Finally, as illustrated in Figure 3.1, the geolocation of current host is obtained.

The presented approach can be integrated into various techniques and schema in order to be used in various application. For instance, it can be integrated into a proof of retrievability service (PoR) such as [3, 54,

55, 66, 72] or into a data transfer monitoring framework such as the one introduced in [26]. Moreover, it can be used as a notification service on transferring a specific piece of data to an unauthorized physical location.

3.3.3 Security Considerations

Alternative approaches to ours use a network of pre-arranged landmarks, situated outside of the target host, as verifiers. They send challenge messages to the host and measure the RTT values and finally estimate the physical location of the host. All these approaches however, require an external network of landmarks.

In contrast, our approach puts the verifier inside the target host, removing in this way the requirement of an organized network of landmarks. In VLOC, the physical location is estimated by sending message from the target hosts to existing websites, rather than from some specified landmarks to the target host.

This however, opens security issues that VLOC needs to address. Since VLOC is in the virtual machine hosted on the cloud, a mistrusted cloud provider could intercept and manipulate all communications between VLOC and the websites. Encrypting the messages is not a solution, since the encrypting key would reside on the cloud and can be extracted from the RAM by the cloud provider. Our solution to this problem is to obfuscate the communications VLOC performs for the purpose of estimating the distance within the regular traffic of applications stored on the virtual machine. The reasoning behind this choice is that the cloud provider could not easily filter out or block these messages and a full packet inspection would be required. Since in our threat model the cloud provider moves data only for the purpose of saving money, breaking our system would simply require more effort than what gained by moving the data.

Therefore, VLOC does not use fixed landmarks, easy to blacklist, but

rather randomly chosen websites as external landmarks. In order to measure required RTT values, we use normal HTTP requests which would be difficult to block without affecting other applications. The cloud provider sees the virtual machine sending an HTTP request to some websites like what many applications do for REST requests or SOAP ones.

An other possible point of attack is the list of websites VLOC will query. Rather than embedding a fixed list in VLOC software, the user can configure online and dynamically at her will the address of the IP-location service VLOC uses to gather list and location of the candidate websites. The size of the list and the number of selected website can be configured dynamically as well.

3.3.4 Limitations

Although VLOC is promising in a practical environment, there are a number of limitations need to be considered. One of the limitations is related to detection of network latency changes (*i.e.*, due to network disruption). This may have an impact on the accuracy of the estimated location. While an adaptive monitoring module capable of such detection is under development, at the moment VLOC adopts the strategy of periodically repeating the measurements. The frequency of such confirmation is a parameter can be configured dynamically.

Furthermore, there are two other parameters in VLOC need to be tuned in order to achieve the best accuracy. Those parameters which are the range of operation, R, and the confidence factor, C, impact on the amount of noise in the measured latencies and consequently on the accuracy of the geolocation estimation procedure. The confidence factor plays a crucial role in the measurement phase as it attempts to handle the fluctuation of the network while the duty of the range of operation is to filter out far web-servers. Since long distances overwhelm the impact of short distances

in training, they reduce the accuracy of geolocation estimation as it is demonstrated in Figure 3.8.

As the cloud provider is considered as an adversary, it can perform some operations to reduce the accuracy of VLOC. For instance, it can inject packet delays on all outgoing and incoming traffic. These delays could be recorded, or they could be randomly generated. At the time of injecting these delays, VLOC faces a slight reduction in accuracy, however, since VLOC observes the environment frequently and adapts itself with the network turbulences, it can get adapted to the such a situation. The cloud provider is agnostic to the purpose of outgoing and incoming packets as VLOC packets are exactly regular HTTP requests. Furthermore, if the cloud provider performs such an operation, it impacts the quality of the service which is an important key in cloud business.

3.4 Empirical Setup

In order to evaluate VLOC, we developed it in form of a web-based tool in PHP/MySQL which collects the data and executes the training and accuracy measurements. The target host is a computer in Trento, Italy and the goal is to estimate the geolocation of this computer.

This section explains the data collection process and describes the data used. It also describes the evaluation measures, the experimental results and their analysis.

3.4.1 Data Collection

As mentioned before, the initialization phase needs to collect geolocational information and measure RTT latencies of a number of randomly chosen websites. In order to do so, we used *Alexa 1-million* [5] list from which the geolocational information of 188, 644 websites were collected. We have

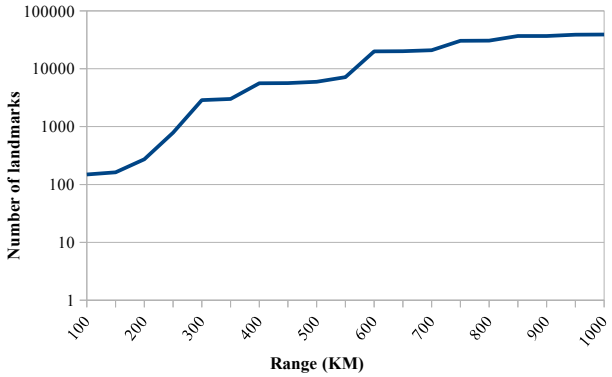


Figure 3.5: The number of used landmarks (websites) per various ranges. Each range refers the maximum distance between the landmarks and the current host.

used *IPaddressAPI.com* [50] as the IP-location service. This operation is performed by Algorithm 1. After collecting such information, we selected 38,892 websites which were geographically located in the vicinity of 1000 KM radius. We measured the RTT values of these website by employing the Algorithm 2. Figure 3.5 illustrates the number of these websites (used as landmarks) in various ranges. The confidence factor for this measurement was set to 10, which means for every website 10 HTTP requests were sent and the average of successful ones was stored as the corresponding RTT value. The HTTP requests used in the experiments were sent through a PHP function named *fsockopen* [73] which initiates a socket connection to a resource in network. We used this function to open a connection to a given website address on port 80 referring to the port of HTTP protocol. Since the application only opens a socket and does not download any web-page, it acts resembling a ping request over HTTP protocol.

3.4.2 Evaluation Measure

Since the main purpose of VLOC is to verify the location of a virtual machine, the evaluation measure must be able to verify the distance between the actual physical location of the machine and its estimated location. We

used error of average distance estimation defined in the Equation 3.8 for accuracy evaluation.

$$E_{avg} = \frac{1}{N} \sum_{i=1}^{N} \|p_e^{(i)} - p_o^{(i)}\| \tag{3.8}$$

where N is the number of data instances participating in test phase, $p_e^{(i)}$ denotes the estimated physical location for i^{th} website in the list and $p_o^{(i)}$ is the observed geolocation (the real physical location) for that website. Finally, E_{avg} refers to the calculated average error in KM.

In order to provide a comprehensive evaluation, we evaluated the approach with different websites which is achieved by applying random combination of measured RTT values. We used cross validation technique [42] to perform such a combination by using 10 fold 5 times setting which works as follows. First, the collected data is divided into 10 parts called folds. Out of these 10 parts, 9 are used to construct the estimation function. The remaining 1 fold is used to evaluate the constructed function. Then the data items get shuffled and the same division and evaluation is repeated for 5 times.

3.4.3 Accuracy

In this section the accuracy of location estimation is discussed. In order to estimate the physical location of a server, first we need to estimate the distances of at least three nearby hosts. The accuracy of distance estimation makes a major impact on the accuracy of physical location estimation (i.e. triangulation procedure). We also compare the accuracy of VLOC with other distance measurement techniques which are GeoProof [3] and distance calculation with the speed of light.

GeoProof uses the following equation for its distance measurement:

$$f(x) = \frac{1}{2}x\frac{4}{9}s10^{-6} \tag{3.9}$$

where x is the given RTT value, $\frac{4}{9}s$ is the measured speed of transferring data over the Internet while s is the speed of light. This function $f(x)$ takes x in milliseconds and computes the distance in KM. Since there is no accuracy evaluation experimental results provided by GeoProof in their paper, in order to compare its accuracy with the accuracy of VLOC, we applied GeoProof distance estimation formula on our measured RTT latencies.

The other experiment is done by using speed of light in fiber as the calculation parameter which would be as follow:

$$f(x) = \frac{1}{2}x(0.66)s10^{-6} \tag{3.10}$$

As the speed of light in fiber is 66 % of the speed of light in vacuum, it can be used as measure for distance calculation. However, this measurement can be solely used in a high speed network with neither routers or other kind of nodes in the middle. We consider it as a theoretical baseline. VLOC provides a more realistic correlation between distance and observed network latency over the Internet. VLOC builds a model representing such a correlation and uses it for distance calculation. Since the model is build based on observation of network latency regardless the type of network environment, it is able to estimate the distance between two hosts based on their message transmission latency. In addition, building a specific model for the current network and updating the model based on the changes in the latency of the network enable it to handle the fluctuation of the transmission latency. This is achieved by tuning the coefficients of the estimation function, introduced in Equation 3.7, with the measured RTT values. Figure 3.6 shows the impact of adaptive approach on the accuracy

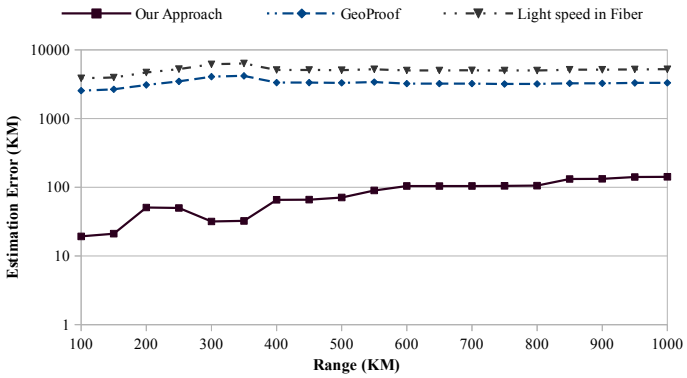
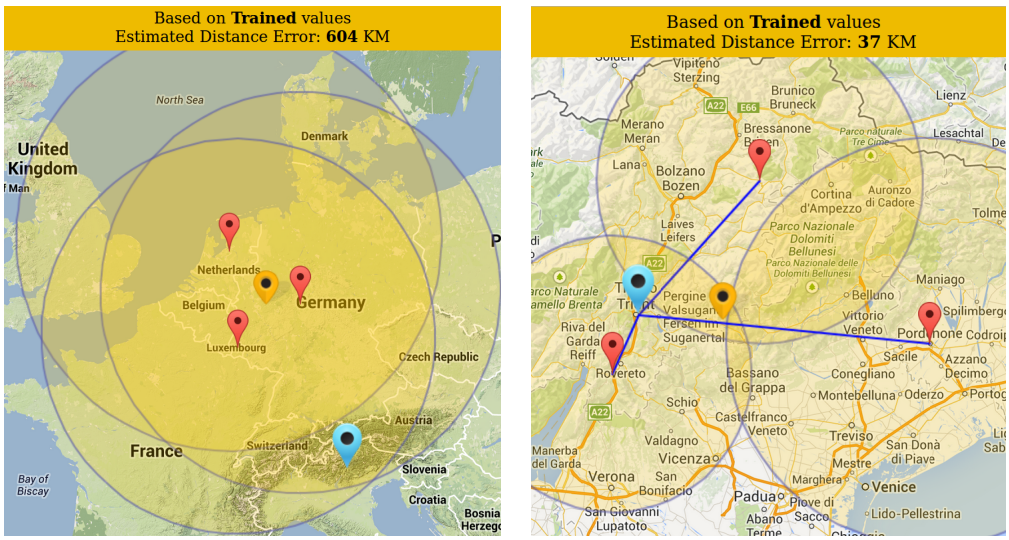


Figure 3.6: A comparison of various distance estimations done by VLOC and its rivals. These results show the average of estimation error in various ranges.

of distance estimation and compares it with non-adaptive approaches.

Once the estimation function is constructed, VLOC will be able to verify the physical location of the current machine. In this stage we provide the accuracy of geolocation estimation. The location of landmarks makes a significant impact on the accuracy of localization in triangulation technique. Figure 3.7 illustrates this impact, which shows the current server needs to be surrounded by the chosen landmarks and using randomly chosen landmarks. Randomly chosen landmarks do not guarantee the best accuracy. According to this fact, we performed the experiment of localization estimation in two fashions including randomly chosen landmarks and optimized ones. The results of these experiments are depicted in Figure 3.8. These results reveal that as the range of operation increases, the optimized chosen landmarks outperform the randomly chosen landmarks.

According to the results shown in Figure 3.8, the best result for geolocation estimation is obtained in range of 150 KM in which 162 landmarks are participating. As the range of the operation grows, the accuracy of the location estimation falls down. In order to utilize triangulation technique in larger ranges, we need to draw larger circles which increases the risk of estimation error. Thus, the landmarks situated in nearby are the best for



(a) An example of extremely bad chosen landmarks.

(b) An example of desirable chosen landmarks.

Figure 3.7: Two observations of randomly chosen landmarks which can be perfect or can give very different location estimation. The light red markers show the locations of the selected landmarks, the blue marker is the current host (Trento), and the yellow marker points to the estimated physical location of current host.

our purpose.

In VLOC, various factors impact on the accuracy represented by the following statement:

$$Acc \propto \frac{P \times C}{F} - \| \frac{d}{dR} f(R) \| \tag{3.11}$$

where Acc is the accuracy, P is the frequency of performing RTT latency measurement in order to keep an updated observation of the network latency, C is the confidence factor used in Algorithm 2, F refers to the network fluctuation which is obtained by calculating the latency differences of a number of HTTP requests transmitted between two identical hosts. $f(R)$ is a function representing the changes of accuracy based on changes of range of operation, R. Small values of R (*i.e.* less than 100 KM in our experiments) do not yield an accurate result because the number of

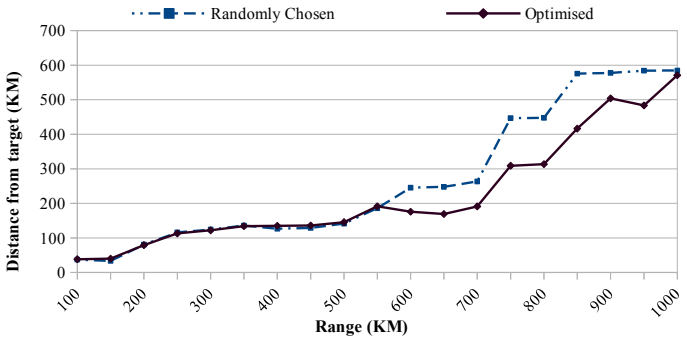


Figure 3.8: Estimation error in localization per various ranges. This figure depicts the results in two landmark selection styles which are optimized and random selection.

landmarks in small ranges are not sufficient. On the other hand, as Figure 3.8 shows, there is an optimum point for R and increasing this value after that point makes a negative impact on the accuracy. Therefore, in this statement, the derivative of such a function is used.

3.5 Chapter Summary

This chapter presents an approach, named VLOC, for verifying the physical location of a virtual machine without using a network of fixed external landmarks nor a GPS enabled device. VLOC is implemented as a software which is able to estimate the physical location of itself and notify the corresponding user if the location is unauthorized. It allows a user to install it on a virtual machine and after initialization it will be ready to be practically used.

VLOC works inside of the target host (inside of the cloud) and does not rely a network of fixed external landmarks; therefore, the implementation cost is quite negligible. All a user needs to do is to install it as a tool on his/her virtual machine and then initialize it. However providing a geolocation service by using a tool installed inside the cloud while the cloud provider is the major adversary brings an important security issue.

Since cloud provider has control over the infrastructure, platform, and the network, he is able to modify the real measurements with fake information. Our strategy against such an attack is to use random websites as external landmarks and obfuscate our messages into a regular protocol such as HTTP. In this scenario, there is a significant cost for the cloud provider to filter the network traffic and modify the information.

The experimental results demonstrate that VLOC is accurate enough for being used in practice. Moreover, it can be integrated into a monitoring framework in order to track a piece of data or into a policy enforcement engine as a policy information point in XACML architecture [67].

Chapter 4

Distributed Auditing for Data Location Compliance in Cloud

In this chapter we introduce a framework, named DLoc, which enables the end-users to track the location of their data after being transferred to the cloud. DLoc does not require a network of monitoring servers (landmarks) and does not need to reside and/or run within the cloud. It uses a proof of data possession technique to guarantee that the cloud storage service possess the particular file and estimates its location(s) in a distributed manner without requiring the collaboration of the data controller or cloud provider.

4.1 Introduction

Steadily increasing data volumes and the rising dependency of business and social life on data ubiquity have led to massive growth of cloud storage services such as Amazon S3, DropBox, or Google Drive. These services allow users to store their data on remote servers independently of geographical location. Cloud storage services utilize a federation schema by maintaining data at different providers which then distributes and replicates the data among different cloud storage providers. This reduces vendor lock-in and

increases data availability through additional redundancy.

Applying such federation schema can raise issues with compliance requirements. Especially the transparent data distribution and replication on the provider-side limit the user's direct control over data flows which lead to potential violations of compliance constraints. Personal data, for instance, sometimes must not leave a particular jurisdiction while the distribution in such a case is reasonable in terms of availability, it clearly can violate privacy compliance regulations such as the EU Data Protection Regulation [71]. Russia [1] and China [2] are also imposing restriction on the location of the data processing.

The approaches introduced to track a file in cloud are divided into two major groups. The first group propose a schema requiring modification of underlying cloud services and collaboration of cloud service providers. The second group observes the environmental parameters (from outside of the cloud) in order to estimate the location of a file in cloud. The parameters include network delay, hop counts, mode of delay, median of delay, standard deviation of delay, and population density. We used some of these features in our previous work, VLOC.

The second group has a clear advantage since it does not require modifying the underlying services; however, they require a wide spread network of servers communicating to each other, pinging cloud storage servers and monitor their data transfer practices. Having such network brings a significant cost to the tracking service. In VLOC, we designed and implemented a technique which monitors the dynamics of the network delay of the cloud service and builds a model out of it and keeps updating the model. It does it through measuring RTT delay from servers which have two major characteristics: a) they are chosen randomly, so cloud provider is not able to

[1] https://techcrunch.com/2016/11/17/linkedin-is-now-officially-blocked-in-russia/
[2] http://www.bbc.co.uk/news/technology-40106826

filter them; b) their physical location is known to VLOC. VLOC needs to be installed on a virtual machine and be initialized with the actual location of the data center; therefore, it can be used by data controllers to monitor and verify the location of their virtual machine in cloud. Data owners need to find the location of their data in cloud and VLoc does not provide such service. In fact, we need a technique which does not require the collaboration of cloud provider or data controller in order to monitor the location of data from a client machine.

In this work, we propose a framework, named DLoc, which does not require a network of monitoring servers and does not need to reside within the cloud. The idea is to distribute the monitoring tasks to DLoc agents. Each user who subscribes for the file tracking service participates in the file tracking procedure as a DLoc agent by letting her phone to challenge the cloud storage services and share her coarse-grained location with our service. DLoc makes use of proof of data possession technique to guarantee that the cloud storage service possess the particular file in question and estimates the location of all copies of files publicly available in the cloud.

The major challenge is to minimize the number of messages going to and coming from the DLoc agents while maximizing the accuracy of location estimation. It achieves that by observing the environment and studying the algorithms used in the system and provide a measurement to evaluate the accuracy and performance.

4.2 *DLoc*

In this section, we explain how **DLoc**, Distributed Data Localization framework, works. Figure 4.1 illustrates a general overview of DLoc and the major steps required to track a file in cloud. There are four major entities:

- ***Data owner*** wants to upload a file into *Cloud Storage B* which is

located in her country. She wants to assure that her file stays in that region.

- **Cloud Storages** are the storage services used as backup storage and file sharing platform.

- **DLoc agents** are actually other smartphone users who use cloud storage services as well. They challenge a given target server and collect network latency information.

- **TPA** is a third party auditor server, which coordinates the DLoc agents and handles the file tracking procedure.

The data owner runs Algorithm 3 on her phone to upload the file. This algorithm encrypts the given file with an encryption key generated by the user's device. Then, it produces a set of meta-data required by DLoc to track the file securely.

Algorithm 3, first, encrypts the given file (F) with the input key (k). Then, it generates the hash value for each block of the encrypted file (h_i). The next step is to compute MAC (Message Authentication Code) for each of hash values of the blocks (m_i) with a randomly generated key. Please note that this random generated key is the same for all blocks of the given file (M_{sk}). The encrypted file (F_c) is uploaded to the cloud and the MAC values ($M = \{m_1, m_2, \ldots, m_n\}$) along with the MAC key (M_{sk}) are published to the TPA. This procedure is shown by ① in Figure 4.1.

When the data owner wants to track her file in the cloud, she queries the TPA (shown by by ② in Figure 4.1). The TPA runs the Algorithm 4 which receives the list of DLoc agents (S), the number of required challenges (c) and the file identifier (F_{id}) which specified in the query coming from the data owner; and then it generates a list of challenge requests (R). The parameter S does not contain all the DLoc agents rather a selected subset

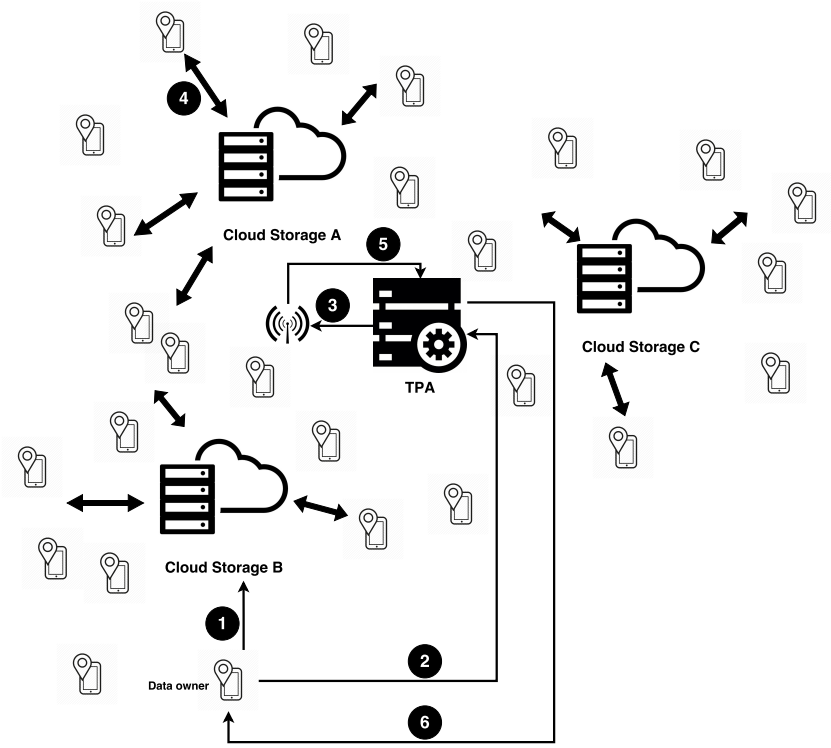


Figure 4.1: System Overview of DLoc.

Input: F: input file; k: encryption key;
Output: F_c: encrypted file; M: set of MAC codes for F_c blocks; M_{sk}: MAC encryption key;

F_c = Encrypt(F, k);
M_{sk} = ***new*** RandomKey();
B_{Fc} = F_c.getBlocks();
for *(* b_i *in* B_{Fc}*)* **do**
    h_i = Hash(b_i);
    m_i = MAC(h_i, M_{sk});
end
$M = \{m_1, m_2, \ldots, m_n\}$;
return $\{F_c, M, M_{sk}\}$;

Algorithm 3: The data owner runs this algorithm on her phone in order to prepare the file for upload and provide required metadata for the tracking procedure.

of them. The selection policy is based on their availability, location, and the number of requests they have performed already. The number of challenges, c, is tunable; as its value grows the accuracy and also the overhead. The algorithm, first chooses c random blocks (B) of the file, then chooses a random member of DLoc agents and assigns a random block to it $\langle s_i, r_i \rangle$. In this setting, it is possible that a block is requested more than once and a DLoc agents receives more than one request. Finally the TPA sends each request to its corresponding DLoc agents, which is indicated by ③ in Figure 4.1.

Input: F_{id}: file identifier; S: list of DLoc agents; c: number of challenges;
Output: R: list of challenge requests;

1 $n = F_{id}$.getNumberOfBlocks();
2 $B = \{b \in \mathbb{N} | (b_i = RandomNumber(0, n)) \overset{c}{_{i=1}} \}$;
3 $R = \{\}$;
4 **for** *(b'_i in B)* **do**
5 $s_i = S$.getRandomMember();
6 $r_i =$ ***new*** Request(F_{id}.getURL(), b'_i);
7 $R \leftarrow \langle s_i, r_i \rangle$;
8 **end**
9 **return** R;

Algorithm 4: Preparing the challenge messages for broadcasting to the DLoc agents.

When a DLoc agent receives a challenge request from the TPA, it performs the Algorithm 5 shown by ④ in Figure 4.1. This algorithm receives a set of requests (R'), challenges the server and provides challenge results (CR) to the TPA. This algorithm has two major tasks. First, to challenge the server whether it possesses the file or not. Second, to estimate the distance between the server and the DLoc agent in order to provide information for distributed localization. In order to challenge the server for the file possession, Since each request (r_i) contain a file block number, this algorithm queries the server for that particular block, then it

computes the hash value of the block (h_i). In the meanwhile it measures the download time (Δt_i) that will be used later for distance estimation. The hash values and the measured round trip time (RTT) of the challenge request are sent to the TPA for further analysis.

In Chapter 3, we demonstrate that there is a direct correlation between RTT and physical distance; however, there are a number of parameters involved which affect the accuracy. The major issue is that DLoc agents are located in various locations and use different network bandwidths. In order to mitigate the effect of the network variety, we take two approaches. The first one is to use an off-the-shelf technique to observe the DLoc agent network connection before challenging the server (N_{speed} in Algorithm 5). It uses an API provided by `http://www.speedtest.net` which encompasses a network of servers around the globe and finds a nearby server and communicate a number of packages, then it provides an observation on the network performance. The results of this API are useful to tune the weights of the RTT values in order to mitigate the effect of different network bandwidths on the estimation procedure. Moreover, due to network load, the traffic goes through different paths which causes various network delays; therefore, it lowers the accuracy of distance estimation based on RTT. In order to tackle this issue, we employ a machine learning technique, to tune the wights of parameters and adapt the estimation to the network fluctuation which is indicated by "$NormalizeNet(N_{speed}, \Delta t)$" in Algorithm 5.

Finally, each DLoc agent sends a set of the hashed value of each block (H), normalized network measurements (N) and its location (L) to the TPA (⑤ in Figure 4.1). The TPA collects all the information from the DLoc agents and carries out two tasks: it verifies the challenges by running Algorithm 6 and determines the location of the server by executing Algorithm 7 which is discussed later.

Input: R': subset of challenge requests;
Output: CR: challenge results;

1 $H = \{\}$; `//Hash values of the blocks`
2 N_{speed} = Network.AnalyzeSpeed();
3 **for** *(r_i in R')* **do**
4 $u = r_i$.getURL();
5 $x = r_i$.getBlockNumber();
6 t_{start} = Now();
7 b_i = **download** file-block #x from u;
8 t_{end} = Now();
9 $\Delta t_i = t_{end} - t_{start}$;
10 h_i = Hash(b_i);
11 $H \leftarrow \langle x, h_i \rangle$;
12 **end**
13 $N = NormalizeNet(N_{speed}, \Delta t)$; `//Network Measurements`
14 L = DeviceLocation();
15 **return** $\{H, N, L\}$;

Algorithm 5: Challenging the server.

Since the TPA possesses the MAC values of all blocks and their key (M_{sk}), by receiving the hash value of each block (h_j) is able to verify its integrity. Algorithm 6 receives the file identifier (F_{id}) and all collected challenge results and evaluates them. Each challenge result consists of a pair of block numbers and its hash value $\langle x_j, h_j \rangle$ which is retrieved from the server by a DLoc agent. This algorithm first computes the MAC value of the block number specified in the challenge result (m'_j) then compares it with the already stored MAC value for the same block (m_j). By doing the same procedure for all the received challenge results, we can verify the integrity of the stored file with a certain level of confidence. The confidence level depends on the number of challenge requests which may cause overheads.

Input: F_{id}: file identifier; CR: challenge results;
Output: verification result;

```
1  M_sk = F_id.getMACkey();
2  for ( c_i in CR) do
3      H_i = c_i.HashValues();
4      for ( ⟨x_j, h_j⟩ in H_i) do
5          m'_j = MAC( h_j, M_sk);
6          m_j = F_id.getMACValue(block# = x_j);
7          if ( m'_j ≠ m_j) then
8              return "Integrity Error!";
9          end
10     end
11 end
12 return "Verified!";
```

Algorithm 6: Verifying the challenges.

4.2.1 Estimating the data location

Algorithm 7, named the localization algorithm, uses Equation 3.5 and estimates the location of the data based on a set of given challenge re-

sults (CR). Each challenge result contains the location of the DLoc agent (center of the circle) and network measurement information to compute the its distance from the server (the radius of the circle). This algorithm estimates at least one location for the data. As the cloud storage provider might create multiple copies of the data on various servers, this algorithm handles this matter as well by determining the locations of all accessible copies of data.

The localization algorithm, first, creates an empty list of points (P). Then, for each given challenge result, it computes the distance from server by calling `Distance()` function. This function basically models the correlation between network delay and distance using a polynomial regression function, which is employed by VLOC as well [33]. The next major step is to calculate the intersection points of the circle of the current challenge results with the results received from the other DLoc agents. Then it verifies the circles and drops the ones which are not useful for localization according to the conditions mentioned above. There is an exception to this. Since in practice there is always a negligible error in distance estimation, sometimes the circles are close to each other but just for few meters, they do not match the condition. In order to overcome this issue, we define an error tolerance range parameter (ε) to compensate the error. The algorithm finds all intersection points amongst all the given circles and keep them in the P list. At the end, it determines the popular ranges (F_L) in which a considerable number of points are estimated. These popular ranges indicate the location of servers storing the data.

4.3 Empirical Evaluation

To evaluate DLoc we run experiments on 4 android devices (playing the role of servers) situated in four cities and in total, $1,422$ web hosts playing

Input: CR: challenge results; ε: error tolerance range;
Output: F_L: locations of the file;

```
1  P = new List();
2  for ( c_i in CR) do
3      d_i = Distance( c_i.Net()); //Network Measurements
4      l_i = c_i.DeviceLocation();
5      for ( c_j in CR ∧ j > i) do
6          d_j = Distance( c_j.Net());
7          l_j = c_j.DeviceLocation();
8          d_ij = ‖l_i − l_j‖; //Euclidean distance
9          if ( d_ij < √((d_i − d_j)^2) ) then
10             continue; //Ignore j
11         end
12         if ( d_ij > d_i + d_j ) then
13             if ( d_ij > d_i + d_j + ε ) then
14                 continue;
15             end
16             inc d_i, d_j until d_ij ≤ d_i + d_j;
17             report "ε is used";
18         end
19         (p_1, p_2) = IntersectPoints(c_i, c_j);
20         P ← p_1;
21         P ← {p_2|p_1 ≠ p_2};
22     end
23 end
24 F_L = A set of the most popular ranges in P;
25 return F_L;
```

Algorithm 7: Localization procedure.

as DLoc agents. As we measure the round trip time value (RTT) such role changing does not influence the final result.

We designed and implemented an android app to challenge the servers and collect network delay measurements between each DLoc agent and the nearby smartphone. Please note that in order to avoid confusion, we use the same terminology that we have explained in the approach. In other words, in the data analysis we do not consider this role changing.

This section explains the data collection process and describes the evaluation measures, the experimental results and their analysis.

4.3.1 Dataset Collection

In our settings there are four servers located in Trento, Turin, Eindhoven, and Leuven and there are many DLoc agents challenging them. We partially used the data collected in [33] including the address and location of numerous landmarks situated near the mentioned cities. Each DLoc agent challenges the server by utilizing an HTTP request for over 15 times a day and measures the RTT values of each challenge. In the following sections we analyze the data collected by DLoc agents and study the effect of various factors on the final results.

4.3.2 Evaluation Goals

The experiments are designed to answer the following research questions:

- **RQ1 Accuracy:** How accurate is DLoc to estimate the location of server hosting the file?
- **RQ2 Environment:** What are the parameters, such as number of DLoc agents, distance of the agents from the server, etc., influencing the accuracy of DLoc?

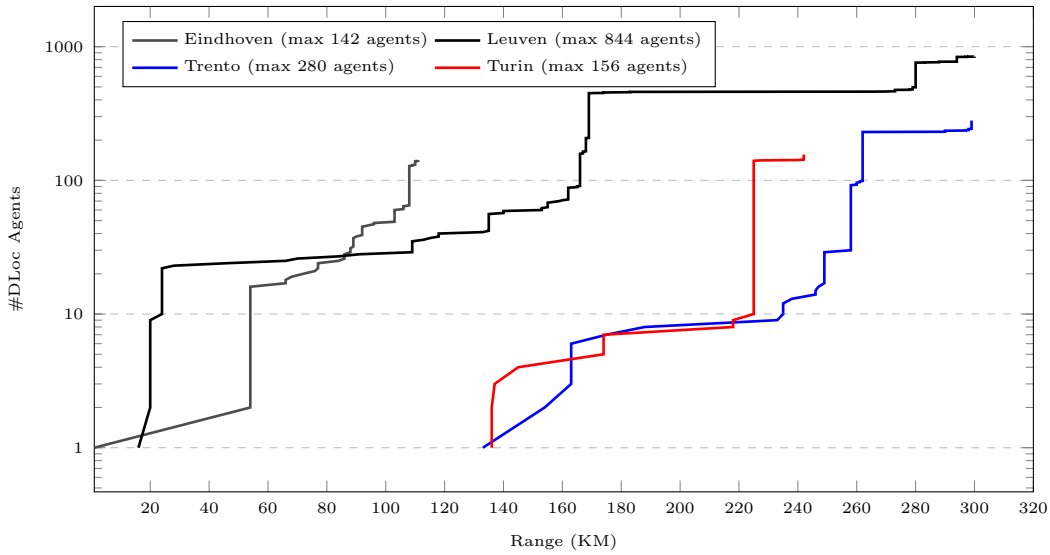


Figure 4.2: The number of DLoc agents per range.

4.3.3 Evaluation Measure

As the main task of DLoc is to determine the location(s) of an uploaded file (data) in the cloud, to evaluate it, the distance between the actual physical location of the machine where data resides and its estimated location. We used error of average distance estimation defined in the Equation 4.1 for accuracy evaluation.

$$E_{avg} = \frac{1}{N} \sum_{i=1}^{N} \|p_e^i - p_o^i\| \tag{4.1}$$

where N is the number of executions of the test, p_e^i denotes the estimated physical location of the server (cloud storage) in the i^{th} test p_o^i is the observed location (the real physical location) for that server. Finally, E_{avg} refers to the calculated average error in KM.

We evaluated the approach with challenges generated by various DLoc agents that spread widely around the servers. In order to provide a comprehensive assessment we apply a random value combination. We used a slightly modified version of cross validation technique [42] to perform such

a it. We trained the system with 80% of the data and test with the remaining 20%. The operation was repeated for a 1000 times by shuffling the data each time. Moreover, we used the same setting for all 4 different servers situated in 4 cities.

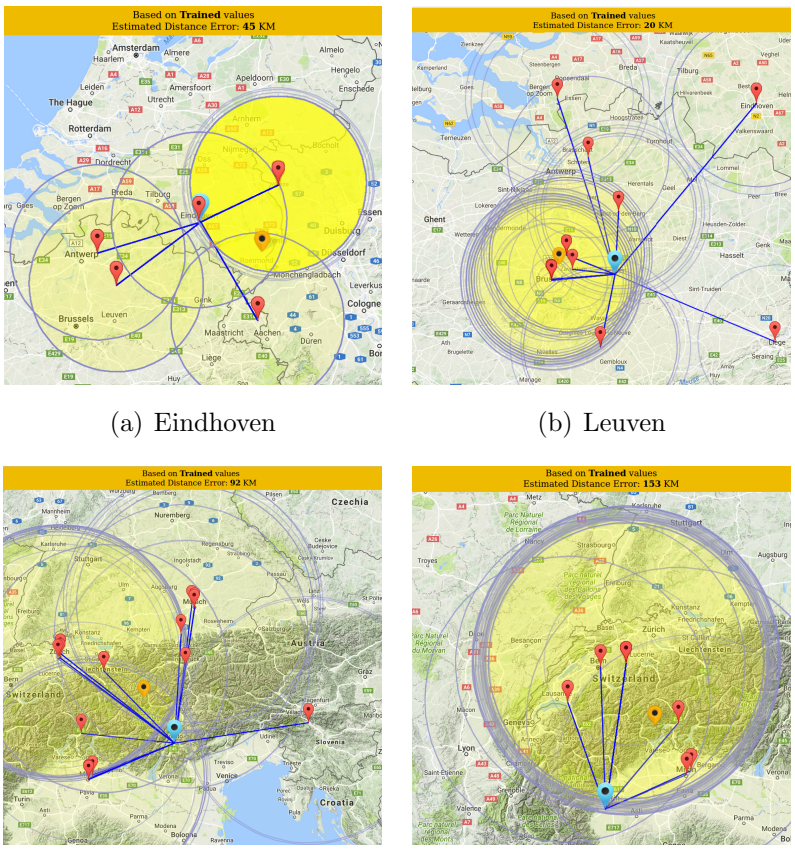
4.3.4 Results and Discussions

The results of the experiment composed of 1000 runs in each of the four cites, i.e., Trento, Turin, Eindhoven, Leuven, are aggregated in Figure 4.3 (a-d), respectively. Figure 4.3 shows the actual location of the file hosting cloud servers, location of the surrounding DLoc agents and the location of the server hosting the file as estimated by DLoc. On the map shown in the figure, the red, yellow and blue markers represent the actual location, the estimated location and the location of the DLoc agents. As shown in the figure, DLoc estimates the location of the server with a reasonable degree of accuracy, i.e., the estimated location is within 92 KM of the actual location for the Trento node, 153 KM for the Turin node, 45 KM for the Eindhoven node and 20 KM for the Leuven node (RQ1). Table 4.1 summarizes the results for each city.

	Eindhoven	**Leuven**	**Trento**	**Turin**
Min error (KM)	22.25	16.54	70.06	134.38
Max error (KM)	47.39	33.92	219.32	295.54
Standard deviation (KM)	2.22	2.02	30.67	30.54

Table 4.1: The summary of the results for each city.

Figure 4.4 illustrates the average error of the location determination for various number of challenges. As the results in this figure show, for the servers located in Eindhoven and Leuven, increasing the number of challenges does not have a significant impact on the estimation error while for the other two servers specially Turin, a notable change can be observed. While one of the reasons is the sparsity of DLoc agents around each server,



(a) Eindhoven (b) Leuven

(c) Trento (d) Turin

Figure 4.3: Screen shots of DLoc estimating a file on the four servers situated in multiple cities. The light red markers show the locations of the DLoc agents, the blue marker indicates the actual location of the server, and the yellow marker points to the estimated physical location of the server.

we also study the influence of distance, between the agents and the server, on the accuracy.

To study the influence of distance on the accuracy of location estimation (RQ2), we unitized the distance into multiple ranges and performed experiments on all DLoc agents situated only in each individual range. Figure 4.5 illustrates the results of such experiment. As it shows in the range of $20 - 40$ KM, only in Leuven there are a number of agents surrounding the server and sent 320 challenges while there is no agent until the range of $140 - 160$ KM where the number of challenges increased and the ac-

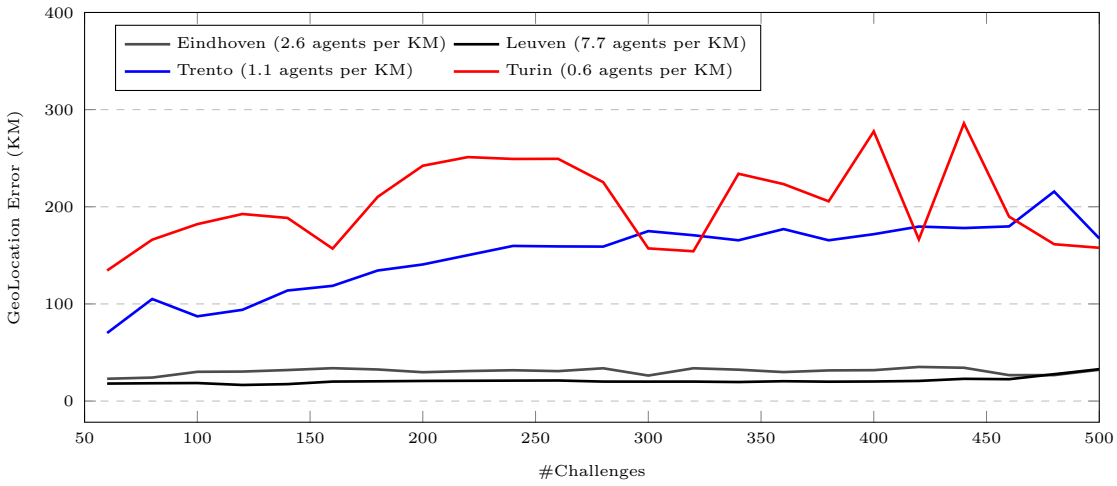


Figure 4.4: GeoLocation error estimation per various number of challenges.

curacy slightly declined due to the distance. The similar steady move is observed for Eindhoven for the ranges of $60-80\ KM$, $100-120\ KM$ and $140-160\ KM$. Turin and Trento have a bit different story; their closest agents are in the ranges of $140-160\ KM$ and $200-220\ KM$ respectively. Moreover, the number of challenges in these ranges are quite small (20 and 85) compare to what the servers located in Eindhoven and Leuven experience in their closest range. These two reasons explain the yielded lower accuracy for Turin and Trento. Therefore, not only the number of DLoc agents influences the accuracy, also their distance from the servers has a notable effect. Which means, the closer to the server the agents are, the less number of challenges is required to track data effectively.

It worth to mention that the obtained results even for Turin and Trento are acceptable as the main usage of DLoc is to monitor the enforcement of jurisdiction regulations which, at its finest granularity, limits the data to boundaries of a country. Moreover, it can have other usage as well including quality of service measurement.

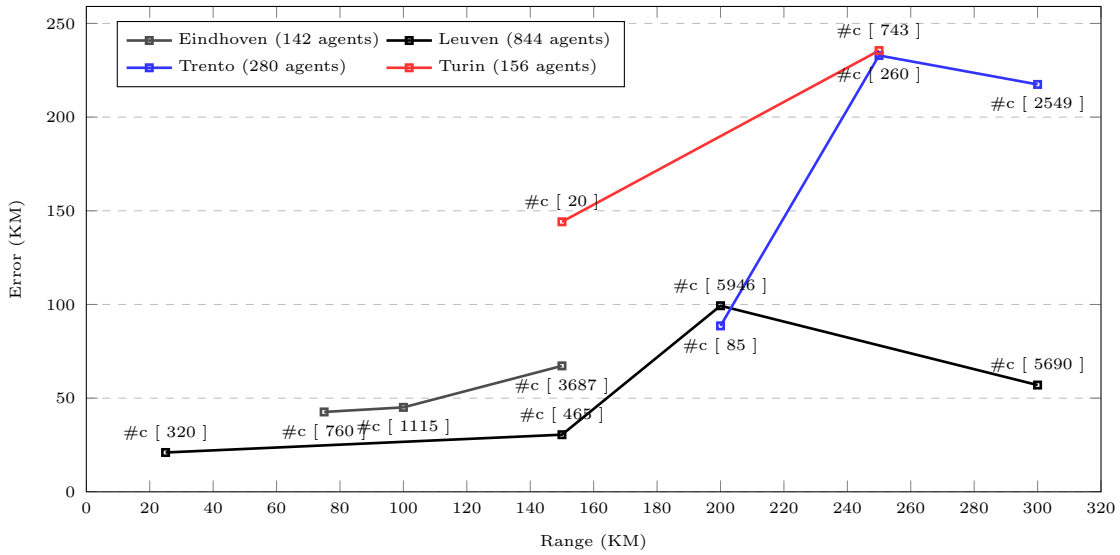


Figure 4.5: Location estimation error per individual ranges. "#c" denotes the number of challenges sent from the DLoc agents in the corresponding range.

4.4 Security and privacy analysis

DLoc does not require the cloud storage provider (CSP) to modify their systems. When DLoc is running, all the CSP can realize is that a user shares her files with a number of other users and the other smartphones (DLoc agents) participating in geolocating the file are chosen randomly and can be anywhere near the server or elsewhere; therefore, the CSP is not able to impose a fake delay on the responses of the challenges it receives.

In order to bypass DLoc, there are two possible scenarios. The first one is to break into the TPA which handles the challenges, DLoc agents and prepares the results. The second scenario is to register a huge number of smart devices (DLoc agents) in the TPA and make them to collude with each other to prepare a fake delay time and fake location. Both scenarios are quite expensive for the CSP to perform. Therefore, the cost of compliance is negligible compare to bypassing DLoc.

4.5 Limitations

Although DLoc is promising in a real world application, there are a number of limitations need to be considered. The first limitation is dependency to the number of DLoc agents and their distance to the target server. If there is not enough DLoc agents in less than about 400 KM of a server, the accuracy of DLoc will fall down.

Moreover since the cloud provider is considered as an adversary, it can inject random delays to the outgoing traffic to reduce the accuracy of DLoc. As the location of DLoc agents are considered as trusted, such random delay can yield different measurement by each agent and increase the estimation error. However, the strength point is that by doing so the cloud provider practically cuts its quality of service; therefore, abusing such limitation is costly for the could provider.

In practice, integrating DLoc to the current cloud storage providers (*e.g.* Amazon, Google drive, etc.) without modification of their systems imposes another limitation on the user. In fact, the data owner must share the file with the other users (giving them access to the file), even if it is encrypted for them to challenge the servers, it is still a limitation.

4.6 Related Work

In the literature, there are a number of approaches to determine the location of data in cloud. Some focus on providing a cloud infrastructure which is able to handle the enforcement of data location policies which certainly require hardware and/or software modification in cloud services. Recently studies have drawn their attentions to finding the correlation between the network delay and geographic distance which then can be used to determine the location of an Internet node. Here we review both groups briefly

with more emphasis on the second group as its more close to our work.

4.6.1 Server side data geolocation

Krau and Fusenig propose an approach utilizing a Trusted Platform Module (TPM) on host platforms for data geolocation in clouds [58]. They assume that a certification authority stores the location of a host with its TMP's identity. Then, the owner of a virtual machine requests a certification of the host in order to transfer data. They also assume that all virtual machines implement a "LocCheck" client which is able to communicate with a "Location verification and integrity check" module implemented in the hypervisor. This solution requires administrative methods to perform the verification of the location. Moreover, it is costly and due to the variety in cloud platforms, it is not able to prevent data replication in arbitrary locations.

Paladi et al. introduce a high-level architecture in cloud storage systems for a trustful location-based mechanism for data transfer control [70]. Fu et al. use a TPM to provide a strong validation of cloud data geolocation [35].

These approaches require the modification of underlaying layer of cloud services which are quite costly and difficult to be adopted by cloud providers.

4.6.2 Delay based data geolocation

Geoping assumes that the hosts with a similar network delay are at the same location [69]. Basically, Geoping challenges the target server from a number of known landmarks and builds a set of path-delay information. To find the location of an unknown target server, it constantly pings the server from the landmarks at known paths and uses Euclidean distance and finally chooses the landmark with the best match.

Constraint-based geolocation employs multilateration, which is used by

DLoc as well, where each landmark draws a circle around itself with a radius of the distance to the target server [40].

Topology-based geolocation improves the accuracy of Constraint-based geolocation by taking into account the network topology and using the relationship between latency and distance in order to estimate the distance between two Internet nodes [56]. It basically transforms the geolocation problem into a convex optimization problem with constraints and solves the problem to find the location of the target. A more recent work [18] enhances the same concept for geolocation which attempts to find the nearest common router and related landmarks to the target node, introduce deviations to landmarks and consider their locations as areas, calculating relative delay between landmarks and target node. Then it defines the distance as constraint conditions to estimate the real deviations of landmarks and the location of the target.

Yong et al. introduce a three layer geolocation algorithm, which employs a large database of landmarks, their relative distances and delay measurements [83]. The first layer is basically a constraint-based geolocation algorithm which finds the area where the target server is situated. The second layer employs the distance constraint-based method and uses the primary landmarks in the area to shrink the possible area. Then, the target is mapped to a near landmark discovered in the secondary area.

Benson et al. discuss how to determine the location of data in cloud storage which spread in diverse locations [12]. They assume that the locations of all data storages are known, that the cloud provider has no exclusive Internet connection between the data centers and that for each data center, there is a trusted third party node located geographically close to it. The proposed approach uses a distance measurement between the data centers to determine the location of the data centers where the user's data is stored. They evaluated their work in Planet Lab network. However,

the assumptions are too strong or expensive for a real world application. Furthermore, the work does not recognize multiple copies of data.

In order to reduce cost, *IGOD* selects a small subset of landmarks with their optimal position based on the diversity parameter [51]. Although the authors even achieved a better accuracy compare to the similar previous works, it still needs a network of fixed landmarks (*e.g.* Planet Lab) which is difficult to implement in practice.

Watson et al. demonstrate that verifying the location of data in a cloud storage has a limited accuracy [84]. They show that a collusion of the cloud provider with a number of malicious host makes it impossible for users to verify the location of their file accurately. The main drawback of this approach is that it requires a set of trusted landmarks exists in order to verify the existence of a file on a host.

GeoProof combines a proof of retrievability scheme with a delay based protocol to determine the distance between a host and a verifier [3, 4]. They assume a tamper proof GPS device in the local network of cloud provider communicating with a third party to verify the location of data. The major drawback of this protocol is that cloud providers are not willing to have a black box attached to their local network. Moreover, the GPS signals received by the device can be faked by a malicious cloud provider.

Gondree and Peterson proposed a schema to tackle such problem by employing a latency function built based on the current network traffic observation [37]. In their schema, there are a number of trusted landmarks which observe the network traffic by transmitting a number of messages amongst themselves and then build a model based on that. The main disadvantage of this approach is the requirement of a dedicated network of landmarks which is quite costly. Moreover, in the model building phase the landmarks send messages amongst themselves in order to find a baseline for the Internet delay which does not quite represent the real environment.

In fact, this scenario does not consider the latencies imposed by cloud mediation services such as authentication, decryption, etc. Therefore, the observation has an inherent limitation which influences the distance estimation.

Abdou et al. show that having a fixed network of landmarks can be manipulated [1]. The location of landmarks will be revealed over time and since usually delay based approaches use UDP or ICMP protocols, an adversary is able to filter them out and play with the delays of the responses in order to misrepresent its own location. DLoc does not suffer from this issue due to two reasons: a) The location of landmarks used by DLoc is not fixed as the landmarks are portable devices. b) DLoc uses the same protocol that the cloud storage provider uses to serve the clients and basically an adversary is not able to differentiate between the real user and a landmark.

There is a parallel work with DLoc which uses network delays and a network of smartphones to estimate the physical location of a server [21]. However, the focus of DLoc is to estimate the location of data (*e.g.* a file) in the cloud. It verifies the server for the possession of user's data and tracks all available online copies of the file on all servers. Moreover, the best error rate reported in their study is 189 KM while the average error rate for DLoc in Leuven and Eindhoven is less than 50 KM and for Turin and Trento less than 150 KM. DLoc proposes a comprehensive framework which adapt itself automatically by observing the environment and remove noisy data.

4.7 Chapter Summary

This chapter introduces DLoc, which determines the location of a file transferred to the cloud. which determines the location of a file transferred to

the cloud. It uses a proof of data possession technique to guarantee that the cloud storage service possess the particular file and estimates its location(s) in a distributed manner without requiring the collaboration of the data controller or cloud provider. DLoc has a number of advantages compare to its rivals. First, it does not require a dedicated network of trusted landmarks which makes it quite economic to be used in a real world setting. Second, it does not require a modification to the cloud services. Third, it is able to deal with multiple copies of data. Fourth, employing machine learning techniques has made DLoc robust against network fluctuations and various types of connections. Finally, since it uses smartphones instead of fixed landmarks, it has motivation for DLoc agents to use the service and participate in the process.

In a real-world scenario where DLoc serves a huge number of smartphone users, therefore it is able to find the locations of data centers precise enough in order to report all the data centers in the world representing a physical risk to all cloud providers. Moreover, since DLoc provides measurements and statistics on where data is stored and how long does it take to be delivered, it can be used to measure the quality of service for content delivery to mobile users and help to improve it.

Chapter 5

Migration to industry, standardization bodies and open sources communities

In this chapter we discuss the relevance, effect and the contribution of our work to industry.

5.1 Introduction

Demonstrating compliant privacy management practices has never been this relevant for businesses. The EU General Data Protection Regulation (EU GDPR) imposes considerable fines (up to 4% of the global annual turnover) to organizations that fail to comply with the new framework effective in May 2018. Two of the most important points of the regulation features are new responsibilities for data processors (who process personal data on behalf of other organizations) and new constraints of trans-border data flows, in particular with the emergence of a new EU-US agreement on the topic, the Privacy Shield, replacing the Safe Harbor agreement, which was invalidated in 2015 by the EU Supreme Court of Justice.

These two characteristic greatly impact cloud platform providers, such

as SAP. First, they most often play the role of data processor, directly or indirectly, since customers (the actual data controllers) subscribe to the cloud offers in order to reach their business goals. Second, the individuals from whom data is collected have a number of rights according to the regulation, including limitation of personal data across borders and/or to third parties.

Meanwhile, cloud services often rely on geographically distributed data centers across the globe for increased access speed and resilience. Moreover, cloud platforms are designed for extensibility – via mobile applications and further SaaS build over the platform services. The extensions are often driven by a network of partner solution providers. In many of the cases, they may also process personal data. In such a context, it is fundamental cloud providers and customers, to have tools to enforce and to monitor personal data flows in the cloud.

The work carried out in this PhD addressed questions related to data processing location in the cloud:

- **VLOC** (a Verifier for physical LOCation of a virtual machine), which is able to verify the physical location of a virtual machine by taking advantage of nearby randomly chosen web-servers. Since VLOC does not rely on a network of fixed landmarks, its implementation is easier and maintenance cost lower than other proposed solutions. VLOC is implemented as a software component which needs to be installed and initialized on a virtual machine. Moreover, it can be used as a background service of a virtual machine or a container.

- **DLoc** (Distributed auditing for Data LOCation compliance in cloud), which determines the location of a file uploaded in cloud and all its publicly available copies. Since DLoc employs the mobile devices as its agents to challenge the server and verify the possession and the

location of the file, it distributes the computation overhead and the cost amongst a significant number of mobile devices. Therefore, as there is no need for a huge network of landmark server, it is quite economic to implement. Moreover, another advantage of DLoc is that cloud storage service providers do not need to make any modifications in their systems.

- **PDTLoc** (Personal Data Transfer LOCation analyzer), which employs both static and dynamic analysis techniques to infer whether the apps violate DPD'25.1. PDTLoc inspects mobile applications and extracts information about the collected personal data and the jurisdictions of the remote servers to which the data is transferred. PDTLoc provides the capacity for treating a large number of applications on reasonably short time.

In the following we discuss the potential impact of these works from an industrial perspective.

5.2 Industry (SAP)

SAP not only provides software, but also hosts personal data collected by its customers in the cloud. One of the main decision factors for customers to adopt cloud services is trust. Exploiting VLOC in the context of SAPs cloud services would provide increased transparency to customers with respect to the data processing locations for SAPs data centers, those of its subsidiary companies (Successfactors, Ariba, Concur) and of its partners. More visibility on critical risk factors for customers is needed overall in order to acquire more cloud customers, especially those who are cloud averse nowadays.

Moreover, SAP also needs to assess the data transfer flows for its own ap-

plications. By demand of SAPs management, we run PDTLoc on all SAP official mobile apps (in total 60 apps in November 2016). We made the results available internally to the SAPs Data Protection Office. PDTLoc has potential for further exploitation as a testing framework for mobile applications, with a focus on information flow analysis, which is particularly suitable for asserting privacy compliance properties. This is interesting for SAP, who regularly publishes new mobile apps, or releases new versions of it. PDTLoc can also serve as a blueprint for large scale privacy analysis of applications in general. It can be well-suited for checking privacy features of third-party applications and platform services (e.g. provided by partners).

It is worthy mentioning that we also had a meeting with *CNIL*[1], the French National Commission on Informatics and Liberty, and presented PDTLoc tool and its capabilities in order to see how it can be used to protect the rights of citizens from their point of view. The constraint they have is that they are not able such technology to prove non-compliance in principle, because they need to prove we did not change the behavior of the apps. However, they can use this tool to detect the suspicious behavior of the apps in terms of data collection and data transfer which can be used to request the data controllers (the app owners) to provide compliance evidences.

5.3 Standardization Bodies

Standardization was not one of the goals for the current topic, however, the results are relevant for standardization organizations and for certification agencies. ISO and other bodies (ETSI, ENISA, and Cloud Security Alliance) are updating their standards and recommendations after the adop-

[1] `https://www.cnil.fr/en/home`

tion of the EU GDPR, often these are based on improvement life cycles that involve the adoption of tools. As a matter of fact, the EU GDPR suggests the creating of privacy certification seals and codes of conduct by industry associations and similar entities. The certification mechanisms and codes of conduct must be approved by a supervisory authority. On their turn, services and software will be awarded a seal if they prove adherence to the terms of the certificate in question. Tools like VLOC and PDTLoc can greatly accelerate the deliverance of such certification seals.

There is a sign of interest about such uses for this technology. After the appearance of the PDTLoc article we were contacted by one of the directors the French Data Protection Authority for presenting the work to their team.

5.4 Open-source Software

The prototypes developed here rely on a large number of open source components. Some examples are the APK tool, SAAF (Static Android Analysis Framework for Android apps) and Monkey (created by google). It may be interesting to consider to release to full frameworks as open sources components, if it fits SAPs exploitation strategy. This depends on potential emerging business model (for instance for partner app certification) and approvals from SAPs legal department.

Chapter 6

Conclusions and Future Work

In this dissertation, we introduce a substantial contribution in the analysis of trans-border personal data flows and enforcement of jurisdictional regulations over that. It is a major debate that may impact how the regulatory framework around the digital economy will evolve. We have highlighted the main concerns in personal data transfers by in principle non-malicious applications, and shown a considerable number of them fail to comply with the EU personal data protection regulation, in the first study of the kind, up to our knowledge. While PDTLoc has been suitable in this case, we believe it can be extended to analyze other information flow properties as well.

To address the issue of server side compliance, we design VLOC which verifies the physical location of a virtual machine without using a network of fixed external landmarks nor a GPS enabled device. VLOC is implemented as a software which able to estimate the physical location of itself and notify the corresponding user if the location is unauthorized. There are a number of approaches using distance bounding protocols to verify the physical location of a host, but as mentioned in Chapter 3, they are suffering from high implementation and maintenance cost or they are quite non-feasible or non-practical. In contrast, VLOC runs inside of the target

host (inside of the cloud) and does not rely a network of fixed external landmarks; therefore, the implementation cost is quite negligible. However residing within the cloud while the cloud provider is the major adversary brings a number of challenges which we have overcome.

In order to enable end-users to track the location of their data after being transferred to the cloud, we propose DLoc framework which does not require a network of monitoring servers and does not need to reside and running within the cloud. It uses a proof of data possession technique to guarantee that the cloud storage service possess the particular file and estimates its location(s) in a distributed manner without requiring the collaboration of the data controller or cloud provider.

In this thesis, we have proposed a bundle of approaches in order to enforce and monitor the jurisdictional regulations on mobile apps. Since collecting and transferring personal data to a remote server composes of multiple elements (*e.g.* mobile devices, mobile apps, data controller and cloud service provider), we introduced a bundle of tools and frameworks to monitor the data transfer in various levels. Implementing such bundle is quite economic compare to rival approaches; however, for some parts of the work, to best of our knowledge, there is no other approaches.

Bibliography

[1] AbdelRahman Abdou, Ashraf Matrawy, and Paul C van Oorschot. Accurate manipulation of delay-based internet geolocation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 887–898. ACM, 2017.

[2] Jagdish Prasad Achara, Franck Baudot, Claude Castelluccia, Geoffrey Delcroix, and Vincent Roca. Mobilitics: Analyzing privacy leaks in smartphones. *ERCIM News*, 2013(93), 2013.

[3] A Albeshri, C. Boyd, and J.G. Nieto. Geoproof: Proofs of geographic location for cloud computing environment. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 506–514, 2012.

[4] Aiiad Albeshri, Colin Boyd, and JuanGonzlez Nieto. Enhanced geoproof: improved geographic assurance for data in the cloud. *International Journal of Information Security*, 13(2):191–198, 2014.

[5] Alexa. Alexa.com, 2014.

[6] Tina Amirtha. Safe Harbor was for EU privacy: But how safe is US data in Europe? *http://www.zdnet.com/article/safe-harbor-was-for-eu-privacy-but-how-safe-is-us-data-in-europ* 2015.

[7] AppFigures. A tracking platform to monitor the sales and downloads of apps. *http://AppFigures.com*, 2016.

[8] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *ACM SIGPLAN Notices*, volume 49, pages 259–269. ACM, 2014.

[9] Monir Azraoui, Kaoutar Elkhiyaoui, Melek Önen, Karin Bernsmed, Anderson Santana Oliveira, and Jakub Sendor. *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance: 9th International Workshop, DPM 2014, 7th International Workshop, SETOP 2014, and 3rd International Workshop, QASA 2014, Wroclaw, Poland, September 10-11, 2014. Revised Selected Papers*, chapter A-PPL: An Accountability Policy Language, pages 319–326. Springer International Publishing, Cham, 2015.

[10] Michael Backes, Sebastian Gerling, Christian Hammer, Matteo Maffei, and Philipp von Styp-Rekowsky. Appguard–fine-grained policy enforcement for untrusted android applications. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 213–231. Springer, 2014.

[11] Walid Benghabrit, Hervé Grall, Jean-Claude Royer, Mohamed Sellami, Monir Azraoui, Kaoutar Elkhiyaoui, Melek Önen, Anderson Santana Oliveira, and Karin Bernsmed. *Cloud Computing and Services Sciences: International Conference in Cloud Computing and Services Sciences, CLOSER 2014 Barcelona Spain, April 3–5, 2014 Revised Selected Papers*, chapter From Regulatory Obligations to Enforceable

Accountability Policies in the Cloud, pages 134–150. Springer International Publishing, Cham, 2015.

[12] Karyn Benson, Rafael Dowsley, and Hovav Shacham. Do you know where your cloud files are? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 73–82. ACM, 2011.

[13] Sven Bugiel, Stephan Heuser, and Ahmad-Reza Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Usenix security*, pages 131–146, 2013.

[14] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 15–26. ACM, 2011.

[15] Johnathon Burket, Lori Flynn, Will Klieber, Jonathan Lim, William Snavely, Jonathan Burket, William Klieber, and Wei Shen. Making didfail succeed: Enhancing the cert static taint analyzer for android app sets, 2015.

[16] Mary Carolan. Data protection commissioner to investigate max schrems claims. *http://www.irishtimes.com/news/crime-and-law/courts/high-court/data-protection-commissioner-to-investigate-max-schrems-claims-2398728*, 2015.

[17] F. Di Cerbo, D. F. Some, L. Gomez, and S. Trabelsi. Ppl v2.0: Uniform data access and usage control on cloud and mobile. In *TEchnical and LEgal aspects of data pRivacy and SEcurity, 2015 IEEE/ACM 1st International Workshop on*, pages 2–7, May 2015.

[18] Jingning Chen, Fenlin Liu, Xiangyang Luo, Fan Zhao, and Guang Zhu. A landmark calibration-based ip geolocation approach. *EURASIP Journal on Information Security*, 2016(1):4, 2016.

[19] Shauvik Roy Choudhary, Alessandra Gorla, and Alessandro Orso. Automated Test Input Generation for Android: Are We There Yet?(E). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 429–440. IEEE, 2015.

[20] Fred Chung. Custom class loading in dalvik.

[21] Gloria Ciavarrini, Valerio Luconi, and Alessio Vecchio. Smartphone-based geolocation of internet hosts. *Computer Networks*, 116:22–32, 2017.

[22] European Commission. European commission - overview on binding corporate rules. *http://ec.europa.eu/justice/data-protection/international-transfers/binding-corporate-rules/index_en.htm*, 2016.

[23] European Commission. European Commission - press release: EU-US Privacy Shield. *http://europa.eu/rapid/press-release_IP-16-216_en.htm*, 2016.

[24] Court of Justice of the European Union. The court of justice declares that the commission's us safe harbour decision is invalid. *http://curia.europa.eu/jcms/upload/docs/application/pdf/2015-10/cp150117en.pdf*, 2015.

[25] Benjamin Davis, Ben Sanders, Armen Khodaverdian, and Hao Chen. I-arm-droid: A rewriting framework for in-app reference monitors for android applications. *Mobile Security Technologies*, 2012(2):17, 2012.

[26] AS. De Oliveira, J. Sendor, A Garaga, and K. Jenatton. Monitoring personal data transfers in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 347–354, 2013.

[27] Anthony Desnos and Patrik Lantz. Droidbox: An android application sandbox for dynamic analysis (2011). *https://code.google.com/p/droidbox*, 2014.

[28] Michael Dietz, Shashi Shekhar, Yuliy Pisetsky, Anhei Shu, and Dan S Wallach. Quire: Lightweight provenance for smart phone operating systems. In *USENIX Security Symposium*, volume 31, 2011.

[29] Serge Egelman, Adrienne Porter Felt, and David Wagner. Choice architecture and smartphone privacy: Theresa price for that. In *The economics of information security and privacy*, pages 211–236. Springer, 2013.

[30] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[31] Ericsson. Europe mobility report appendix. *http://www.ericsson.com/res/docs/2014/emr-november2014-regional-appendices-europe.pdf*, 2014.

[32] Mojtaba Eskandari, Maqsood Ahmad, Anderson Santana De Oliveira, and Bruno Crispo. Analyzing remote server locations for personal data transfers in mobile apps. In *International Symposium on Privacy Enhancing Technologies*. Springer, 2017.

[33] Mojtaba Eskandari, Anderson Santana De Oliveira, and Bruno Crispo. Vloc: An approach to verify the physical location of a virtual machine in cloud. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 86–94. IEEE, 2014.

[34] European Court of Justice. Commission Decision of 26 july 2000 pursuant to directive 95/46/ec of the european parliament and of the council on the adequacy of the protection provided by the safe harbour privacy principles and related frequently asked questions issued by the us department of commerce. *Official Journal L 215 , 25/08/2000 P. 0007 - 0047 URL:* `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000D0520:EN:HTML`, 2000.

[35] Dong Lai Fu, Xin Guang Peng, and Yu Li Yang. Trusted validation for geolocation of cloud data. *The Computer Journal*, page bxu144, 2014.

[36] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. *AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale*. Springer, 2012.

[37] Mark Gondree and Zachary N.J. Peterson. Geolocation of data in the cloud. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 25–36, New York, NY, USA, 2013. ACM.

[38] Google. Monkey Tool. `http://developer.android.com/tools/help/monkey.html`, 2015.

[39] Ben Gruver. Smali/Baksmali Tool. *https://github.com/JesusFreke/smali/wiki*, 2015.

[40] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions On Networking*, 14(6):1219–1232, 2006.

[41] G.P. Hancke and M.G. Kuhn. An RFID Distance Bounding Protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 67–73, 2005.

[42] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Model assessment and selection. In *The Elements of Statistical Learning*, Springer Series in Statistics, pages 219–259. Springer New York, 2009.

[43] Dominik Herrmann and Jens Lindemann. Obtaining personal data and asking for erasure: Do app vendors and website owners honour your privacy rights? *CoRR*, abs/1602.01804, 2016.

[44] Paul De Hert and Vagelis Papakonstantinou. The proposed data protection regulation replacing directive 95/46/ec: A sound system for the protection of individuals. *Computer Law & Security Review*, 28(2):130–142, 2012.

[45] Stephan Heuser, Adwait Nadkarni, William Enck, and Ahmad-Reza Sadeghi. Asm: A programmable interface for extending android security. In *USENIX Security*, volume 14, pages 1005–1109, 2014.

[46] Johannes Hoffmann, Martin Ussath, Thorsten Holz, and Michael Spreitzenbarth. Slicing Droids: Program Slicing for Smali Code. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1844–1851, New York, NY, USA, 2013. ACM.

[47] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. These aren't the droids you're looking for:

Retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[48] IBM. Watson libraries for analysis.

[49] IDC Press Release. Smartphone os marketshare, 2016.

[50] IPaddressAPI.com. Ipaddressapi.com, 2017.

[51] Chetan Jaiswal and Vijay Kumar. Igod: Identification of geolocation of cloud datacenters. In *Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th*, pages 665–672. IEEE, 2015.

[52] Cheol Jeon, WooChur Kim, Bongjae Kim, and Yookun Cho. Enhancing security enforcement on unmodified android. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1655–1656. ACM, 2013.

[53] Jinseong Jeon, Kristopher K Micinski, Jeffrey A Vaughan, Ari Fogel, Nikhilesh Reddy, Jeffrey S Foster, and Todd Millstein. Dr. android and mr. hide: fine-grained permissions in android applications. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM, 2012.

[54] Ari Juels and Burton S. Kaliski, Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 584–597, New York, NY, USA, 2007. ACM.

[55] Seny Kamara and Kristin Lauter. Cryptographic cloud storage. In *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin Heidelberg, 2010.

[56] Ethan Katz-Bassett, John P John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards ip geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84. ACM, 2006.

[57] Jinyung Kim, Yongho Yoon, Kwangkeun Yi, Junbum Shin, and SWRD Center. ScanDal: Static analyzer for detecting privacy leaks in android applications. *MoST*, 12, 2012.

[58] Christoph Krauß and Volker Fusenig. Using trusted platform modules for location assurance in cloud networking. In *International Conference on Network and System Security*, pages 109–121. Springer, 2013.

[59] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. IccTA: Detecting inter-component privacy leaks in Android apps. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, pages 280–291. IEEE Press, 2015.

[60] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. Chex: statically vetting android apps for component hijacking vulnerabilities. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 229–240. ACM, 2012.

[61] P. Massonet, S. Naqvi, C. Ponsard, J. Latanicki, B. Rochwerger, and M. Villari. A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1510–1517. IEEE, 2011.

[62] MaxMind.Inc. Freegeoip.net, 2014.

[63] Peter Mell and Tim Grance. The nist definition of cloud computing, 2011.

[64] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM symposium on information, computer and communications security*, pages 328–332. ACM, 2010.

[65] A Noman and C. Adams. Dlas: Data location assurance service for cloud computing environments. In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pages 225–228. IEEE, 2012.

[66] Ali Noman and Carlisle Adams. Providing a data location assurance service for cloud storage environments. *J. Mob. Multimed.*, 8(4):265–286, 2012.

[67] OASIS-Standard. extensible access control markup language (xacml) version 2.0, 2005.

[68] Damien Octeau, Patrick McDaniel, Somesh Jha, Alexandre Bartel, Eric Bodden, Jacques Klein, and Yves Le Traon. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis. *Effective Inter-Component Communication Mapping in Android with Epicc: An Essential Step Towards Holistic Security Analysis*, 2013.

[69] Venkata N Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for internet hosts. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 173–185. ACM, 2001.

[70] Nicolae Paladi and Antonis Michalas. one of our hosts in another country: Challenges of data geolocation in cloud storage. In *Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2014 4th International Conference on*, pages 1–6. IEEE, 2014.

[71] European Parliament and of the Council. General data protection regulation, final version dated 27 april 2016. *Online at http://data.europa.eu/eli/reg/2016/679/oj*, 2016.

[72] Zachary N. J. Peterson, Mark Gondree, and Robert Beverly. A position paper on data sovereignty: The importance of geolocating data in the cloud. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, Berkeley, CA, USA, 2011. USENIX Association.

[73] PHP. fsockopen function, 2013.

[74] Vaibhav Rastogi, Yan Chen, and William Enck. AppsPlayground: automatic security analysis of smartphone applications. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 209–220. ACM, 2013.

[75] Jason Reid, Juan M. Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2Nd ACM Symposium on Information, Computer and Communications Security*, ASIACCS '07, pages 204–213, New York, NY, USA, 2007. ACM.

[76] European Parliament. Directive 95/46/ec of the european parliament and of the Council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free move-

ment of such data. *http://eur-lex.europa.eu/eli/dir/1995/46/oj*, 1995.

[77] IDC Press Release. Worldwide smartphone market will see the first single-digit growth year on record, according to idc. *http://www.idc.com/getdoc.jsp?containerId=prUS40664915*, 2015.

[78] Brian Cantwell Smith. *Procedural Reflection in Programming Languages*. PhD thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 1982.

[79] David Sounthiraraj, Justin Sahs, Garret Greenwood, Zhiqiang Lin, and Latifur Khan. Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps. In *In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14*. Citeseer, 2014.

[80] The Tcpdump Group. TCP-Dump, 2015.

[81] Connor Tumbleson and Ryszard Winiewski. APK tool - a tool for reverse engineering android apk files, 2016.

[82] Raja Vallée-Rai, Phong Co, Etienne Gagnon, Laurie Hendren, Patrick Lam, and Vijay Sundaresan. Soot-a Java bytecode optimization framework. In *Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research*, page 13. IBM Press, 1999.

[83] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. Towards street-level client-independent ip geolocation. In *NSDI*, volume 11, pages 27–27, 2011.

[84] Gaven J Watson, Reihaneh Safavi-Naini, Mohsen Alimomeni, Michael E Locasto, and Shivaramakrishnan Narayan. Lost: location

based storage. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, pages 59–70. ACM, 2012.

[85] Fengguo Wei, Sankardas Roy, Xinming Ou, et al. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1329–1341. ACM, 2014.

[86] Wikipedia. Polynomial regression, 2014.

[87] Wikipedia. Trilateration, 2014.

[88] Wikipedia. Geographic coordinate system. *Online at https://en.wikipedia.org/wiki/Geographic_coordinate_system*, 2016.

[89] Wikipedia. Triangle. *Online at https://en.wikipedia.org/wiki/Triangle*, 2016.

[90] Wikipedia. Triangulation. *Online at https://en.wikipedia.org/wiki/Triangulation*, 2016.

[91] Xposed. Xposed framework. *Online at http://repo.xposed.info/*, 2017.

[92] Rubin Xu, Hassen Saïdi, and Ross J Anderson. Aurasium: practical policy enforcement for android applications. In *USENIX Security Symposium*, volume 2012, 2012.

[93] Zhemin Yang, Min Yang, Yuan Zhang, Guofei Gu, Peng Ning, and X Sean Wang. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1043–1054. ACM, 2013.

[94] Sara Zaske. Germany's privacy leaders gather to discuss suspending us safe harbor. *http://www.zdnet.com/article/germanys-privacy-leaders-gather-to-discuss-suspending-us-safe-harbo* 2015.

[95] Yury Zhauniarovich, Maqsood Ahmad, Olga Gadyatskaya, Bruno Crispo, and Fabio Massacci. Stadyna: addressing the problem of dynamic code updates in the security analysis of android applications. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 37–48. ACM, 2015.

[96] Cong Zheng, Shixiong Zhu, Shuaifu Dai, Guofei Gu, Xiaorui Gong, Xinhui Han, and Wei Zou. Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 93–104. ACM, 2012.

[97] Yibing Zhongyang, Zhi Xin, Bing Mao, and Li Xie. DroidAlarm: an all-sided static analysis tool for Android privilege-escalation malware. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 353–358. ACM, 2013.